การเขียนโปรแกรม

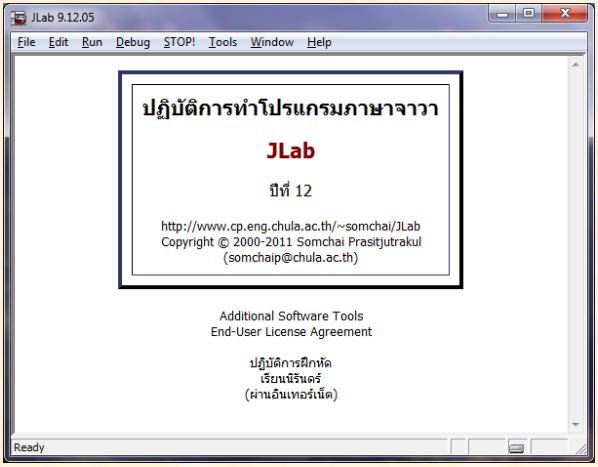
แบบฝึกปฏิบัติ : ฉบับวาจาจาวา

สมชาย ประสิทธิ์จูตระกูล

ภาควิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์
จุฬาลงกรณ์มหาวิทยาลัย

การเขียนโปรแกรม

แบบฝึกปฏิบัติ : ฉบับวาจาจาวา



สมชาย ประสิทธิ์จูตระกูล

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

จุฬาลงกรณ์มหาวิทยาลัย

การเขียนโปรแกรม แบบฝึกปฏิบัติ : ฉบับวาจาจาวา / สมชาย ประสิทธิ์จูตระกูล

1. การเขียนโปรแกรม (คอมพิวเตอร์)
2. จาวา (คอมพิวเตอร์)

005.133

ISBN 978-616-551-428-6

พิมพ์ครั้งที่ 1 (ธันวาคม 2554)

จัดพิมพ์โดย
ภาควิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์
จุฬาลงกรณ์มหาวิทยาลัย
พญาไท กรุงเทพฯ 10330
http://www.cp.eng.chula.ac.th

คำนำ

วิชา 2110101 การเขียนโปรแกรมคอมพิวเตอร์เป็นหนึ่งในวิชาพื้นฐานทางวิศวกรรมศาสตร์ ที่นิสิตชั้นปีที่ 1 ทุกคนของคณะวิศวกรรมศาสตร์ต้องลงทะเบียนเรียน วัตถุประสงค์หลักของวิชานี้ คือ ให้นิสิตเข้าใจหลักการในการใช้คำสั่งต่าง ๆ ของภาษาโปรแกรม เพื่อเขียนโปรแกรมคอมพิวเตอร์ให้ตรงตามข้อกำหนดที่ได้รับ การเขียนโปรแกรมเป็นความสามารถที่ต้องลงมือฝึกปฏิบัติด้วยตนเอง เหมือนกับทักษะอื่นๆ ทางวิศวกรรมที่จำเป็นต้องฝึก ๆ ๆ จึงจะเห็นผล ไม่สามารถได้มาด้วยการอ่าน ๆ ๆ

แบบฝึกปฏิบัติการเขียนโปรแกรมเล่มนี้ถูกจัดทำขึ้น เพื่อให้นิสิตได้ศึกษาเนื้อหาและเตรียมตัวก่อนเข้าเขียนโปรแกรมในห้องปฏิบัติการที่จัดขึ้นเป็นกิจกรรมเสริมการเรียนรายสัปดาห์ โดยแบบฝึกปฏิบัติการชุดต่าง ๆ มีตัวตรวจที่ใช้กับซอฟต์แวร์ JLab เพื่อตรวจสอบความถูกต้องของโปรแกรมอย่างอัตโนมัติ นิสิตจะได้ฝึกเขียน แก้ปัญหา หาที่ผิด ตรวจสอบความถูกต้องของโปรแกรมที่ได้พัฒนาขึ้น หากมีปัญหาใดที่ต้องการคำอธิบายเพิ่มเติมระหว่างการเขียนโปรแกรมในห้องปฏิบัติการ นิสิตสามารถสอบถามปัญหาได้กับพี่นิสิตช่วยสอน นอกจากนี้ แบบฝึกปฏิบัติการเล่มนี้ยังมีแบบฝึกหัดเพิ่มเติมให้นิสิตได้ฝึกทำโจทย์เสริมอื่น ๆ ด้วย ในกรณีที่นิสิตต้องการฝึกปฏิบัติเพิ่มเติมสามารถติดตั้ง JLab ที่เครื่องคอมพิวเตอร์ส่วนตัวจาก http://www.cp.eng.chula.ac.th/~somchai/JLab และสำหรับผู้สนใจศึกษารายละเอียดของคำสั่ง การใช้งาน รวมทั้งตัวอย่างโปรแกรมมากมาย สามารถอ่านได้จากหนังสือ “เริ่มเรียนเขียนโปรแกรม : ฉบับวาจาจาวา”

ผู้เขียนต้องขอขอบคุณผู้บริหารภาควิชาวิศวกรรมคอมพิวเตอร์ที่ให้การสนับสนุนการพัฒนาซอฟต์แวร์ JLab และแบบฝึกปฏิบัติการเขียนโปรแกรมคอมพิวเตอร์ ที่ใช้เสริมการสอนวิชาการเขียนโปรแกรมมาตั้งแต่ปีการศึกษา 2545 ขอขอบคุณบุคลากรของศูนย์คอมพิวเตอร์ คณะวิศวฯ ที่ให้บริการทั้งเครื่องปฏิบัติการและเครื่องแม่ข่ายอย่างราบรื่นตั้งแต่ปี พ.ศ. 2545 ขอขอบคุณคณาจารย์ที่ร่วมกันสอนและปรับปรุงวิชา 2110101 ตลอดมา และท้ายสุดที่ต้องขอบคุณก็คือ นิสิตคณะวิศวฯ กว่าเจ็ดพันคนใน 9 ปีที่ผ่านมาที่ช่วยกันใช้ (และที่ถูกบังคับให้ใช้ ☺) ได้พบข้อบกพร่องของระบบ และนำมาแก้ไขปรับปรุงจนสามารถใช้งานได้อย่างมีเสถียรภาพ

รศ. ดร. สมชาย ประสิทธิ์จูตระกูล
ภาควิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์
จุฬาลงกรณ์มหาวิทยาลัย
๕ ธันวาคม ๒๕๕๔

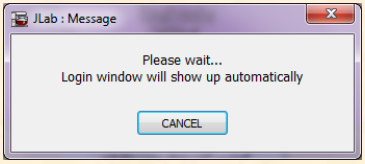

bird.gif

สารบัญ

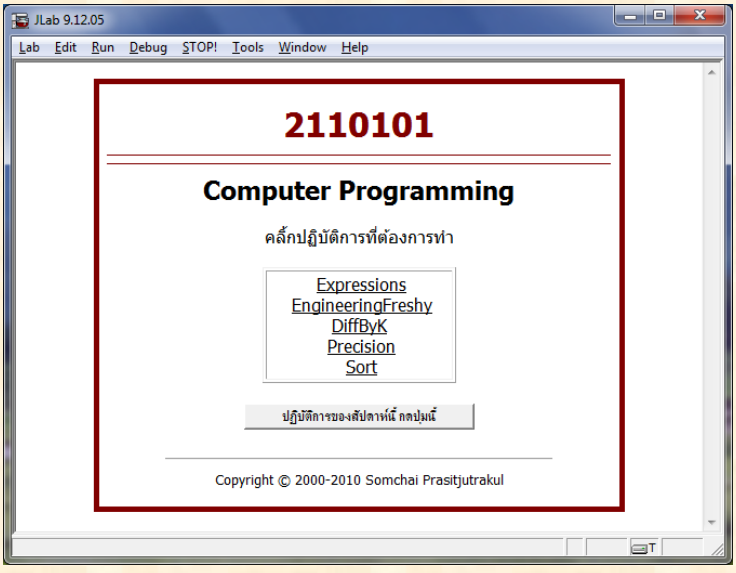

v 9.12.05
สมชาย ประสิทธิ์จูตระกูล

ปฏิบัติการชุดนี้อาศัยซอฟต์แวร์ JLab ในการเขียนและตรวจผลลัพธ์ ที่ได้รับการติดตั้งในห้องปฏิบัติการ ศูนย์คอมพิวเตอร์ คณะวิศวกรรมศาสตร์ หัวข้อนี้นำเสนอวิธีการใช้งานที่เพียงพอกับการทำปฏิบัติการของวิชาการเขียนโปรแกรมเบื้องต้น (ในกรณีที่ต้องการติดตั้ง JLab เพื่อใช้งานที่เครื่องคอมพิวเตอร์ส่วนตัว ขอให้อ่านรายละเอียดการติดตั้งในภาคผนวกของหนังสือ "เริ่มเรียนเขียนโปรแกรม : ฉบับวาจาจาวา")

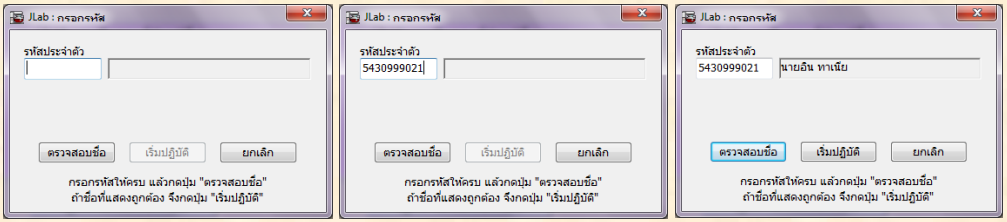
JLab ทำงานสองแบบ แบบปกติมีไอคอนเป็นรูป  และแบบใช้ในห้องปฏิบัติการมีไอคอนเป็นรูป  (JLab-online) อย่าลืมว่า ในการทำปฏิบัติการการเขียนโปรแกรมของวิชา 2110101 เราใช้แบบหลัง หลังจากเปิดเครื่องและใส่เลขประจำตัวและรหัสผ่านเพื่อเข้าใช้วินโดว์เรียบร้อยแล้ว ให้คลิ้กสองครั้งที่ไอคอน JLab-online บนเดสก์ท็อปเพื่อสั่งทำงาน ในกรณีไม่มีไอคอนบนเดสก์ท็อป ให้เรียกจากเมนู Start → All Programs → JLab → JLab-Online (Lab) เมื่อสั่งให้ทำงานแล้ว JLab จะเชื่อมต่อเข้าสู่เครื่องแม่ข่าย โดยแสดงวินโดว์ข้างล่างนี้ในช่วงที่กำลังติดต่อกับเครื่องแม่ข่าย



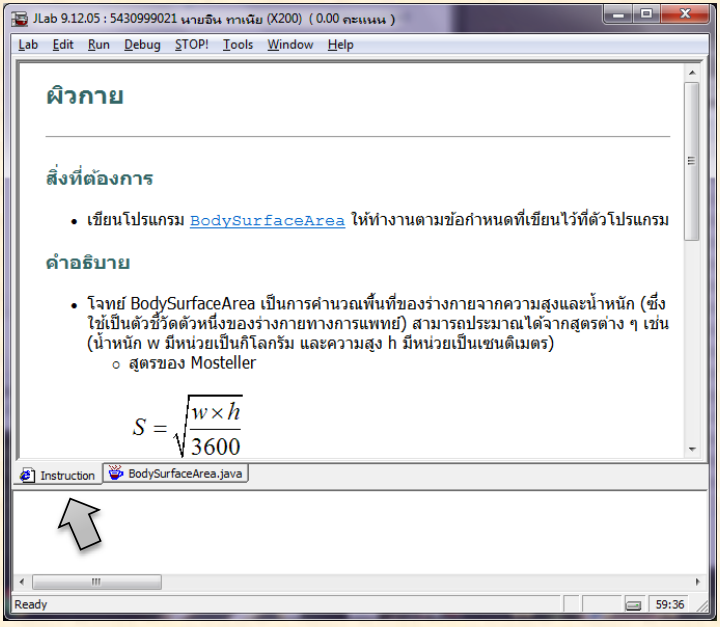
หากยังไม่อยู่ในช่วงเวลาที่พร้อมให้บริการ ก็จะแสดงวินโดว์นี้ค้าง โปรแกรมจะตรวจสอบความพร้อมเป็นระยะๆ ถ้าพร้อมเมื่อไรก็จะแสดงหน้าแรกข้างล่างนี้ (อาจแตกต่างจากที่แสดงนี้บ้าง)

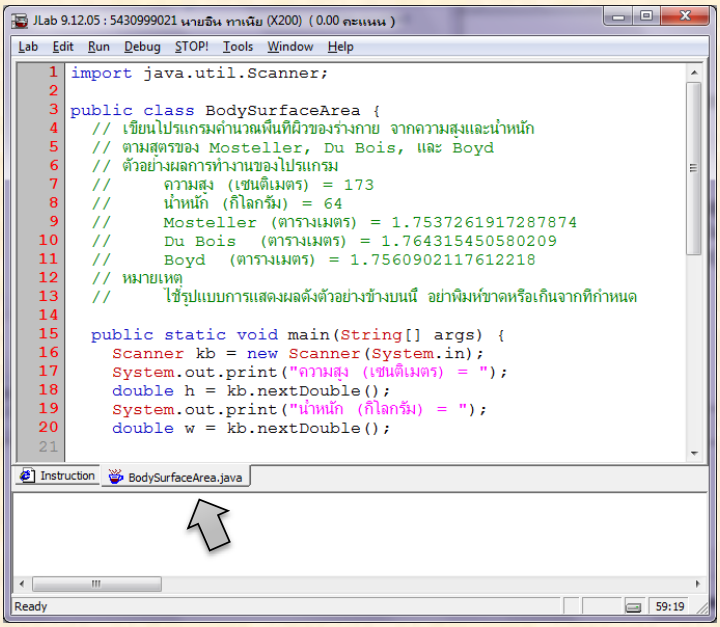


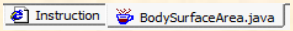
ให้กดปุ่มหรือคลิ้กที่ข้อความแสดงหัวข้อปฏิบัติการที่สนใจทำ อาจมีหัวข้อที่ให้ทำเป็นรายสัปดาห์ หรืออาจมีหัวข้อที่ให้ทำย้อนหลัง เมื่อกดหรือคลิ้กแล้ว ระบบจะแสดงวินโดว์ให้กรอกรหัสนิสิต กรอกแล้วกดปุ่ม ตรวจสอบชื่อ ระบบจะแสดงชื่อสกุล หากทุกอย่างถูกต้อง ก็กดปุ่ม เริ่มปฏิบัติ



JLab จะติดต่อกับเครื่องแม่ข่ายเพื่อนำชุดปฏิบัติการเข้าสู่ระบบ เมื่อทุกอย่างเรียบร้อย จะแสดงโจทย์ดังตัวอย่าง

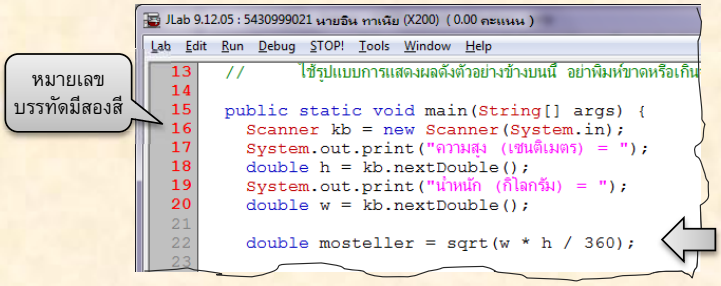




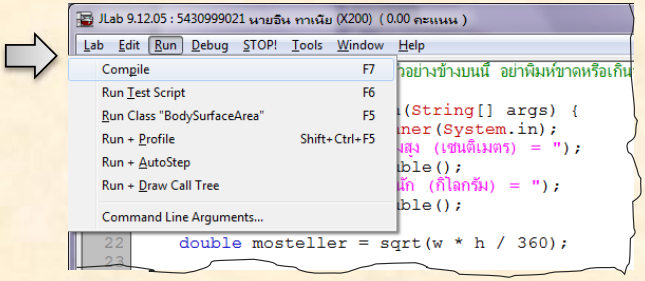
ชุดปฏิบัติการหนึ่งจะมีหน้าโจทย์กับหน้าที่แสดงคลาส (ถ้ามีหลายคลาส ก็มีหลายหน้า) ผู้ใช้สามารถเปลี่ยนหน้าได้ด้วยการกดที่แผ่นแสดงชื่อหน้า 

เมื่ออ่านโจทย์จนเข้าใจว่าให้ทำอะไรแล้ว ก็เลือกหน้าคลาสที่ต้องการแก้ไข ให้สังเกตที่หมายเลขบรรทัดกำกับคำสั่ง จะมีสองสี สีแดงกับสีเทา บรรทัดใดที่มีหมายเลขสีแดงกำกับคือบรรทัดที่ไม่สามารถแก้ไขได้ ส่วนบรรทัดที่มีหมายเลขบรรทัดเป็นสีเทานั้นแก้ไขเพิ่มเติมได้ เหตุที่ต้องป้องกันบางบรรทัดไม่ให้แก้ไข ก็เพื่อลดปัญหาในการเขียนโปรแกรม ชุดคำสั่งส่วนนี้เป็นส่วนที่ผู้เขียนโจทย์เตรียมไว้ให้ล่วงหน้า เช่น การประกาศคลาส การอ่านข้อมูลขาเข้า หรือการแสดงผลลัพธ์ เป็นต้น

โจทย์ของตัวอย่างที่แสดงในรูปก่อนหน้านี้ ต้องการให้เขียนคำสั่งเพื่อคำนวณพื้นที่ผิวของร่างกาย ตามความสูงและน้ำหนักที่ได้รับทางแป้นพิมพ์ ด้วยสูตร 3 สูตร ขอแสดงให้ดูสัก 1 สูตร รูปข้างล่างนี้แสดงตัวอย่างที่เพิ่มคำสั่งคำนวณตามสูตรแรก



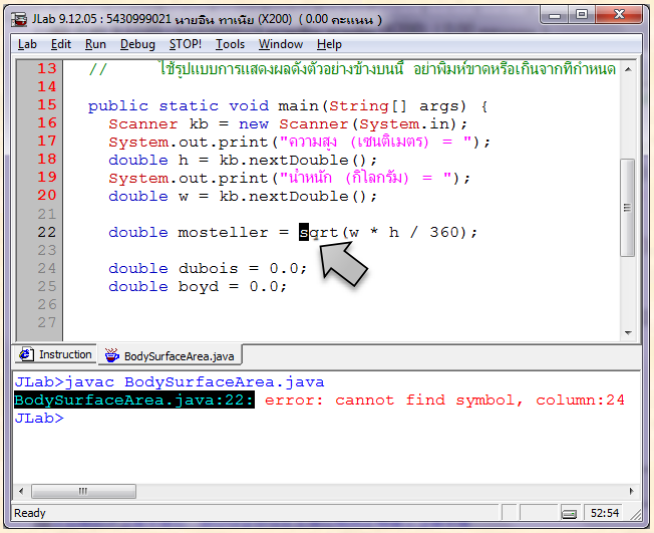
ถ้ามั่นใจว่าเขียนถูกต้อง ก็ลองสั่งให้ JLab ช่วยแปล (compile) โปรแกรม เพื่อตรวจว่าเขียนถูกต้องตามไวยากรณ์ของภาษาจาวาหรือไม่ การสั่งให้แปลนี้กระทำได้โดยเลือกเมนู Run → Compile หรือกดปุ่ม F7 ก็ง่ายดี ดังรูป



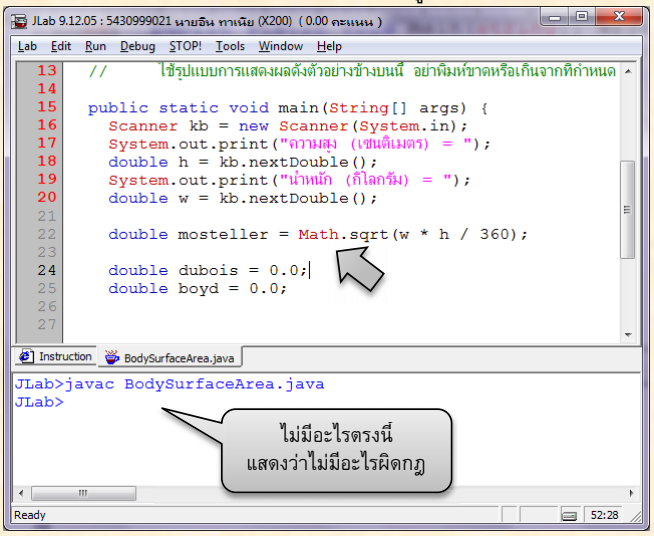
จากตัวอย่าง คำสั่งในบรรทัดที่ 22 **`double mosteller = sqrt(w * h / 360);`** ที่ป้อนเข้าไปนั้นผิด ในที่นี้ผิดที่คำว่า **`sqrt`** เพราะตัวแปลไม่สามารถเข้าใจว่าคืออะไร ไม่ใช่คำสั่งในภาษา และก็ไม่ใช้บริการที่ระบบรู้จัก ตัวแปลโปรแกรมจะแสดงคำอธิบายถึงข้อผิดพลาดที่พบ ดังแสดงในรูปต่อไปนี้ ที่บริเวณด้านล่างของวินโดว์ JLab บริเวณนี้มีไว้เพื่อแสดงผลลัพธ์จากการสั่งแปล แสดงผลลัพธ์ในการทำงานของโปรแกรม และยังเป็นบริเวณที่แสดงผลลัพธ์จากการตรวจความถูกต้องในการทำงานของโปรแกรมด้วย ในตัวอย่างนี้ ตัวแปลแสดง

`BodySurfaceArea.java:22 error: cannot find symbol, column:24`

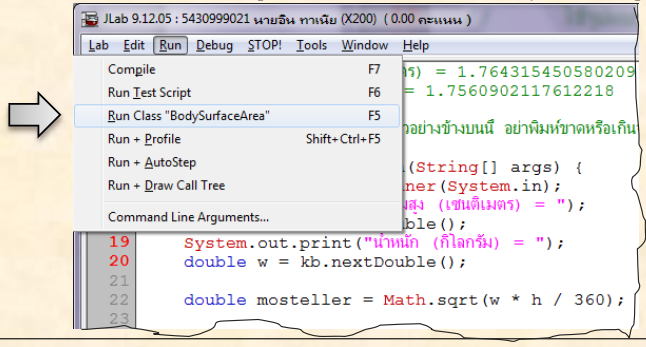
หมายความว่า พบความผิดพลาดที่บรรทัดที่ 22 คอลัมน์ที่ 24 ของคลาส **`BodySurfaceArea.java`** โดยอธิบายว่า ผิดเพราะ **`cannot find symbol`** เนื่องจากที่คอลัมน์ 24 มีคำว่า **`sqrt`** ที่เขาไม่รู้จักนั่นเอง หากดูในรูป ระบบจะแสดงตัวพื้นดำให้เห็นเด่นชัดที่บรรทัดที่ 22 คอลัมน์ 24 ตรงตามตัวแปลโปรแกรมพบความผิดพลาดด้วย



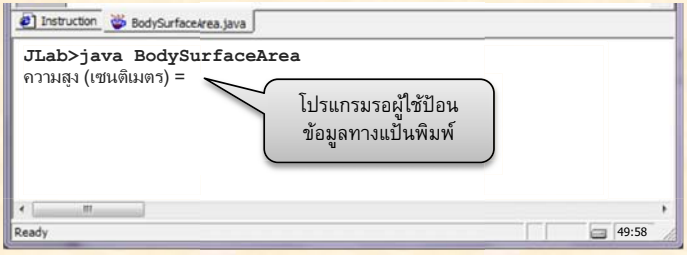
คิดเล็กน้อยก็รู้ว่าเขียน **sqrt** ไม่ได้ ต้องเขียน **Math.sqrt** จึงจะถูกต้อง ก็แก้ไขแล้วสั่งแปลใหม่ ได้ดังรูป



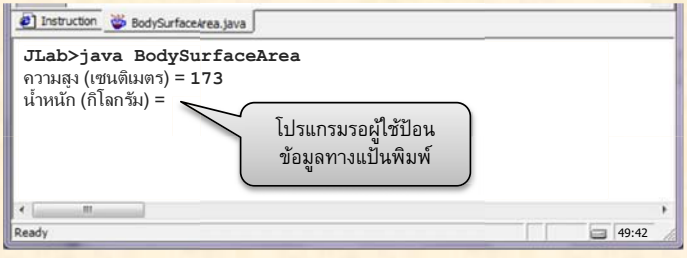
ตัวแปลไม่ได้แจ้งอะไรว่าผิดกฎ แต่ยังไม่รู้ว่าจะทำงานถูกต้องหรือไม่ ต้องสั่งทำงานดูจึงจะรู้ว่า ถูกต้องตามความต้องการของโจทย์หรือไม่ เราสั่งทำงานด้วยเมนู Run → Run Class หรือกดปุ่ม F5 ดังรูป



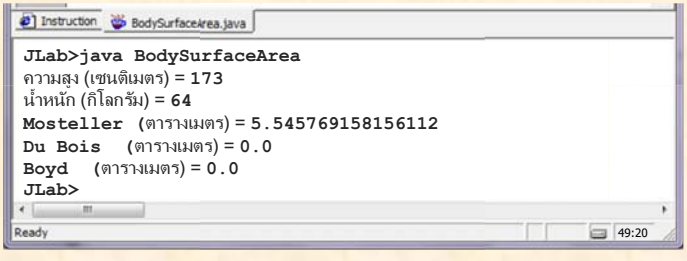
การสั่งโปรแกรมทำงานที่คลาสใด ก็คือการสั่งให้เมท็อด **main** ของคลาสนั้นทำงาน (จึงต้องเลือกหน้าของคลาสที่ต้องการให้ทำงานก่อน แล้วสั่งทำงาน) และที่สำคัญคือ ต้องรู้ว่า เมื่อสั่งทำงานแล้วโปรแกรมทำอะไร จะได้คาดว่าต้องทำอะไร ป้อนข้อมูลอะไร จะได้ผลลัพธ์อะไร ในกรณีของคลาส **BodySurfaceArea** โปรแกรมจะแสดงข้อความและรอรับความสูงและน้ำหนักจากผู้ใช้ทางแป้นพิมพ์ คำนวณพื้นที่ผิวของร่างกายตามสูตร 3 สูตร และแสดงผลลัพธ์ที่คำนวณได้จากสูตรทั้งสาม เมื่อสั่ง **main** ของ **BodySurfaceArea** ทำงาน จะแสดงข้อความดังรูปข้างล่างนี้ (บรรทัดแรกที่แสดงว่า JLab>java BodySurfaceArea เป็นข้อความจากระบบ ไม่ใช่จากตัวโปรแกรม)



หลังแสดงข้อความ ความสูง (เซนติเมตร) = โปรแกรมที่เขียนจะรอให้ผู้ใช้ป้อนความสูง ในที่นี้เราป้อน 173 แล้วกดปุ่ม enter ↵ โปรแกรมก็ทำงานต่อ แสดงข้อความใหม่เพื่อรอรับน้ำหนักจากผู้ใช้ดังรูป

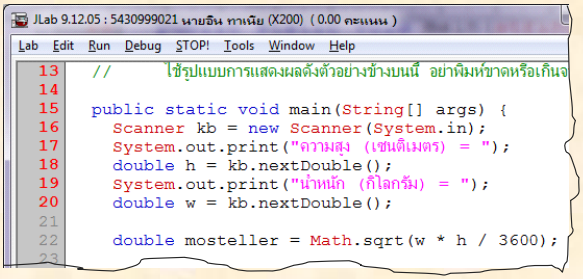


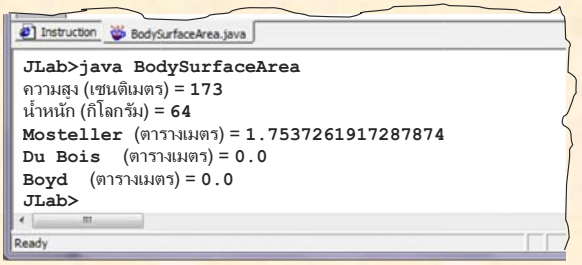
ในที่นี้เราป้อน 64 แล้วกดปุ่ม enter ↵ โปรแกรมทำงานต่อ คำนวณ และแสดงผลลัพธ์ดังรูป



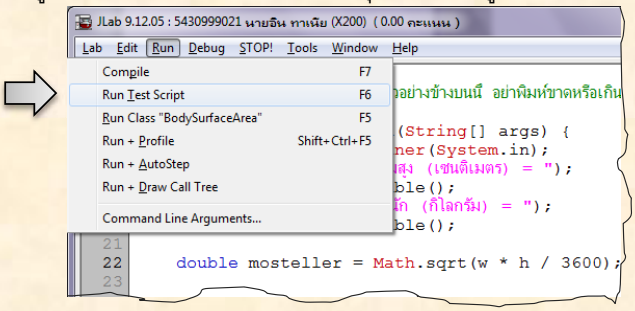
เนื่องจากเราเพิ่งเขียนไปแค่สูตรเดียว ก็ตรวจสอบความถูกต้องเฉพาะสูตรแรก แล้วจะรู้ได้อย่างไรว่าถูกต้อง ก็คงต้องคำนวณเองด้วยเครื่องคำนวณตามสูตร แล้วตรวจดู สูตรแรกคือ $\sqrt{\frac{w \times h}{3600}} = \sqrt{\frac{64 \times 173}{3600}} \approx 1.7537$ ไม่ตรงกับที่แสดงซึ่งคือ 5.545769158156112

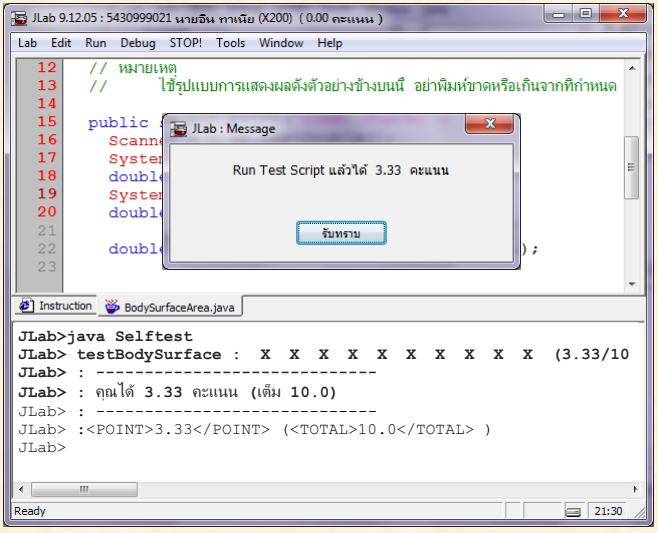
เมื่อพบว่าทำงานไม่ถูกต้อง ก็ต้องกลับไปดูที่โปรแกรมตรงคำสั่งที่เขียน ดูดี ๆ ที่บรรทัดที่ 22 จะพบว่า เขียนตัวเลขในสูตรคำนวณผิด เขียนว่า 360 แต่ควรเป็น 3600 เมื่อเห็นเช่นนี้ ก็แก้ไขให้ถูกต้อง แล้วลองสั่งทำงานใหม่ ป้อนความสูงและน้ำหนัก แล้วดูผลที่ได้ ตามรูปต่อไปนี้



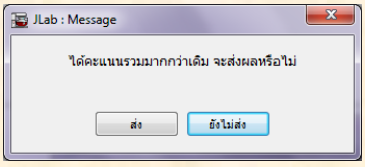


ครั้งนี้ได้ผลลัพธ์ที่ตรงกับที่คำนวณเอง เมื่อมั่นใจว่า น่าจะเขียนได้ถูกต้อง ก็ลองให้ตัวตรวจตรวจให้คะแนนดู (ถึงแม้จะยังเขียนไม่เสร็จครบทั้ง 3 สูตร เขียนเสร็จแค่สูตรแรก ก็เรียกตัวตรวจลองตรวจได้) การเรียกให้ตัวตรวจทำงานทำได้จากเมนู Run → Run Test Script หรือกดปุ่ม F6 แล้วดูผลการตรวจ ดังตัวอย่างข้างล่างนี้



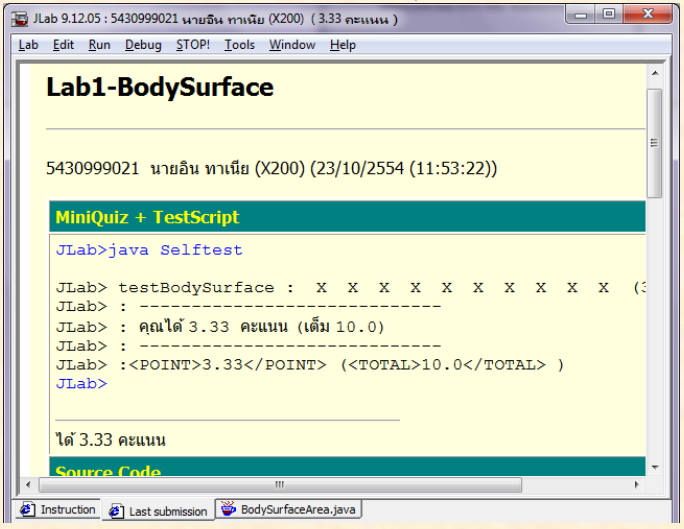


จากตัวอย่าง แสดงให้เห็นว่า ได้ 3.33 คะแนน จากคะแนนเต็ม 10 ก็น่าจะเดาได้ว่า ที่เขียนคำนวณสูตรแรกนั้นถูกต้อง จึงได้คะแนน 1 ใน 3 ของคะแนนเต็ม ตัวตรวจจะแสดงวินโดว์ให้รับทราบคะแนนที่ได้ เมื่อผู้ใช้กดปุ่มรับทราบ ระบบจะแสดงอีกวินโดว์ คราวนี้จะถามว่า ต้องการส่งหรือไม่ ดังรูปข้างล่างนี้

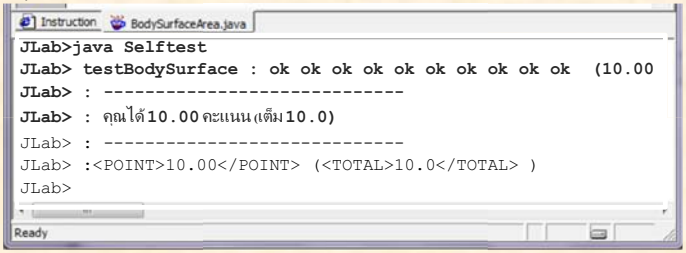


ทั้งนี้จะถามเช่นนี้ก็เมื่อการตรวจครั้งล่าสุดได้คะแนนมากกว่าคะแนนมากสุดในอดีตที่ได้มาของปฏิบัติการนี้ การกดส่งทำให้ JLab บันทึกผลและโปรแกรมที่เขียนไว้ในเครื่องแม่ข่าย ระบบไม่ได้จำกัดจำนวนครั้งของการส่ง ผู้ใช้สามารถส่งได้เสมอ (และแนะนำให้ส่งทุกครั้งที่ได้คะแนนเพิ่ม เผื่อกรณีฉุกเฉินที่เกิดปัญหากับระบบ ระบบจะสามารถนำชุดปฏิบัติการที่เคยส่งมาทำต่อได้) หรือจะเลือกไม่ส่ง ก็ไม่เป็นไร แต่อย่าลืมส่งบ้าง จะได้มีคะแนนเก็บไว้เป็นหลักฐาน (หากทำจนเสร็จ โดยไม่ส่งเลย เครื่องแม่ข่ายก็จะไม่มีคะแนนการตรวจเก็บไว้แต่อย่างใด)

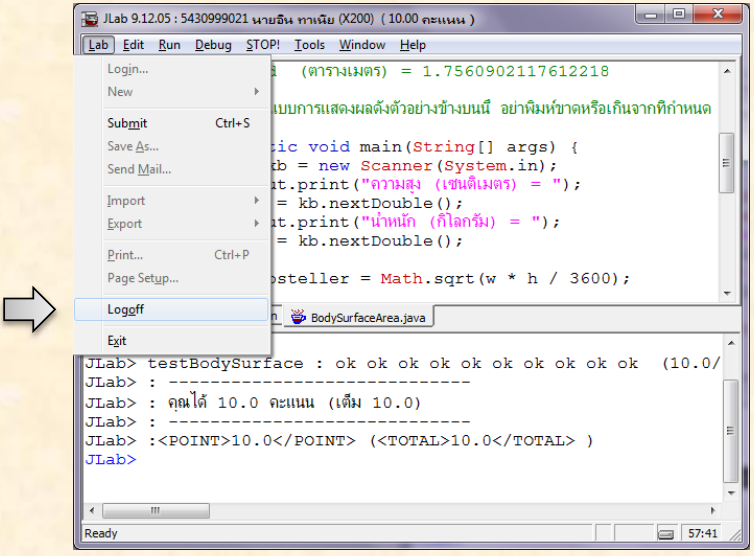
ในกรณีที่มีการส่งผลการตรวจ ระบบจะแสดงผลการตรวจที่ได้ส่งครั้งล่าสุดเพิ่มให้อีกหน้าหนึ่ง (Last submission) ดังรูปข้างล่างนี้ (สังเกตที่หัววินโดว์ของ JLab แสดงคะแนนมากที่สุดของโจทย์ข้อนี้เท่าที่ได้ตรวจมา)



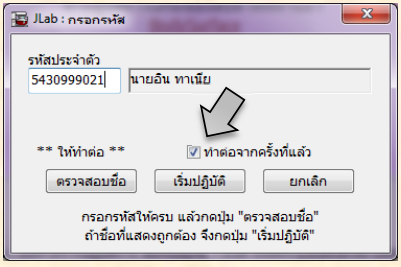
ก็คงต้องเขียนโปรแกรมต่อ ให้ทำตามที่โจทย์ต้องการจนครบถ้วน รูปข้างล่างนี้แสดงตัวอย่างของผลการตรวจที่ทำงานได้ถูกต้องในทุกกรณีของการตรวจ และได้คะแนนเต็ม และก็ควรส่งเมื่อได้คะแนนเต็ม จริงไหม ?



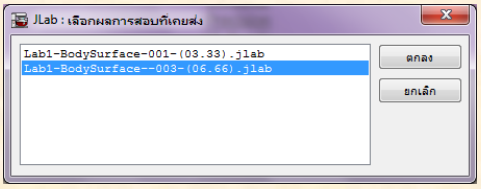
ในกรณีที่ไม่ได้ส่งตอนที่ระบบแจ้งให้ส่ง แต่ต้องการส่งทีหลัง ก็ให้เลือกเมนู Lab → Submit เมื่อเขียนได้ถูกต้อง ตรวจแล้วได้คะแนนเต็ม และส่งผลเรียบร้อย เป็นอันเสร็จสิ้นปฏิบัติการการเขียนโปรแกรม สามารถเลิกการทำงานได้ด้วยการเลือกเมนู Lab → Logoff หรือ Lab → Exit เป็นการจบการทำงานของ JLab



หากออกจาก JLab โดยที่ยังทำไม่เสร็จ แล้วเรียก JLab ทำงานใหม่ (จะเป็นในวันเดียวกัน หรือวันหลังก็ตาม) เมื่อเลือกทำปฏิบัติการเดิมที่เคยทำ ในวินโดว์ที่ JLab ให้ป้อนเลขประจำตัว เมื่อกดปุ่ม ตรวจสอบชื่อ JLab จะแสดงทางเลือกว่า จะทำต่อจากโปรแกรมล่าสุดครั้งที่แล้วหรือไม่ รูปข้างล่างนี้แสดงตัวอย่างการเลือกแบบทำต่อจากครั้งที่แล้ว



แต่ถ้าไม่เลือกทำต่อจากครั้งที่แล้ว หมายความว่า ไม่ต้องการครั้งล่าสุด แต่ต้องการเลือกเองจากชุดปฏิบัติการที่เคยส่งในอดีต หลังกดปุ่มเริ่มปฏิบัติ JLab จะแสดงอีกวินโดว์ให้ผู้ใช้เลือกทำต่อจากชุดที่เคยส่ง ดังตัวอย่างในรูปข้างล่างนี้



คำสั่งที่ใช้บ่อยใน JLab

- ปุ่มลัด
  - F1 เปิดแฟ้มช่วยเหลือ ให้ค้นและแสดงเมท็อดของคลาสมาตรฐานในระบบจาวา
  - F5 สั่งเมท็อด main ของคลาสที่แสดงอยู่เริ่มทำงาน
  - F6 สั่งให้ตัวตรวจเริ่มทำงาน (ในกรณีที่เปิดโจทย์ปฏิบัติการ ซึ่งมีตัวตรวจให้มาด้วย)
  - F7 สั่งตัวแปล (compiler) ทำงาน
  - ctrl-C copy ข้อความที่เลือกลงคลิปบอร์ด
  - ctrl-X cut ข้อความที่เลือกออก และนำลงคลิปบอร์ด
  - ctrl-V paste ข้อความจากคลิปบอร์ดลงในโปรแกรม
  - ctrl-Z undo การแก้ไขที่เพิ่งทำ
  - ctrl-Y redo การแก้ไขที่เพิ่ง undo
- เมนูคำสั่ง
  - Lab → Login ติดต่อกับเครื่องแม่ข่าย เพราะเข้าใช้ระบบ
  - Lab → Logoff เลิกทำปฏิบัติการ แต่ JLab ยังทำงานอยู่
  - Lab → Exit ปิดการทำงานของ JLab อยู่
  - Lab → Submit บันทึกชุดปฏิบัติการพร้อมคะแนนครั้งล่าสุดไว้ที่เครื่องแม่ข่าย
  - Edit → Comment ทำให้บรรทัดที่เลือกไว้เป็น comment (โดยเติม // ไว้ด้านหน้า)
  - Edit → Uncomment ลบ // ออกจากบรรทัดที่เลือกไว้
  - STOP สั่งให้โปรแกรมที่ทำงานอยู่ หยุดทำงานทันที (มักใช้เมื่อโปรแกรมทำงานนานเกินไป ซึ่งอาจเกิดจากเหตุที่โปรแกรมทำงานในวงวนแบบไม่สิ้นสุด)

ปฏิบัติการที่ 1 : ผิวกาย

ผลการเรียนรู้

- เขียนจำนวนจริงแบบสัญกรณ์ทางวิทยาศาสตร์
- เลือกใช้ประเภทของตัวแปรให้เหมาะสม
- เขียนนิพจน์ให้คำนวณตามสูตรคณิตศาสตร์ที่กำหนดให้

เนื้อหา

การคำนวณพื้นที่ของผิวกายจากความสูงและน้ำหนัก (ซึ่งใช้เป็นตัวชี้วัดตัวหนึ่งของร่างกายทางการแพทย์) สามารถประมาณได้จากสูตรต่างๆ ดังนี้ (น้ำหนัก w มีหน่วยเป็นกิโลกรัม และความสูง h มีหน่วยเป็นเซนติเมตร)

สูตรของ Mosteller $$S = \sqrt{\frac{w \times h}{3600}}$$

สูตรของ Du Bois $$S = \frac{71.84 \times w^{0.425} \times h^{0.725}}{10^4}$$

สูตรของ Boyd $$S = 3.207 \times 10^{-4} \times h^{0.3} \times (1000w)^{(0.7285 - 0.0188(3 + log_{10} w))}$$

แหล่งอ้างอิง : http://en.wikipedia.org/wiki/Body_surface_area

สิ่งที่ต้องเขียน

รหัสต้นฉบับเริ่มต้น (**`BodySurfaceArea.java`**) ที่ให้มามีคำสั่งแสดงและรับข้อมูลขาเข้าจากผู้ใช้ พร้อมกับคำสั่งแสดงผลลัพธ์ให้เรียบร้อยแล้ว ให้เขียนเฉพาะคำสั่งคำนวณพื้นที่ผิวด้วยสูตรทั้งสามจากข้อมูลขาเข้าให้ได้ผลลัพธ์ตามที่กำหนดให้ดังนี้

ข้อมูลขาเข้า (รับจากแป้นพิมพ์)

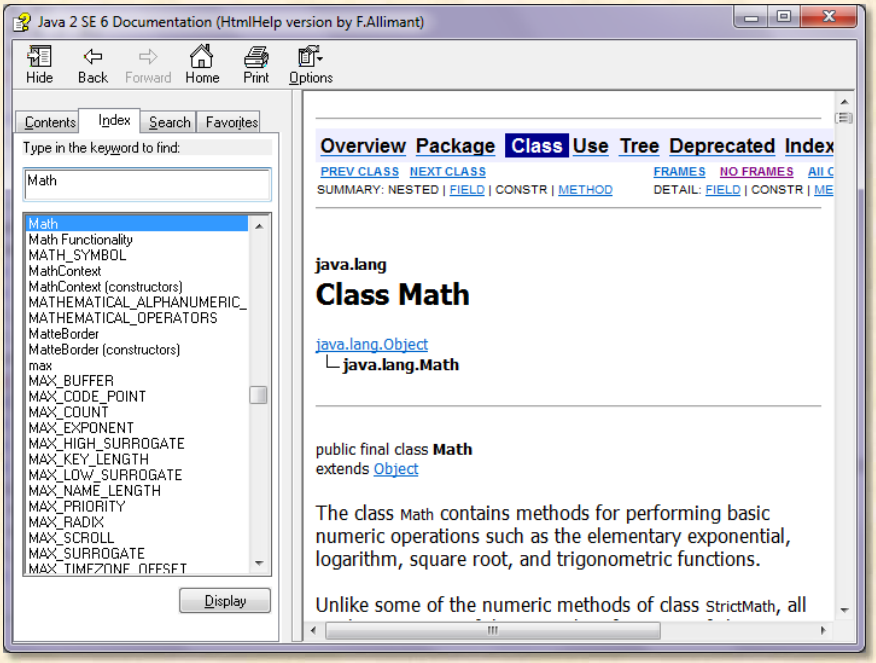
- ตัวแปร **`w`** เก็บน้ำหนัก (หน่วยเป็นกิโลกรัม)
- ตัวแปร **`h`** เก็บความสูง (หน่วยเป็นเซนติเมตร)

ผลลัพธ์ (แสดงทางจอภาพ)

- ตัวแปร **`mosteller`** เก็บพื้นที่ผิวกาย (หน่วยเป็นตารางเมตร) คำนวณตามสูตรของ Mosteller
- ตัวแปร **`dubois`** เก็บพื้นที่ผิวกาย (หน่วยเป็นตารางเมตร) คำนวณตามสูตรของ Du Bois
- ตัวแปร **`boyd`** เก็บพื้นที่ผิวกาย (หน่วยเป็นตารางเมตร) คำนวณตามสูตรของ Boyd

ข้อแนะนำ

นอกจากตัวดำเนินการ **`+`** **`-`** **`*`** และ **`/`** ที่ใช้ในการบวก ลบ คูณ และหารจำนวนแล้ว การคำนวณพื้นที่ผิวตามสูตรที่ให้มานี้ ต้องทราบวิธีการคำนวณรากที่สอง การยกกำลัง และการหาค่า log ระบบจาวามีคลังคำสั่งที่ให้บริการการคำนวณทางคณิตศาสตร์มากมาย ซึ่งสามารถอ่านรายละเอียดได้จากแฟ้มช่วยเหลือ (กดปุ่ม F1 ใน JLab) แล้วค้นคำว่า **`Math`** ดังรูป แสดงเอกสารอธิบายการใช้งานฟังก์ชันทางคณิตศาสตร์มากมาย



ในที่นี้ขออธิบายเฉพาะฟังก์ชันที่จำเป็นในการเขียนคำสั่งของปฏิบัติการครั้งนี้

- **`Math.sqrt( x )`** แทนการคำนวณค่าของ $\sqrt{x}$
- **`Math.pow( x, y )`** แทนการคำนวณค่าของ x^y
- **`Math.log10( x )`** แทนการคำนวณค่าของ $\log_{10} x$
  (ถ้าเขียน **`Math.log(x)`** จะเป็นการหาค่าของ $\log_e x$)

ตัวอย่างเช่น ต้องการหาค่าของ $\sqrt{a + \sqrt[4]{b + \log_{10}(c + a)}}$ เขียนแทนได้ดังนี้

```
Math.sqrt( a + Math.pow( b + Math.log10(c + a), 0.25 ) )
```

จะเลือกใช้ **`Math.pow( x, 0.5 )`** แทน **`Math.sqrt( x )`** ก็ไม่ผิด
แต่อย่าเขียน **`Math.pow( x, 1/2 )`** แทน **`Math.sqrt( x )`** เพราะมันผิด รู้ไหมว่าทำไม ?

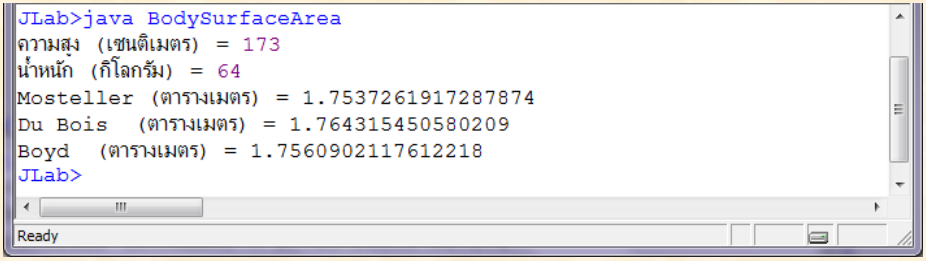
อ่านเพิ่มเติม

- "เริ่มเรียนเขียนโปรแกรม" บทที่ 2

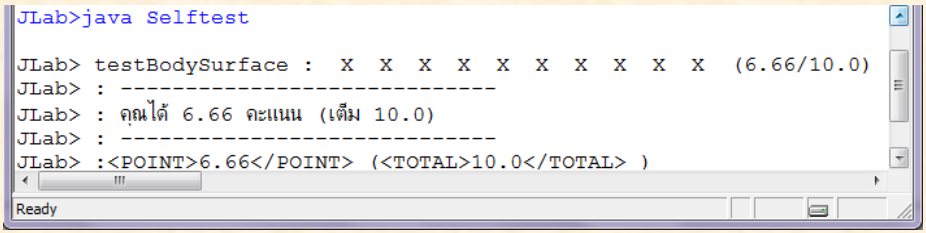
รหัสต้นฉบับเริ่มต้น

```
import java.util.Scanner;

public class BodySurfaceArea {
  // เขียนโปรแกรมคำนวณพื้นที่ผิวของร่างกาย จากความสูงและน้ำหนัก
  // ตามสูตรของ Mosteller, Du Bois, และ Boyd
  // หมายเหตุ : ใช้คำสั่งรับข้อมูลและแสดงผลลัพธ์ที่ได้เขียนไว้แล้วในโปรแกรม

  public static void main(String[] args) {
    Scanner kb = new Scanner(System.in);
    System.out.print("ความสูง (เซนติเมตร) = ");
    double h = kb.nextDouble();
    System.out.print("น้ำหนัก (กิโลกรัม) = ");
    double w = kb.nextDouble();



    double mosteller =



    double dubois =



    double boyd =



    System.out.println("Mosteller (ตารางเมตร) = " + mosteller);
    System.out.println("Du Bois   (ตารางเมตร) = " + dubois);
    System.out.println("Boyd   (ตารางเมตร) = " + boyd);
  }

}
```

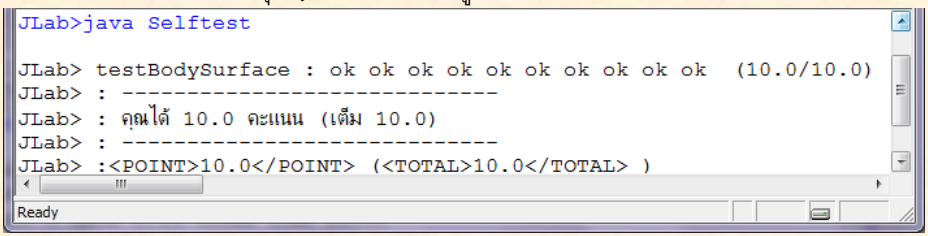
- กรณีต้องการทดสอบด้วยตนเอง : กดปุ่ม F5 เพื่อสั่งโปรแกรมทำงานเริ่มที่เมท็อด **main** โปรแกรมที่เขียนไว้ จะรอรับความสูงและน้ำหนัก คำนวณตามสูตร แล้วแสดงผลลัพธ์ทางจอภาพ เช่น ถ้าป้อนความสูง 173 น้ำหนัก 64 และโปรแกรมเขียนได้ถูกต้อง ควรจะได้ผลการทำงานดังแสดงข้างล่างนี้



- กรณีต้องการให้ระบบตรวจให้คะแนนอัตโนมัติ : กดปุ่ม F6 ระบบจะสั่งให้โปรแกรมตัวตรวจทำงาน โดยตัวตรวจจะส่งข้อมูลทดสอบจำนวนหนึ่ง ป้อนให้กับโปรแกรมที่เขียน จากนั้นอ่านผลลัพธ์กลับมาตรวจสอบว่า ถูกต้องหรือไม่ กระทำการทดสอบเช่นนี้ หลาย ๆ ครั้ง แล้วรายงานคะแนนที่ได้ รูปแรกข้างล่างนี้แสดงกรณีที่ยังเขียนโปรแกรมไม่สมบูรณ์ มีบางกรณีที่โปรแกรมทำงานได้ผลลัพธ์ไม่ถูกต้อง คะแนนจึงยังไม่เต็ม



ถ้าโปรแกรมผ่านการทดสอบทุก ๆ กรณี จะได้ผลดังรูปข้างล่างนี้



แบบฝึกหัดเพิ่มเติม

1. จงเขียนนิพจน์คณิตศาสตร์เพื่อคำนวณสูตรต่าง ๆ ต่อไปนี้

$(a + b^2c)^2$	`(a + b*b*c)*(a + b*b*c)` หรือ `Math.pow(a + b*b*c, 2)`
$(1+x)^{-2}$	
$(x_1 - x_2)(x_1 - x_3)$	
$\sqrt{(a-b)(b-c)(c-a)}$	
$\frac{c}{a^{-1} + b^{-1}}$	
$\frac{1}{\sqrt{5}}(a+b)$	
$\frac{(x_1 - x_2)^3}{x_1^{-2} + x_2^{-2}}$	
$1 + x + x^2 + x^3 + x^4 + x^5$	
$\sqrt{\left\|x - \frac{y}{z^2}\right\|}$	
$\frac{\sqrt{\left(1 + \frac{a}{b} - \frac{c-d}{h}\right)}}{xy}$	

2. จงเขียนสูตรคณิตศาสตร์จากนิพจน์ต่าง ๆ ต่อไปนี้

`Math.sqrt(a*a + b*b)`	$\sqrt{a^2 + b^2}$
`4*Math.PI*(r*r*r)`	
`1+ x + x*x/(2*1) + x*x*x/(3*2*1) + x*x*x*x/(4*3*2*1)`	
`(x/y + 1)(x*2 - Math.pow(x,3))`	
`Math.pow(Math.sin(x), 2) + Math.pow(Math.cos(y), 2)`	
`Math.sqrt(Math.pow(2, Math.log10(x)))`	
`(Math.abs(x)+Math.abs(y))/(x+y)*a`	
`(1 + Math.exp(t*Math.PI))*p/q - 3`	
`Math.pow(Math.E, 2*Math.PI)`	
`Math.sqrt(2*Math.PI*n)*Math.pow(n/Math.E, n)`	
`(-a + Math.sqrt(a/b-2))/-2*b`	

ปฏิบัติการที่ 2 : การประมาณค่า π

ผลการเรียนรู้

- ใช้คำสั่งทำงานเป็นวงวนเพื่อทำกลุ่มคำสั่งกลุ่มหนึ่งซ้ำ ๆ
- ใช้ตัวแปรเสริมเพื่อนับจำนวนรอบ และกำกับสถานะของการคำนวณ
- เข้าใจขีดจำกัดความแม่นยำในการคำนวณของเครื่องคอมพิวเตอร์

เนื้อหา

อัตราส่วนของเส้นรอบวงกับเส้นผ่านศูนย์สูตรของวงกลมใดๆ เป็นค่าคงตัว มีค่าประมาณ $\frac{355}{113}$ หรือที่จำกันได้ทั่วไปว่า ประมาณ 3.14159 เขียนแทนด้วยสัญลักษณ์ π ค่าคงตัวนี้เป็นจำนวนอตรรกยะ หาค่าได้จากสูตรข้างล่างนี้

$$\pi = 4\left(1-\frac{1}{3}+\frac{1}{5}-\frac{1}{7}+\frac{1}{9}-\frac{1}{11}+\cdots\right)$$

$$\pi = 4\left(\frac{2}{3}\cdot\frac{4}{3}\cdot\frac{4}{5}\cdot\frac{6}{5}\cdot\frac{6}{7}\cdot\frac{8}{7}\cdots\right)$$

$$\pi = \sqrt{12}\left(1-\frac{1}{3\cdot 3}+\frac{1}{5\cdot 3^2}-\frac{1}{7\cdot 3^3}+\cdots\right)$$

$$\pi = 2\left(1+\frac{1}{3}\left(1+\frac{2}{5}\left(1+\frac{3}{7}(1+\cdots)\right)\right)\right)$$

และอื่น ๆ อีกมากมาย (ผู้สนใจสามารถค้นคำว่า approximations of pi ในอินเทอร์เน็ต) สำหรับปฏิบัติการนี้ จะขอสนใจ 2 สูตรแรกที่แสดงข้างบนนี้ (ซึ่งเป็นสูตรประมาณค่า π ที่ไม่ค่อยดีนัก ☺)

สิ่งที่ต้องเขียน

รหัสต้นฉบับเริ่มต้น (**Pi.java**) ที่ให้มามีคำสั่งรับข้อมูลขาเข้าจากผู้ใช้ พร้อมกับคำสั่งแสดงผลลัพธ์ให้เรียบร้อยแล้ว ให้เขียนเฉพาะคำสั่งคำนวณค่าประมาณของ π จากสองสูตรแรกที่แสดงข้างบนนี้

ข้อมูลขาเข้า (รับจากแป้นพิมพ์)

- ตัวแปร **k** คือ จำนวนพจน์ที่ต้องการให้คำนวณในสูตรเพื่อประมาณค่าของ π เช่น

  k = 4 คือ ให้คำนวณ

  $$Pi1 = 4\left(1-\frac{1}{3}+\frac{1}{5}-\frac{1}{7}\right), \quad Pi2 = 4\left(\frac{2}{3}\cdot\frac{4}{3}\cdot\frac{4}{5}\cdot\frac{6}{5}\right)$$

  k = 6 คือ ให้คำนวณ

  $$Pi1 = 4\left(1-\frac{1}{3}+\frac{1}{5}-\frac{1}{7}+\frac{1}{9}-\frac{1}{11}\right), \quad Pi2 = 4\left(\frac{2}{3}\cdot\frac{4}{3}\cdot\frac{4}{5}\cdot\frac{6}{5}\cdot\frac{6}{7}\cdot\frac{8}{7}\right)$$

  เพื่อให้ง่ายต่อการคำนวณ กำหนดให้ค่า **k** ที่ใช้ในการทดสอบ เป็นจำนวนคู่ที่มากกว่า 0 เสมอ

ผลลัพธ์ (แสดงทางจอภาพ)

- ตัวแปร **Pi1** และ **Pi2** เก็บค่าประมาณของ π ที่คำนวณตามสูตร โดยใช้ **k** พจน์ในการคำนวณ

ข้อแนะนำ

จากสูตรแรกของการประมาณค่า π พบว่า หากเขียนการคำนวณนี้ในรูปของผลบวก จะได้ดังแสดงข้างล่างนี้

$$\pi = 4\left(1-\frac{1}{3}+\frac{1}{5}-\frac{1}{7}+\frac{1}{9}-\frac{1}{11}+\cdots\right) = 4\sum_{m=1}^{\infty}\frac{(-1)^{m-1}}{2m-1}$$

มีการคำนวณจำนวนพจน์เป็นอนันต์ แต่เราสนใจแค่ **k** พจน์ เขียนผลบวกใหม่ได้

$$\pi \approx 4\sum_{m=1}^{\mathbf{k}}\frac{(-1)^{m-1}}{2m-1}$$

เราสามารถเขียนวงวนที่ควบคุมการทำงานซ้ำๆ เป็นจำนวน **k** รอบ ได้หลายแบบ

กรณีใช้ **while**

```
int m = 1;
while (m <= k) {
    …
  m++;
}
```

กรณีใช้ **for**

```
for (int m = 1; m <= k; m++) {
    …
}
```

ภายในวงวน ก็เพียงแค่คำนวณค่าของ $(-1)^{m-1}/(2m-1)$ แล้วรวมค่าที่คำนวณได้นี้เข้าไปในตัวแปรที่เก็บผลรวม เมื่อวนคำนวณจนครบ ก็คูณด้วย 4 ได้ค่าประมาณของ π ตามสูตรที่หนึ่ง

สำหรับสูตรที่สอง จะยุ่งกว่าสูตรที่หนึ่งเล็กน้อย ขอเตือนไว้ก่อนว่า อย่าใช้วิธีคำนวณผลคูณของเศษให้ครบ k พจน์ก่อน ตามด้วยการคำนวณผลคูณของส่วนอีก k พจน์ แล้วนำผลคูณของเศษหารด้วยผลคูณของส่วน วิธีนี้อาจได้ค่าผิด เมื่อ k มีค่ามาก ลองคิดดู สมมติ $k = 400$ ผลคูณของเศษ และผลคูณของส่วนจะมีค่ามากแค่ไหน ค่าของผลคูณอาจมากกว่าค่าที่ตัวแปรจะเก็บไว้ได้ หรือถ้าเก็บได้ก็อาจมีความแม่นยำลดลง (อย่าลืมว่าคอมพิวเตอร์ประมวลผลจำนวนด้วยตัวแปรที่มีขนาดของหน่วยความจำคงที่ เช่น **int** ใช้ 4 ไบต์ต่อตัว เก็บค่าได้ประมาณ บวกลบสองพันล้าน ส่วน **double** ใช้ 8 ไบต์ต่อตัว ถึงแม้จะเก็บค่าที่มีขนาดใหญ่มากถึงบวกลบ 10^{308} แต่เก็บได้ละเอียดแค่ 15-16 หลักเท่านั้น) จึงขอแนะนำให้หาผลคูณของเศษหารด้วยส่วนของแต่ละพจน์ เช่น $k = 4$

ให้คำนวณ $\pi \approx 4\left(\left(\frac{2}{3}\right)\cdot\left(\frac{4}{3}\right)\cdot\left(\frac{4}{5}\right)\cdot\left(\frac{6}{5}\right)\right)$ อย่าคำนวณ $\pi \approx 4\left(\frac{2\cdot4\cdot4\cdot6}{3\cdot3\cdot5\cdot5}\right)$

นอกจากนี้ อย่าใช้ตัวแปร **int** เพราะ 2/3 ให่ค่าเป็น 0 ไม่ใช่ 0.6666667 ใช่ไหม?

ข้อแนะนำ (เพิ่มเติม)

ถ้าคุณเป็นคนหนึ่งที่คิดจะคำนวณค่าของ $(-1)^{m-1}$ ด้วยการใช้ **Math.pow(-1,m-1)** ซึ่งทำได้ไม่ผิด แต่อยากให้ลองคิดใหม่ ทำอีกแบบ ทำไมไม่สร้างตัวแปรขึ้นมาสักตัว ให้มีค่าเริ่มต้นเป็น 1 แล้วเปลี่ยนค่าของตัวแปรนี้โดยการคูณด้วย –1 ในแต่ละรอบ ทำให้ค่าของมันเปลี่ยนสลับระหว่าง บวกหนึ่งกับลบหนึ่งไปเรื่อย ๆ จึงสามารถนำค่าของตัวแปรนี้ไปใช้แทน $(-1)^{m-1}$ ได้ แถมยังทำงานเร็วกว่าด้วยเพราะไม่ต้องเสียเวลายกกำลัง

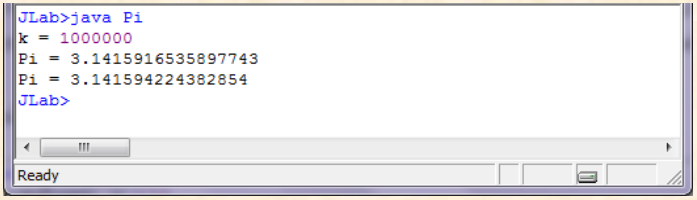
อ่านเพิ่มเติม

- “เริ่มเรียนเขียนโปรแกรม” บทที่ 3

รหัสต้นฉบับเริ่มต้น

```
import java.util.Scanner;

public class Pi {
  // pi1 = 4*(1 - 1/3 + 1/5 - 1/7 + 1/9 - 1/11 + ...)
  // pi2 = 4*(2/3 * 4/3 * 4/5 * 6/5 * 6/7 * 8/7 * ...)
  public static void main(String[] args) {
    Scanner kb = new Scanner(System.in);
    System.out.print("k = ");
    int k = kb.nextInt();

    double pi1 = 0, pi2 = 0;









    System.out.println("Pi = " + pi1);  // แสดงผลของสูตรแรก
    System.out.println("Pi = " + pi2);  // แสดงผลของสูตรที่สอง
  }

}
```

- กรณีต้องการทดสอบด้วยตนเอง : กดปุ่ม F5 เพื่อสั่งโปรแกรมทำงานเริ่มที่เมท็อด **main** โปรแกรมที่เขียนไว้ จะรอรับค่าของ **k** ที่ระบุจำนวนพจน์ในการประมาณค่า π คำนวณเสร็จก็แสดงผลลัพธ์ทางจอภาพ เช่น ถ้าป้อนให้ **k** = 1000000 ควรจะได้ผลการทำงานดังแสดงข้างล่างนี้



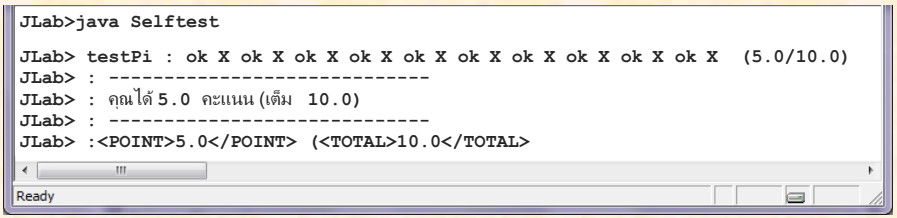
หมายเหตุ : ค่าที่ได้อาจไม่ตรงกับหลักท้าย ๆ ที่แสดงข้างบนนี้ก็ได้ ทั้งนี้ขึ้นกับการวางลำดับการคำนวณ เช่น หากลองสั่งคำสั่งข้างล่างนี้ทำงาน จะได้ผลไม่เท่ากัน

```
double a = 1001, b = 1002, c = 1003;
System.out.println((a * c) / (b * b));
System.out.println((a / b) * (c / b));
```

$\frac{ac}{b^2}$

$\frac{a}{b}\frac{c}{b}$

- กรณีต้องการให้ระบบตรวจให้คะแนนอัตโนมัติ : กดปุ่ม F6 ระบบจะสั่งให้โปรแกรมตัวตรวจทำงาน โดยตัวตรวจจะส่งข้อมูลทดสอบ ป้อนให้กับโปรแกรมที่เขียน จากนั้นอ่านผลลัพธ์กลับมาตรวจสอบว่า ถูกต้องหรือไม่ กระทำการทดสอบเช่นนั้น หลาย ๆ ครั้ง แล้วรายงานคะแนนที่ได้ รูปข้างล่างนี้แสดงกรณีที่ยังเขียนโปรแกรมไม่สมบูรณ์ มีบางกรณีที่โปรแกรมทำงานได้ผลลัพธ์ไม่ถูกต้อง ได้คะแนนยังไม่เต็ม



แบบฝึกหัดเพิ่มเติม

1. อยากทราบว่า วงวนแต่ละวงข้างล่างนี้ พิมพ์ * กี่ตัว

```
int k = 0;
while (true) {
  if (k >= n) break;
  System.out.print("*");
  k++;
}
```

```
int k = 0;
while (true) {
  System.out.print("*");
  k++;
  if (k > n) break;
}
```

```
int k = n;
while (true) {
  System.out.print("*");
  k--;
  if (k != 0) break;
}
```

```
int k = n;
while (true) {
  System.out.print("*");
  k--;
  if (k == 0) break;
}
```

2. จงตอบแต่ละคำถามข้างล่างนี้ (ในช่องด้านซ้าย) และเขียนวงวน `while` ใหม่แบบไม่ใช้ `while(true)` ในช่องด้านขวา

เมื่อออกจากวงวนข้างล่างนี้แล้ว `k` มีค่าเท่าไร
`int k = 10;`
`while(true) {`
`  if ( k > 100 ) break;`
`  k += 3;`
`}`	

เมื่อออกจากวงวนข้างล่างนี้แล้ว `k` มีค่าเท่าไร
`int k = 8;`
`while(true) {`
`  if ( k > 17 ) break;`
`  if ( k % 7 == 0 ) break;`
`  ++k;`
`}`	

เติมในช่องว่าง ให้ทำคำสั่ง `k += 2` เป็นจำนวน 10 ครั้ง
`int k = 0;`
`while(true) {`
`  if ( k > _____ ) break;`
`  k += 2;`
`}`	

หลังจากทำคำสั่งข้างล่างนี้แล้วจะแสดงค่าใด
`int x = 100;`
`int c = 0;`
`while(true) {`
`  if (x <= 0) break;`
`  c++;`
`  x /= 2;`
`}`
`System.out.println(c);`	

คำสั่งสุดท้ายของชุดคำสั่งข้างล่างนี้จะแสดงอะไร
`int k = 0, j = 1;`
`while(true) {`
`  if ( k > 4 ) break;`
`  j = 0;`
`  while(true) {`
`  if ( j > 3 ) break;`
`  ++j;`
`  }`
`  ++k;`
`}`
`System.out.println( k + j );`	

3. จงเขียนโปรแกรมที่คำนวณสูตร $\sum_{k=1}^{n}\frac{1}{k}$ โดยรับค่า n จากผู้ใช้ทางแป้นพิมพ์ เช่น $n=2$ จะได้ $\sum_{k=1}^{2}\frac{1}{k}=\frac{1}{1}+\frac{1}{2}=1.5$

หรือ $n=5$ จะได้ $\sum_{k=1}^{5}\frac{1}{k}=\frac{1}{1}+\frac{1}{2}+\frac{1}{3}+\frac{1}{4}+\frac{1}{5}=2.83333$ เป็นต้น

4. จงเขียนโปรแกรมที่คำนวณค่าของ $1+\cfrac{1}{1+\cfrac{1}{1+\cfrac{1}{\ddots}}}$ คำนวณลึกลงไป k ชั้น ค่า k นี้รับจากผู้ใช้ทางแป้นพิมพ์

เช่น $k=1$ ได้ $1+1=2$, $k=2$ ได้ $1+\cfrac{1}{1+1}=1.5$, $k=3$ ได้ $1+\cfrac{1}{1+\cfrac{1}{1+1}}\approx 1.67$

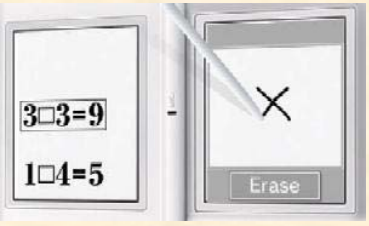
ปฏิบัติการที่ 3 : เกมฝึกสมอง

ผลการเรียนรู้

- ใช้คำสั่ง **if** เพื่อทดสอบเงื่อนไขว่าจะให้ทำหรือไม่ทำคำสั่งที่เตรียมไว้

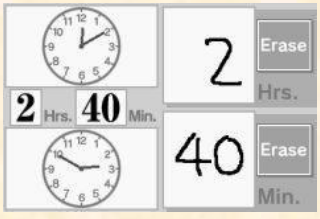
เนื้อหา

ปฏิบัติการนี้ประกอบด้วยโจทย์ย่อยสองข้อ มาจากเกม Brain Age™ ของเครื่องเล่นเกมพกพา Nintendo DS ดังนี้



Sign Finder

- เกมนี้ทดสอบความไวในการเติมเครื่องหมาย + − × หรือ / ให้กับ สมการง่าย ๆ ที่แสดงบนจอ
- ตัวอย่างในรูปทางขวา ถามว่า 3 ทำอะไรกับ 3 ถึงจะเท่ากับ 9
- คำตอบก็คือ ×



Time Lapse

- เกมนี้ทดสอบการคำนวณช่วงเวลาจากเริ่มถึงจบว่า ห่างกันเท่าไร
- ตัวอย่างในรูปทางขวา ถามว่าเริ่มเวลา 12:10 น. (นาฬิการูปบน) จบที่ เวลา 2:50 น. (นาฬิการูปล่าง) ห่างกันเท่าไร
- คำตอบก็คือ 2 ชั่วโมง 40 นาที
- กำหนดให้เวลาเริ่มกับจบห่างกันน้อยกว่า 12 ชั่วโมง

สิ่งที่ต้องเขียน

เขียนโปรแกรม **SignFinder.java** ให้ทำงานดังนี้

- รับตัวเลขจากแป้นพิมพ์ 3 จำนวน
- แสดงเครื่องหมาย **+ - x** หรือ **/** ที่เมื่อนำสองจำนวนแรกมาดำเนินการกันแล้ว ได้จำนวนที่สาม
- ดังตัวอย่างข้างล่างนี้ (ให้สังเกตว่า การคูณใช้ ตัวอักษรเอ็กซ์เล็ก ไม่ใช่เครื่องหมายดอกจัน)

```
JLab>java SignFinder
จำนวนทั้งสาม = 3,3,9
เครื่องหมายที่ต้องการคือ x
Ready
```

เขียนโปรแกรม **TimeLapse.java** ให้ทำงานดังนี้

- รับเวลาเริ่มต้นจากแป้นพิมพ์ ประกอบด้วยตัวเลข 2 จำนวน (ชั่วโมง และนาที เลขชั่วโมงอยู่ในช่วง 1 - 12)
- รับเวลาสิ้นสุดจากแป้นพิมพ์ ประกอบด้วยตัวเลข 2 จำนวน (ชั่วโมง และนาที เลขชั่วโมงอยู่ในช่วง 1 - 12)
- แสดงผลต่างของเวลาทั้งสองในรูปแบบ ดังตัวอย่างข้างล่างนี้

```
JLab>java TimeLapse
เวลาเริ่มต้น = 12,10
เวลาสิ้นสุด = 2,50
2:40
Ready
```

ข้อแนะนำ

Sign Finder : คงต้องใช้คำสั่ง `if` หลาย ๆ คำสั่งทดสอบว่า ควรใช้บวก ลบ คูณ หรือหาร ต่อเนื่องกัน ขอเน้นว่าไม่ต้องห่วงกรณีที่ใช้เครื่องหมายทั้งสี่ไม่ได้ เพราะผู้สร้างข้อมูลทดสอบให้กับเกมนี้ประกันว่า ต้องมีสักเครื่องหมายแน่นอนที่ใช้ได้ บางคนอาจเลือกใช้คำสั่ง `if-else` ซ้อน ๆ กันในการทดสอบก็ย่อมทำได้ นอกจากนี้ อาจมีคำถามว่า ถ้าสามารถใส่เครื่องหมายได้หลายแบบจะให้แสดงอะไร เช่น 0 ทำอะไรกับ 0 เพื่อให้ได้ 0 (กรณีนี้ใส่ได้ทั้ง + − และ ×) คำตอบก็คือ อะไรก็ได้ อ้อ อย่าลืมว่า การหารนั้น เป็นการหารแบบจำนวนเต็ม คือ หารแล้วปัดเศษทิ้ง

Time Lapse : หากงงว่าจะเขียนคำสั่งให้กับข้อนี้อย่างไร ขอแนะนำให้ลองพิจารณาจากกรณีตัวอย่างต่อไปนี้

- เริ่ม 11:40 จบ 12:05 เป็นเวลา 0 ชั่วโมง 25 นาที
- เริ่ม 11:05 จบ 11:40 เป็นเวลา 0 ชั่วโมง 35 นาที
- เริ่ม 11:05 จบ 12:40 เป็นเวลา 1 ชั่วโมง 35 นาที
- เริ่ม 11:05 จบ 1:40 เป็นเวลา 2 ชั่วโมง 35 นาที (โจทย์กำหนดให้ชั่วโมงเป็นเลข 1 ถึง 12)
- เริ่ม 11:05 จบ 10:00 เป็นเวลา 10 ชั่วโมง 55 นาที (โจทย์ให้เริ่มกับจบห่างกันน้อยกว่า 12 ชั่วโมง)

ลองคิดว่า เราต้องคำนวณอย่างไร จึงได้คำตอบตามตัวอย่างข้างบนนี้ กรณีที่ง่ายคือ กรณีที่เลขจบมากกว่าเลขเริ่ม แต่จะยุ่งขึ้นเมื่อเลขจบน้อยกว่าเลขเริ่ม (สองกรณีสุดท้าย) จะคิดอย่างไร อย่าลืมว่า 1 ชั่วโมงมี 60 นาที ดังนั้นเราสามารถลดเลขของชั่วโมงลง 1 แล้วไปเพิ่มอีก 60 ให้กับเลขนาที นอกจากนี้ หากชั่วโมงเริ่ม 11 จบ 1 เรารู้ว่าห่างกัน 2 แต่ถ้านำ จบ – เริ่ม จะได้ 1 – 11 = –10 ซึ่งไม่ถูก จะทำอย่างไรให้กลายเป็น 2 เราต้องลองหลาย ๆ กรณี เมื่อมั่นใจ ก็พยายามเปลี่ยนเงื่อนไข มาเป็นการทดสอบด้วย `if` หรือ `if-else` ก็จะสามารถเขียนเป็นโปรแกรมที่สมบูรณ์ได้ การเขียนโปรแกรมต้อง เขียน → ทดสอบ → แก้ไข → ทดสอบ → แก้ไข → … → ทดสอบ → ถูกต้อง ถือว่าเป็นเรื่องปกติ ต้องหัดสังเกต สร้างข้อมูลทดสอบ และหาที่ผิด ก็จะสำเร็จ ขอให้มีเวลา มีความพยายาม

อ่านเพิ่มเติม

- “เริ่มเรียนเขียนโปรแกรม” บทที่ 3

รหัสต้นฉบับเริ่มต้น

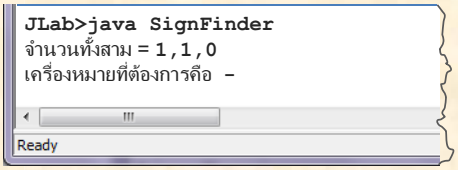
```
import java.util.Scanner;

public class SignFinder {

  public static void main(String[] args) {
    // การสร้าง Scanner ข้างล่างนี้ทำให้ผู้ใช้สามารถป้อนจำนวนแต่ละตัว คั่นด้วย , ได้
    Scanner kb = new Scanner(System.in).useDelimiter("\\s*[,\\s]\\s*");
    System.out.print("จำนวนทั้งสาม = ");
    int a = kb.nextInt();
    int b = kb.nextInt();
    int c = kb.nextInt();

    String sign = "";   // ใช้เครื่องหมาย + - x และ /   << คูณใช้ x ไม่ใช่ *









    System.out.println("เครื่องหมายที่ต้องการคือ " + sign);

  }

}
```

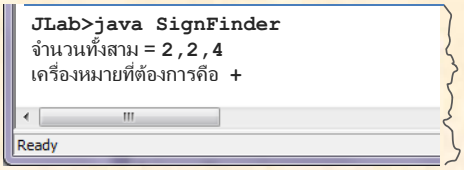
รหัสต้นฉบับเริ่มต้น

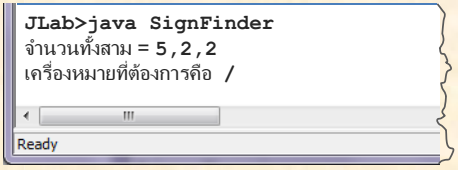
```
import java.util.Scanner;

public class TimeLapse {

  public static void main(String[] args) {
    // การสร้าง Scanner ข้างล่างนี้ทำให้ผู้ใช้สามารถป้อนจำนวนแต่ละตัว คั่นด้วย , ได้
    Scanner kb = new Scanner(System.in).useDelimiter("\\s*[,\\s]\\s*");
    System.out.print("เวลาเริ่มต้น = ");
    int h1 = kb.nextInt();
    int m1 = kb.nextInt();
    System.out.print("เวลาสิ้นสุด = ");
    int h2 = kb.nextInt();
    int m2 = kb.nextInt();

    int dh = 0;   // ใช้ตัวแปรนี้เก็บผลต่างชั่วโมง
    int dm = 0;   // ใช้ตัวแปรนี้เก็บผลต่างนาที
```

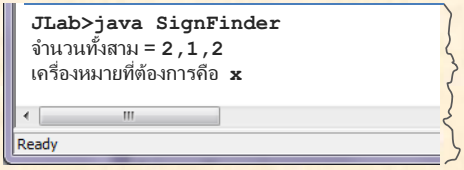
```
    System.out.println(dh + ":" + dm);

  }

}
```

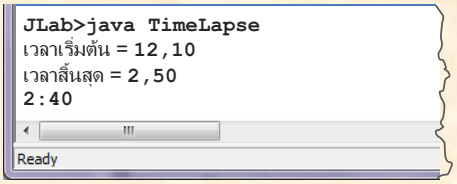
- กรณีต้องการทดสอบด้วยตนเอง : ให้เลือกคลาสที่ต้องการสั่งทำงาน (ในปฏิบัติการนี้คือ **SignFinder** หรือ **TimeLapse**) แล้วกดปุ่ม F5 เพื่อสั่งโปรแกรมทำงานเริ่มที่เมท็อด **main** ของคลาสนั้น ดูรูปต่าง ๆ ข้างล่างนี้ ใช้เป็นกรณีทดสอบได้ด้วยตนเอง และควรลองข้อมูลทดสอบกรณีอื่น ๆ ดูด้วย

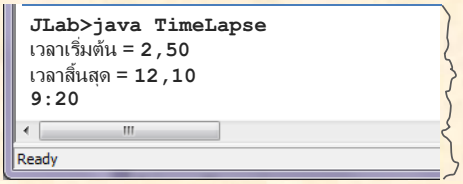




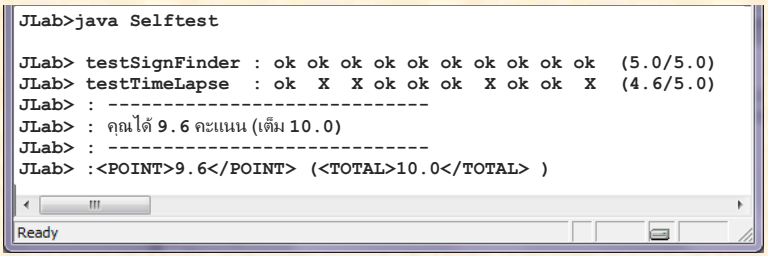








- กรณีต้องการให้ระบบตรวจให้คะแนนอัตโนมัติ : กดปุ่ม F6 ระบบจะสั่งให้โปรแกรมตัวตรวจทำงาน โดยตัวตรวจจะส่งข้อมูลทดสอบจำนวนหนึ่ง ป้อนให้กับโปรแกรมที่เขียน จากนั้นอ่านผลลัพธ์กลับมาตรวจสอบว่า ถูกต้องหรือไม่ ระบบกระทำการทดสอบเช่นนั้น หลาย ๆ ครั้ง แล้วรายงานคะแนนที่ได้ รูปข้างล่างนี้แสดงกรณีที่ยังเขียนโปรแกรมไม่สมบูรณ์ ผิดนิดเดียวในโปรแกรม **TimeLapse** จึงได้คะแนนเกือบเต็ม คงต้องกลับไปดูว่ามีอะไรผิด



แบบฝึกหัดเพิ่มเติม

1. จงเขียนโปรแกรมรับความยาวด้านแต่ละด้านของสามเหลี่ยม จากนั้นตรวจสอบพร้อมกับแสดงผลการตรวจสอบว่าเป็นความยาวด้านที่ประกอบกันเป็นสามเหลี่ยมได้หรือไม่ (ข้อแนะนำ : ผลรวมของความยาวด้านสองด้านของสามเหลี่ยมใด ๆ ต้องยาวกว่าด้านที่สาม)

2. เรามักพบโฆษณาส่งเสริมการขายประเภท “ซื้อ 5 แถม 1” ซึ่งโดยทั่วไปมักหมายความว่าซื้อสินค้า 6 ชิ้น แล้วทางร้านจะไม่คิดราคาชิ้นที่มีราคาน้อยสุด จงเขียนโปรแกรมเพื่อรับราคาสินค้า 6 จำนวน แสดงผลรวมราคาของสินค้าทั้งหก แสดงราคาสินค้าที่น้อยสุด และแสดงเงินที่ต้องชำระ

3. จงเขียนโปรแกรมหาค่ามากสุดของจำนวน 10 จำนวนที่รับเข้ามาทางแป้นพิมพ์

4. จงเขียนโปรแกรมแสดงจำนวนเต็ม a, b และ c ทุกจำนวนที่น้อยกว่า 500 ที่ค่าของ $a^2 + b^2$ เท่ากับ c^2 (เช่น $3^2 + 4^2 = 5^2$) โดยไม่แสดงค่าซ้ำ เช่น เคยแสดง 3, 4, 5 แล้ว จะไม่แสดง 4, 3, 5 (ข้อแนะนำ : คงต้องใช้วงวนซ้อนกันถึงสามชั้นเพื่อแปรค่าในตัวแปร a, b และ c)

ปฏิบัติการที่ 4 : พหูพจน์

ผลการเรียนรู้

- ใช้บริการต่าง ๆ ของสตริง
- ใช้คำสั่งทดสอบเพื่อแยกการทำงานเป็นกรณีต่าง ๆ

เนื้อหา

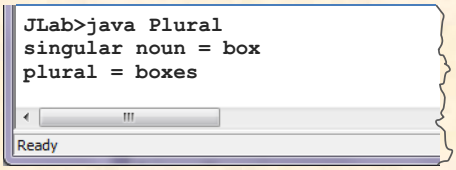
รูปของคำนามในภาษาอังกฤษมีทั้งแบบเอกพจน์และพหูพจน์ การเขียนคำนามในรูปพหูพจน์จากรูปเอกพจน์มีกฎการเขียนแบบง่ายๆ (ไม่ครอบคลุมทุกกรณี) ดังนี้

- ถ้าเป็นคำนามที่ลงท้าย s, x หรือ ch ทำเป็นพหูพจน์ได้ด้วยการเติม es ต่อท้าย
  (เช่น box → boxes, witch → witches เป็นต้น)
- ถ้าลงท้ายด้วย y แต่ตัวอักษรก่อน y ไม่ใช่สระ, ให้เปลี่ยน y เป็น i แล้วเติม es ต่อท้าย
  (เช่น fly → flies, memory → memories เป็นต้น)
- ถ้าไม่ตรงกับกฎสองข้อข้างบนนี้, ให้ต่อท้ายด้วย s เลย
  (เช่น computer → computers, boy → boys เป็นต้น)

สิ่งที่ต้องเขียน

เขียนโปรแกรม **`Plural.java`** ให้ทำงานดังนี้

- รับสตริงจากแป้นพิมพ์ สตริงที่รับมานี้คือ คำนามภาษาอังกฤษในรูปของเอกพจน์
- สร้างสตริงใหม่ที่เป็นพหูพจน์ของคำนามที่ได้รับ
- แสดงรูปพหูพจน์ที่หาได้ทางจอภาพ
- ดังตัวอย่างข้างล่างนี้



ข้อแนะนำ

เราได้ใช้สตริงมาตั้งแต่ชั่วโมงแรกๆ เพื่อแสดงข้อความทางจอภาพ และได้รู้จักการนำสตริงมาต่อกันด้วย + สตริงไม่ใช่ข้อมูลพื้นฐานในจาวา เป็นข้อมูลแบบอ็อบเจกต์ (จะได้เรียนเรื่องอ็อบเจกต์ในปลายภาค) การจัดเก็บและจัดการสตริงอยู่ในคลาสมาตรฐานชื่อ **`String.java`** เมื่อต้องการเรียกใช้บริการของสตริง จึงต้องเรียกเมท็อดที่เขียนไว้ในคลาส **`String`** (ลองกด F1 แล้วค้นคำว่า **`String`** ดู)

ตารางข้างล่างนี้แสดงบางเมท็อดของสตริง การเรียกใช้เมท็อดของสตริงอยู่ในรูปแบบ **`string.method(…)`** มีตัวสตริงอยู่หน้าจุดและชื่อเมท็อดอยู่หลังจุด

การเรียกใช้	ความหมาย
`s.length()`	คืนความยาวของ **`s`** (ซึ่งคือจำนวนอักขระใน **`s`**)
`s.trim()`	คืนสตริงใหม่ที่ตัดอักขระว่างทางซ้ายและขวาของ **`s`**
`s.toUpperCase()`	คืนสตริงใหม่ที่เหมือน **`s`** แต่ตัวอักษรอังกฤษทุกตัวเป็นตัวใหญ่หมด
`s.toLowerCase()`	คืนสตริงใหม่ที่เหมือน **`s`** แต่ตัวอักษรอังกฤษทุกตัวเป็นตัวเล็กหมด
`s.substring(i, j)`	คืนสตริงใหม่ที่ได้จากตัวที่ **`i`** ถึง **`j-1`** ของ **`s`**
`s.equals(t)`	คืนค่า **`true`** ถ้า **`s`** เหมือนกับ **`t`** ทุกอักขระ ถ้าไม่เหมือนคืน **`false`**
`s.indexOf(t, i)`	คืนตำแหน่งใน **`s`** ที่พบ **`t`** เริ่มจากดัชนี **`i`** ใน **`s`** ถ้าไม่พบคืน **`-1`**

การเรียกใช้บางเมท็อดของสตริงนั้นอาจรู้สึกว่า ต้องการให้เกิดการเปลี่ยนแปลงกับสตริง เช่น ใช้ **`s.toUpperCase()`** เพราะอยากเปลี่ยนเป็นตัวใหญ่หมด แต่นี่ไม่ได้เปลี่ยนสตริง **`s`** เลย แต่จะได้สตริงใหม่ที่เหมือน **`s`** เป็นตัวใหญ่หมด การเรียกเมท็อดเหล่านี้จึงต้องมีตัวแปรมารับผลลัพธ์ด้วย เช่น **`t = s.toUpperCase()`** จะได้ผลเก็บใน **`t`** ส่วน **`s`** นั้นไม่เปลี่ยนแปลง ถ้าต้องการให้ **`s`** เปลี่ยนเป็นของใหม่ ก็ต้องเขียน **`s = s.toUpperCase()`** การเขียน **`s.toUpperCase()`** โดดๆ จึงไม่มีผลอะไรเลย ขอเน้นอีกครั้งว่า ไม่มีเมท็อดใดเลยของสตริง ที่เรียกแล้ว จะเปลี่ยนแปลงสตริง (ที่เขียนอยู่ทางซ้ายหน้าเครื่องหมายจุด)

สำหรับโจทย์การแสดงพหูพจน์ คงต้องรู้วิธีหยิบตัวหรือสองตัวทางขวาของคำเอกพจน์ที่ได้รับ เพื่อนำไปทดสอบว่าเป็นแบบใด การหยิบตัวใดในสตริง กระทำได้ด้วยเมท็อด **`substring`** เช่น **`s.substring(2, 4)`** จะได้สตริงย่อยตั้งแต่ตัวที่ 2 ถึง (4 – 1) อย่าลืมว่าตัวอักษรซ้ายสุดของสตริงในจาวา ถือว่ามีตำแหน่งเป็น 0 ดังนั้น ถ้า **`s = "abcdef"`** ผลของ **`s.substring(2, 4)`** คือ **`"cd"`** แล้วเราจะหยิบตัวขวาสุดมาอย่างไร ? (ก็คงต้องหาความยาวของสตริงด้วย **`length()`** ก่อน แล้วก็ …)

มีอีกสามประเด็นที่อยากให้คำนึงถึง ประเด็นแรกคือ กรณีที่ผู้ใช้เผลอป้อนช่องว่างทางซ้ายหรือขวาของคำด้วย เราก็ควรจะลบช่องว่างเหล่านี้ทิ้งด้วย **`trim`** ประเด็นต่อมาคือ ผู้ใช้อาจป้อนตัวอังกฤษใหญ่หรือเล็กก็ได้ ถ้าจะต้องทดสอบทุกกรณี ก็เห็นว่าจะยุ่งเกินไป จึงควรเปลี่ยนคำที่ได้รับเป็นตัวเล็กให้หมดก่อน (หรือจะเป็นตัวใหญ่ก็ได้) จะทำให้การตรวจสอบที่จะตามมากระทำได้ซับซ้อนน้อยลง และประเด็นสุดท้ายคือ การทดสอบว่า สตริงสองตัวเท่ากันหรือไม่นั้น เราไม่ใช้เครื่องหมาย **`==`** เหมือนกรณีของจำนวน แต่ใช้เมท็อด **`equals`** แทน

อ่านเพิ่มเติม

- “เริ่มเรียนเขียนโปรแกรม” บทที่ 4

รหัสต้นฉบับเริ่มต้น

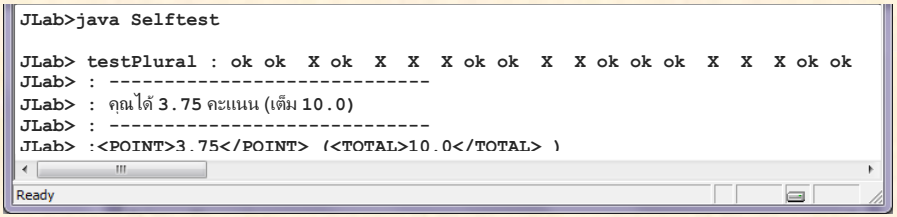
```
import java.util.Scanner;

public class Plural {

  public static void main(String[] args) {
    Scanner kb = new Scanner(System.in);
    System.out.print("singular noun = ");
    String s = kb.nextLine();
    String p = "";     // ตัวแปร p ไว้เก็บรูปพหูพจน์ของคำนามที่เก็บใน s

    System.out.println("plural = " + p); // นำ p มาแสดงเป็นผลลัพธ์
  }

}
```

การตรวจ

- กรณีต้องการทดสอบด้วยตนเอง : กดปุ่ม F5 เพื่อสั่งโปรแกรมทำงานเริ่มที่เมท็อด **main** โปรแกรมที่เขียนไว้ จะรอรับคำ แล้วแสดงพหูพจน์ทางจอภาพ
  เราควรทดสอบคำในทุกรูปแบบ โดยสร้างคำ ที่ไม่จำเป็นต้องมีความหมายก็ได้ เพราะเป็นแค่การทดสอบเท่านั้น เช่น fox, fdes, fch, fdh, fay, fey, fiy, foy, fuy, fy ก็น่าครอบคลุมพอสมควร (ลองเติมเองว่าแต่ละกรณีจะได้รูปพหูพจน์เป็นอะไร)

  fox _____ fdes _____ fch _____
  fdh _____ fay _____ fey _____
  fiy _____ foy _____ fuy _____
  fy _____

- กรณีต้องการให้ระบบตรวจให้คะแนนอัตโนมัติ : กดปุ่ม F6 ระบบจะสั่งให้โปรแกรมตัวตรวจทำงาน โดยตัวตรวจจะส่งข้อมูลทดสอบจำนวนหนึ่ง ป้อนให้กับโปรแกรมที่เขียน จากนั้นอ่านผลลัพธ์กลับมาตรวจสอบว่า ถูกต้องหรือไม่ สำหรับโจทย์นี้ ตัวตรวจจะผลิตคำต่าง ๆ เพื่อทดสอบทุก ๆ รูปแบบเพื่อให้ครอบคลุมทั้งสามกฎที่กำหนดไว้



แบบฝึกหัดเพิ่มเติม

1. จงเขียนชุดคำสั่งที่ทำหน้าที่ดังนี้ต่อไปนี้ (เขียนหลาย ๆ คำสั่ง หลาย ๆ บรรทัด ก็ได้ ไม่จำเป็นต้องเป็นทั้งคลาส ให้ถือว่า มีตัวแปรสตริงชื่อ `s` แล้ว และให้เก็บผลลัพธ์ในตัวแปร `t`)

   ก) ดึงสตริงย่อยที่อยู่ระหว่าง `[` กับ `]` ในสตริง `s` เช่น ถ้า `s = "--[ok]--"` จะได้ `"ok"`

   ข) ดึงสตริงย่อยที่อยู่ระหว่าง `<b>` กับ `</b>` ในสตริง `s` เช่น ถ้า `s = "..<b>ok</b>.."` จะได้ `"ok"`

   ค) สร้างสตริงใหม่ที่เหมือน `s` แต่เปลี่ยนทุกตัวใน `s` ที่เป็น `"a"` ให้เป็น `"x"` เช่น ถ้า `s = "mamaa"` จะได้ `"mxmxx"` (ต้องบอกว่าสตริงมีเมท็อด `replace` ที่ทำตามที่ต้องการ ขออย่าใช้ `replace` ในข้อนี้นะ)

   ง) สร้างสตริงใหม่ (เรียกว่าสตริง “ติดอ่าง”) ที่เหมือน `s` แต่ทำซ้ำทุกตัวอักษรใน `s`
   เช่น ถ้า `s = "banana"` จะได้ `"bbaannaannaa"`

2. จงเขียนโปรแกรมรับข้อความทางแป้นพิมพ์หนึ่งบรรทัด แล้วแสดงทางจอภาพว่า ข้อความที่ได้รับมีกี่ “คำ” นิยามให้หนึ่งคำ คือ ลำดับอักษรที่ติดกันไม่มีช่องว่าง เช่น " `What a wonderful world`" มี 4 คำ

3. จงเขียนโปรแกรมรับวันที่แบบสตริงทางแป้นพิมพ์หนึ่งบรรทัด มีรูปแบบของข้อมูลขาเข้าดังนี้

   - เริ่มด้วยชื่อเดือน ตามด้วยช่องว่าง ตามด้วยเลขวันที่ของเดือน ตามด้วย comma ตามด้วยช่องว่าง และปิดท้ายด้วยเลขปี เช่น December 16, 2003 หรือ March 2, 1999 เป็นต้น

   จากนั้นเปลี่ยนวันที่ที่ได้รับ ให้เป็นวันที่ในรูปแบบ d/m/y แล้วแสดงทางจอภาพ เช่น รับ March 2, 1999 จะแสดง 2/3/1999

ปฏิบัติการที่ 5 : เลขเด็ด

ผลการเรียนรู้

- การอ่านแฟ้มข้อมูล
- การใช้แถวลำดับเก็บข้อมูล
- การใช้วงวนเพื่อประมวลผลแฟ้มข้อมูล และการประมวลผลอาเรย์

เนื้อหา

สลากกินแบบรัฐบาลออกจำหน่ายเดือนละสองครั้ง ผู้คนส่วนใหญ่สนใจเลขท้ายของรางวัลที่ 1 เลขท้าย 3 ตัว และเลขท้าย 2 ตัว ดังตัวอย่างข้างล่างนี้

สำนักงานสลากกินแบ่งรัฐบาล ช่วยราษฎร์ เสริมรัฐ ยึนหยัดคุณธรรม

ผลการออกรางวัลสลากกินแบ่งรัฐบาล

งวดที่ 26 ประจำวันที่ 16 มกราคม พ.ศ. 2554 และเป็น

งวดที่ 25 ของสลากกาชาดพิเศษ และเป็นงวดที่ 10 ของสลากบำรุงการกุศลงวดพิเศษ (คนพิการและศิริราช)

พิมพ์แจก	ตรวจผลรางวัลทันใจ หรือฮ็อตหลังได้ที่ โทร. 1900-1900-10 . 0-2629 - 1000				ตรวจผลทาง Internet www.glo.or.th
รางวัลที่ 1		เลขท้าย 3 ตัว			เลขท้าย 2 ตัว
สลากกินแบ่งรัฐบาล รางวัลละ 2,000,000 บาท สลากกาชาดพิเศษและสลากการกุศล รางวัลละ 3,000,000 บาท	รางวัลละ 2,000 บาท				รางวัลละ 1,000 บาท
281062	185	227	546	758	2 3

น่าสงสัยว่า เลขท้ายสองตัวใด ที่ออกบ่อยสุด และมีเลขใดหรือไม่ ที่ไม่เคยออกเลย

สิ่งที่ต้องเขียน

เขียนโปรแกรม (**Lottery.java**) อ่านแฟ้มข้อมูลการออกสลากกินแบ่งเพื่อหาว่า

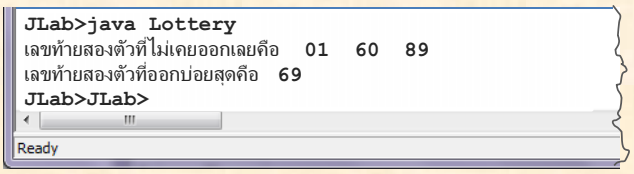
- เลขท้ายสองตัวใด ออกบ่อยครั้งที่สุด (อาจมีมากกว่าหนึ่งตัว)
- เลขใดไม่เคยเป็นผลการออกรางวัลเลขท้ายสองตัวเลย (อาจมีมากกว่าหนึ่งตัว หรือไม่มีเลย)

ข้อมูลขาเข้า (แฟ้ม)

- มีแฟ้ม lottery.txt ภายในเก็บผลการออกสลากกินแบ่งรัฐบาล แต่ละบรรทัดประกอบด้วยวันเดือนปีของงวดที่ออก เลขรางวัลที่ 1 เลขท้ายสามตัว และเลขท้ายสองตัว ดังตัวอย่างข้างล่างนี้

```
01/16/2538  922388  186  667  253  022  40
02/01/2538  198162  195  087  805  574  48
. . .
01/07/2553  480239  450  893  918  961  68
```

ผลลัพธ์ : แสดงทางจอภาพ 2 บรรทัด ดังตัวอย่างข้างล่างนี้ (แฟ้มข้อมูลทดสอบอาจไม่ได้ผลตามที่แสดงในตัวอย่าง)



```
JLab>java Lottery
เลขท้ายสองตัวที่ไม่เคยออกเลยคือ  01  60  89
เลขท้ายสองตัวที่ออกบ่อยสุดคือ  69
JLab>JLab>
```

Ready

ข้อแนะนำ

โจทย์ข้อนี้ต้องการแค่เลขท้ายสองตัวซึ่งเป็นส่วนท้ายสุดของบรรทัด การอ่านส่วนนี้มาใช้กระทำได้หลายวิธี เช่น

- อ่านมาหนึ่งบรรทัดด้วย **`nextLine`** แล้วหยิบตัวที่ 40 กับ 41 ด้วย **`substring`** ได้เลขท้ายสองตัวที่ต้องการ (ตำแหน่งที่ 40 กับ 41 ได้มาจากการนับตำแหน่งของข้อมูลในแฟ้มตัวอย่าง)
- อ่านมาหนึ่งบรรทัดด้วย **`nextLine`** หาตำแหน่งของช่องว่างตัวขวาสุด (ด้วย **`lastIndexOf`** ของสตริง) แล้วค่อยหยิบเลขท้ายสองตัวออกมาด้วย **`substring`**
- ให้สังเกตว่า แต่ละบรรทัดมีข้อมูลอยู่ 7 ส่วน แต่ละส่วนคั่นด้วยช่องว่าง ส่วนสุดท้ายคือ เลขท้ายสองตัวที่ต้องการในแต่ละบรรทัด ลองคิดถึงเมท็อด **`next`** และ **`nextInt`** ของ **`Scanner`** ดู น่าจะใช้ง่ายกว่าสองวิธีแรกที่นำเสนอข้างต้น

วิธีแรกมีข้อเสียตรงที่ตำแหน่งของข้อมูลในแฟ้มต้องตรงตามที่นับ ห้ามขาดห้ามเกิน หากมีสักบรรทัดมีช่องว่างคั่นสักสามตัว หรือเหลือตัวเดียว ก็ผิด ถ้าจะใช้วิธีที่สอง ก็อย่าลืมต้อง **`trim`** สตริงที่อ่านเข้ามาก่อน ไม่เช่นนั้น หากมีช่องว่างทางขวาสักตัว ก็จะค้นได้ตำแหน่งที่ผิด สำหรับวิธีที่หนึ่งและสองนั้น สิ่งที่ได้มาเป็นสตริง ถ้าต้องการเปลี่ยนเป็นจำนวน ก็ต้องใช้บริการ **`Integer.parseInt( … )`** เพื่อเปลี่ยนสตริงเป็น **`int`** แต่ถ้าเลือกวิธีที่สาม ก็ต้องเข้าใจการทำงานของ **`next`** และ **`nextInt`** ของ **`Scanner`** (หากลืมไปแล้วก็เปิดหนังสือหน้าที่ 22)

เมื่ออ่านเลขท้ายสองตัวมาได้แล้ว ต้องทำอะไรต่อ หากยังคิดไม่ออก ขอให้คิดกรณีหาเลขท้ายสองตัวที่ไม่เคยออกก่อน (จะง่ายกว่าหาเลขท้ายสองตัวที่ออกบ่อยสุด) เราจะรู้ว่าเลขใดไม่เคยออก ก็ต้องนับว่า เลขแต่ละตัวออกไปแล้วกี่ครั้ง 00 ออกกี่ครั้ง 01 ออกกี่ครั้ง … 99 ออกกี่ครั้ง ถ้านับได้หมด เลขที่ไม่เคยออก ก็คือเลขที่ออกไปแล้ว 0 ครั้งนั่นเอง การจะนับให้กับทุกตัว ก็ต้องมีตัวแปร 100 ตัวในการจำจำนวนครั้งที่เคยออก เราควรสร้างตัวแปร 100 ตัว 100 ชื่อหรือ? ไม่ควรแน่ ๆ แต่เราควรใช้อะไรมาเก็บ อะไรล่ะที่สร้างทีเดียวได้ 100 ช่อง แต่มีชื่อเดียว ถ้ายังคิดไม่ออก ขอให้ไปอ่านตัวอย่าง *โปรแกรมเก็บสถิติคะแนนสอบ* ในหนังสือหน้าที่ 161 ทำในทำนองเดียวกัน อ่านเลขท้ายสองตัวแต่ละตัว แล้วไปเพิ่มในช่องที่แทนตัวนับของเลขท้ายนั้น อ่านแฟ้มจนหมด แล้วใช้วงวนหาช่องที่มีค่าเป็น 0 ซึ่งแทนการไม่เคยออกเลยนั่นเอง

สำหรับเลขท้ายสองตัวที่ออกบ่อยสุด คงต้องมีสองวงวน วงวนหนึ่งเพื่อหาจำนวนครั้งที่เคยออกมากสุด และอีกวงวนเพื่อหาว่าช่องใดเก็บค่าที่เท่ากับค่าที่มากสุดนั้นบ้าง

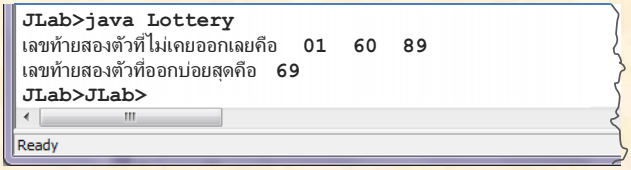
อ่านเพิ่มเติม

- “เริ่มเรียนเขียนโปรแกรม” บทที่ 5 (เรื่องการอ่านแฟ้ม) และบทที่ 7 (เรื่องแถวลำดับ 1 มิติ)

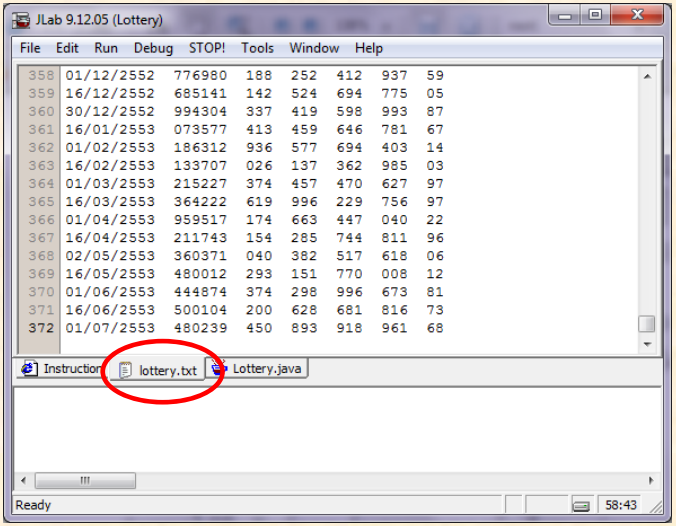
รหัสต้นฉบับเริ่มต้น

```
import java.util.Scanner;
import java.io.*;

public class Lottery {
  public static void main(String[] args) throws IOException {
    Scanner in = new Scanner(new File("lottery.txt"));









    System.out.print("เลขท้ายสองตัวที่ไม่เคยออกเลยคือ ");
    // แสดงรายการของเลขที่ไม่เคยออก
    // แสดงต่อ ๆ กันไปบนบรรทัดเดียวกัน  แต่ละตัวคั่นด้วยช่องว่าง เช่น  12  41  32
    // ถ้าไม่มีเลขที่ไม่เคยออก ก็ไม่ต้องแสดงอะไร









    System.out.println();
    System.out.print("เลขท้ายสองตัวที่ออกบ่อยสุดคือ ");
    // แสดงรายการของเลขที่ออกบ่อยสุด
    // แสดงต่อ ๆ กันไปบนบรรทัดเดียวกัน แต่ละตัวคั่นด้วยช่องว่าง เช่น  69  18
    // เลขที่ออกบ่อยสุด อาจมีมากกว่าหนึ่งตัว









    System.out.println();
  }
}
```

- กรณีต้องการทดสอบด้วยตนเอง : กดปุ่ม F5 เพื่อสั่งโปรแกรมทำงานเริ่มที่เมท็อด **main** ในชุดทดสอบมีแฟ้ม lottery.txt ที่เก็บสถิติการออกสลากตั้งแต่ปี พ.ศ. 2538 ถึง 2553 ถ้าเขียนโปรแกรมได้ถูกต้อง จะได้ผลดังแสดงข้างล่างนี้



ถ้าอยากจะลองเปลี่ยนข้อมูลแล้วลองดูใหม่ ก็ทำได้ โดยกดที่หน้า lottery.txt (แสดงดังรูปข้างล่างนี้) แล้วเพิ่ม ลบ หรือเปลี่ยนข้อมูล เช่น ลองเพิ่มบรรทัดใหม่ที่มีเลขท้าย 01 ดู สั่งทำงานใหม่ ก็น่าจะได้ผลที่ไม่มี 01 อยู่ในรายการของเลขท้ายที่ไม่เคยออก เป็นต้น (เมื่อเปลี่ยนแปลงแฟ้ม lottery.txt แล้วก็สั่งทำงานได้เลย ระบบจะบันทึกแฟ้มให้อัตโนมัติ)



- กรณีต้องการให้ระบบตรวจให้คะแนนอัตโนมัติ : กดปุ่ม F6 ระบบจะสั่งให้โปรแกรมตัวตรวจทำงาน โดยตัวตรวจจะ*สร้างข้อมูลสุ่ม* บันทึกลงแฟ้ม lottery.txt แล้วสั่งเมท็อด **main** ของคลาส **Lottery** ทำงาน จากนั้นอ่านผลลัพธ์กลับมาตรวจสอบว่า ถูกต้องหรือไม่ กระทำการทดสอบเช่นนั้น 10 ครั้ง แล้วรายงานคะแนนที่ได้

  หมายเหตุ : ข้อมูลที่ตัวตรวจสร้างเป็นแบบสุ่ม อยู่ในกรอบที่กำหนดไว้ คือ หนึ่งบรรทัดในแฟ้มมีข้อมูล 7 ส่วน แต่ละส่วนคั่นด้วยช่องว่าง การตรวจแต่ละครั้งจึงมีข้อมูลทดสอบที่ต่างกัน แต่มีรูปแบบเดียวกัน

แบบฝึกหัดเพิ่มเติม

1. จงเขียนชุดคำสั่งที่รับชื่อแฟ้มจากผู้ใช้ทางแป้นพิมพ์ แล้วแสดงจำนวนบรรทัดของแฟ้มนี้ทางจอภาพ

2. จงเขียนชุดคำสั่งที่รับชื่อแฟ้มจากผู้ใช้ทางแป้นพิมพ์ แล้วแสดงทางจอภาพว่า แฟ้มนี้มีสตริง “love” อยู่กี่แห่ง

3. จงเขียนชุดคำสั่งที่อ่านแฟ้ม lottery.txt เพื่อแสดงความถี่ของตัวเลข 0 ถึง 9 ที่ออกในเลขท้ายสองตัว

4. แฟ้ม song.txt เก็บเนื้อเพลง (ภาษาอังกฤษ) จงเขียนโปรแกรมเพื่อแสดงเนื้อเพลงจาก song.txt แต่แสดงกลับลำดับ (เช่น I see skies of blue and clouds of white ก็แสดง white of clouds and blue of skies see I)

5. แทนที่จะหาเลขท้ายสองตัวที่ออกบ่อยสุด จงเขียนโปรแกรมที่อ่านจากแฟ้ม lottery.txt เพื่อหาว่า เลขท้ายสามตัวใด ที่ออกบ่อยสุด

ปฏิบัติการที่ 6 : ฐานนิยม

ผลการเรียนรู้

- การเขียนเมท็อดที่มีการรับข้อมูลขาเข้าและคืนผลลัพธ์
- การเรียกเมท็อดหนึ่งจากอีกเมท็อดหนึ่ง
- การประมวลผลข้อมูลในอาเรย์

เนื้อหา

- ฐานนิยม (mode) ของข้อมูลชุดหนึ่งคือ ข้อมูลที่ปรากฏซ้ำกันในชุดนั้นเป็นจำนวนมากที่สุด
  - เช่น ฐานนิยมของ {2,1,2 ,1,3,4,8,1,1} คือ 1
  - ถ้ามีจำนวนซ้ำมากสุดเท่ากันหลายค่า ให้ค่าใดเป็นฐานนิยมก็ได้
- ตัวหมู่มาก (majority) ของข้อมูลชุดหนึ่งคือ ข้อมูลที่ปรากฏซ้ำกันในชุดนั้นเป็นจำนวนเกิน 50%
  - เช่น ตัวหมู่มากของ {2,1,2 ,1,1,1,1,1} คือ 1
  - {2,1,2 ,1,1,1,2,3} ไม่มีตัวหมู่มาก (มีฐานนิยม แต่มีซ้ำกันเป็นจำนวนไม่เกินครึ่ง)

สิ่งที่ต้องเขียน

เมท็อด 3 เมท็อดในคลาส **`Utils`** ดังนี้

`public static int count(int[] d, int x)`

- คืนจำนวนครั้งที่ **`x`** ปรากฏในอาเรย์ **`d`**
- เช่น **`d = {1,1,2,3,1,3}, count(d,3)`** จะคืน **`2`**, **`count(d,1)`** คืน **`3`**

`public static int mode(int[] d)`

- คืนฐานนิยมของข้อมูลในอาเรย์ **`d`**
- เช่น **`d = {1,1,2,3,1,3}, mode(d)`** จะคืน **`1`**

`public static boolean majority(int[] d)`

- คืน **`true`** เมื่อ **`d`** มีตัวหมู่มาก ถ้าไม่มี คืน **`false`**
- เช่น **`d = {1,1,2,3,1,3}, majority(d)`** จะคืน **`false`**
- **`d = {1,1,2,1,1,3}, majority(d)`** จะคืน **`true`**

สำหรับ **`mode`** และ **`majority`** ให้ถือว่า **`d`** ที่ได้รับเป็นอาเรย์ที่มีขนาดอย่างน้อยหนึ่งช่องขึ้นไป

ข้อแนะนำ

- ควรเขียน **count** ให้เสร็จก่อน จากนั้นเขียน **mode** แล้วค่อยตามด้วย **majority**
- อ่านข้อกำหนดของแต่ละเมท็อดอย่างละเอียด โดยเฉพาะ **mode** ให้คืนค่าที่เป็นฐานนิยม ไม่ใช่ตำแหน่ง ไม่ใช่ความถี่
- เมื่อเขียนหนึ่งเมท็อดเสร็จแล้ว ก็ทดสอบความถูกต้องเลย ไม่ต้องรอเขียนให้เสร็จหมดทั้งสามเมท็อด
- เนื่องจากอาเรย์ที่เราได้รับมาในเมท็อด เป็นอาเรย์เดียวกับที่ผู้ส่งส่งมาให้ จึงไม่ควรแก้ไขค่าของอาเรย์ นำค่าไปใช้ก็พอ อย่าแก้ไข เพราะไม่มีความจำเป็นต้องทำเช่นนั้น
- เมท็อดมีประโยชน์มากๆ เมื่อถูกนำไปใช้งาน ขณะกำลังเขียนเมท็อดหนึ่ง หากสามารถเรียกใช้เมท็อดที่เคยเขียนไปแล้วให้เป็นประโยชน์ได้ ย่อมดีมากๆ เพราะนี่คือวัตถุประสงค์หลักประการหนึ่งการเขียนเมท็อด
- ถ้าเขียนคำสั่งต่างๆ ไปเรื่อย ๆ แล้วพบว่า ซ้ำกับชุดคำสั่งที่เคยเขียนมา ก็เป็นโอกาสดีที่จะดึงส่วนซ้ำกันออกเป็นเมท็อด
- การเรียกเมท็อดซ้ำแล้วซ้ำอีกด้วยข้อมูลชุดเดิมที่ส่งให้เมท็อด ย่อมเสียเวลาโดยเปล่าประโยชน์ ทำไมถึงไม่เรียกครั้งแรกแล้วเก็บผลไว้ในตัวแปร เพื่อนำมาใช้ใหม่ในภายหลัง ประหยัดเวลา ไม่ต้องเรียกซ้ำๆ เช่น พิจารณาเมท็อดข้างล่างนี้

```
public static int f(int n) {
  return a(n-1) + a(n-1) + a(n-1);
}
public static int g(int n) {
  int x = a(n-1);
  return x + x + x;
}
public static int h(int n) {
  return 3*a(n-1);
}
public static int a(int n) {
  int x = 0;
  for (int i=0; i<n; i++) {
    x += i*i*i*i;
  }
  return x;
}
```

เมท็อด **f** เรียกใช้ **a** ด้วยการเรียก **a(n-1)** สามครั้ง ทำไมไม่ทำแบบที่แสดงในเมท็อด **g** ที่เรียก **a(n-1)** ได้ผลเก็บไว้ แล้วค่อยบวกกันสามครั้ง หรือไม่ก็เขียนแบบเมท็อด **h** ที่เรียก **a(n-1)** แล้วคูณด้วย **3**

- ควรเขียนเมท็อด **main** เพิ่มเพื่อทดสอบการทำงานของแต่ละเมท็อดที่เขียนขึ้น

อ่านเพิ่มเติม

- “เริ่มเรียนเขียนโปรแกรม” บทที่ 6 (เรื่องเมท็อด) และบทที่ 7 (เรื่องอาเรย์ 1 มิติ)

รหัสต้นฉบับเริ่มต้น

```
public class Utils {
  //---------------------------------------------------------------
  public static int count(int[] d, int x) {
  }
  //---------------------------------------------------------------
  public static int mode(int[] d) {
  }
  //---------------------------------------------------------------
  public static boolean majority(int[] d) {
  }
  //---------------------------------------------------------------
  public static void main(String[] args) {
  }
  //---------------------------------------------------------------
}
```

- กรณีต้องการทดสอบด้วยตนเอง : กดปุ่ม F5 เพื่อสั่งโปรแกรมทำงานเริ่มที่เมท็อด **main** เนื่องจากคลาส **Utils** ไม่ได้เขียนเมท็อด **main** ไว้ก่อน จึงควรเขียนชุดคำสั่งให้กับ **main** ให้เรียบร้อย อันประกอบด้วย คำสั่งสร้างอาเรย์ ตั้งค่าเริ่มต้น เรียกเมท็อดที่ต้องการทดสอบ แสดงผลลัพธ์ทางจอภาพ แล้วตรวจสอบเองว่า ได้ผลการทำงานของเมท็อดถูกต้องหรือไม่ อย่าลืมว่า ไม่จำเป็นต้องเขียนให้เสร็จทั้งสามเมท็อด แล้วค่อยทดสอบ เขียนเมท็อดใดเสร็จ ก็ทดสอบเมท็อดนั้นทันที ถ้าทำงานถูกต้อง ก็จะได้สบายใจว่า หากต้องการเรียกใช้บริการของเมท็อดที่ทดสอบแล้วในเมท็อดอื่น จะได้ทำงานถูกต้องแน่ๆ

- กรณีต้องการให้ระบบตรวจให้คะแนนอัตโนมัติ : กดปุ่ม F6 ระบบจะสั่งให้โปรแกรมตัวตรวจทำงาน โดยตัวตรวจจะสร้างอาเรย์และเติมข้อมูลทดสอบหลายรูปแบบ เรียกเมท็อดทั้งสาม และตรวจผลว่า ถูกต้องหรือไม่ กระทำการทดสอบเช่นนั้น หลาย ๆ ครั้ง แล้วรายงานคะแนนที่ได้ กรณีที่ควรคำนึงนอกจากการทำงานที่ถูกต้องตามข้อกำหนด มีดังนี้

    - เมท็อดทั้งสามต้องไม่เปลี่ยนแปลงค่าใด ๆ ในอาเรย์ที่ได้รับ (ไม่มีความจำเป็นต้องทำเช่นนี้ เมท็อดทั้งสามมีหน้าที่ให้บริการดูลักษณะอะไรบางอย่างของข้อมูลขาเข้าเท่านั้น)
    - อาเรย์ที่ป้อนให้เมท็อด **mode** และ **majority** มีขนาดอย่างน้อยหนึ่งช่อง (เพราะถ้าส่งไปศูนย์ช่อง ความหมายของ **mode** และ **majority** จะไม่ค่อยชัดเจน ตัวตรวจจึงส่งอาเรย์อย่างน้อยหนึ่งช่อง) แต่สำหรับ **count** นั้น อาจเป็นศูนย์ช่องได้
    - ขอเน้นอีกครั้งว่า **mode** ต้องคืนค่าของฐานนิยม ไม่ใช่ตำแหน่งของอาเรย์ และไม่ใช่ความถี่ของฐานนิยม

แบบฝึกหัดเพิ่มเติม

1. จงเขียนเมท็อดที่คืน ค่าน้อยสุด จากพารามิเตอร์ที่เป็นจำนวนเต็ม 4 ตัว

2. จงเขียนเมท็อดที่รับ **d** (อาเรย์ของจำนวนเต็ม), **x** (จำนวนเต็ม) และ **j** (จำนวนเต็ม) เพื่อค้นว่า **x** เก็บอยู่ที่ตำแหน่งใดใน **d** โดยเริ่มค้นตั้งแต่ตำแหน่ง **j** จนถึงตำแหน่งสุดท้ายใน **d** ถ้าหาไม่พบให้คืน **-1**

3. จงเขียนเมท็อดที่หาว่าในอาเรย์ของจำนวนเต็ม **d** ที่ได้รับ มีตัวซ้ำกันหรือไม่ (ให้เขียนโดยใช้เมท็อดในข้อที่ 2 ให้เป็นประโยชน์ อย่าใช้เมท็อด **count** ที่เคยเขียนมา)

4. จงเขียนเมท็อดเพื่อตรวจสอบว่าอาเรย์ของจำนวนจริง **d** ที่ได้รับ เก็บข้อมูลที่เรียงจากน้อยไปมาก (ซ้ายไปขวา) หรือไม่

5. จงเขียนเมท็อด **copy** เพื่อทำสำเนาอาเรย์ของจำนวนเต็ม **d** ที่ได้รับ ถ้าเรียก **`int[] a = copy(d);`** จะได้อาเรย์ **a** ที่มีขนาดและข้อมูลภายในเหมือนกับของ **d** ทุกประการ

6. จงเขียนเมท็อดที่คืน ค่าหารร่วมมาก ของจำนวนเต็ม 2 จำนวนที่ได้รับ (วิธีหา หรม. หาได้จากอินเทอร์เน็ต หรือไม่ก็ดูในหนังสือหน้าที่ 66)

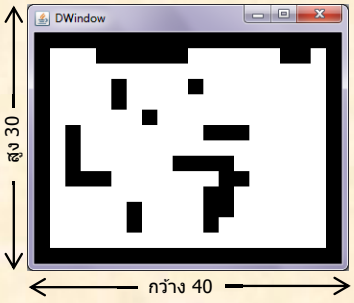
ปฏิบัติการที่ 7 : เลียบกำแพง

ผลการเรียนรู้

- การใช้อาเรย์สองมิติ
- การใช้คำสั่งทดสอบ **if** และ **if-else** กับเงื่อนไขซับซ้อนที่มีตัวดำเนินการตรรกะ

เนื้อหา

กำหนดให้ห้องห้องหนึ่งมีผนังล้อมรอบสี่ด้าน และมีกำแพงถูกสร้างขวางไว้ในห้องแบบสุ่ม ดังรูป

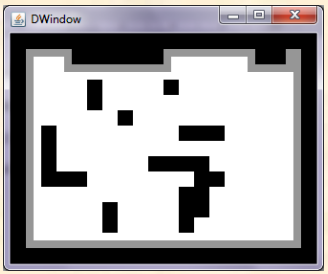


สีดำแทนกำแพง สีขาวแทนที่ว่าง ห้องนี้มีขนาด $w \times h$ (กว้าง × สูง) หากห้องนี้กว้าง 40 สูง 30 และแบ่งห้องนี้ออกเป็นช่องๆ ช่องหนึ่งกว้าง 1 สูง 1 ก็สามารถแทนห้องนี้ได้ด้วยอาเรย์สองมิติ **int[40][30]** คือ กว้าง 40 และ สูง 30 สมมติว่าอาเรย์นี้ชื่อ **m** จะได้ว่า **m[0][0]** แทนตำแหน่งมุมซ้ายบน และ **m[39][29]** แทนตำแหน่งมุมขวาล่าง ดังนั้น **m[x][y]** จึงอ้างอิงตำแหน่ง (x, y) ของห้อง แต่ละช่องในอาเรย์เก็บค่า 0 (แทนที่ว่าง) หรือ 1 (แทนกำแพง)

คำเตือน : อย่าไปคิดว่า เมื่อนำแต่ละช่องในอาเรย์มาวางทาบบนห้องแล้วตำแหน่งในอาเรย์จะตรงกับของห้อง เพราะถ้าเป็นเช่นนั้น จะต้องอ้างอิงตำแหน่ง (x, y) ของห้องด้วย **m[y][x]** ของอาเรย์ ที่กำหนดให้ลักษณะของอาเรย์มีรูปแบบที่ **m[x][y]** อ้างอิงตำแหน่ง (x, y) ของห้องนั้น ก็เพื่อให้เราเขียน x ก่อน y ตามความคุ้นเคยในการเขียนระบบพิกัด

ในคลาส **Maze** มีเมท็อด **createMaze** (เขียนไว้แล้ว) ทำหน้าที่สร้างห้องและเติมกำแพงในห้องแบบสุ่ม โดยประกันว่า กำแพงและช่องว่างทั้งหลายจะกว้างอย่างน้อย 2 ช่องเสมอ (เพื่อให้การเดินเข้าซอกใด ๆ ในห้องนี้ เมื่อเดินเข้าได้ จะสามารถเดินสวนออกมาได้เสมอ โดยไม่ทับทางเดิม) เส้นดำที่เห็นในรูปมีความหนา 2 ช่อง

สิ่งที่ต้องเขียน



เริ่มจากตำแหน่ง (2,2) จงหาทางเดินเลียบกำแพงตามเข็มนาฬิกาไปเรื่อย ๆ จนวกกลับมาที่ตำแหน่งตั้งต้น เปรียบเสมือนเริ่มที่ (2,2) หน้าหันไปด้านทิศตะวันออก ใช้มือซ้ายแตะกำแพง แล้วเดินไปเรื่อย ๆ โดยมือซ้ายแตะกำแพงไปตลอดการเดินทาง รูปทางขวานี้แสดงทางเดินเลียบกำแพงด้วยเส้นสีเทา เมท็อด **findPath** ข้างล่างนี้หาทางเดินเลียบกำแพงแบบตามเข็ม

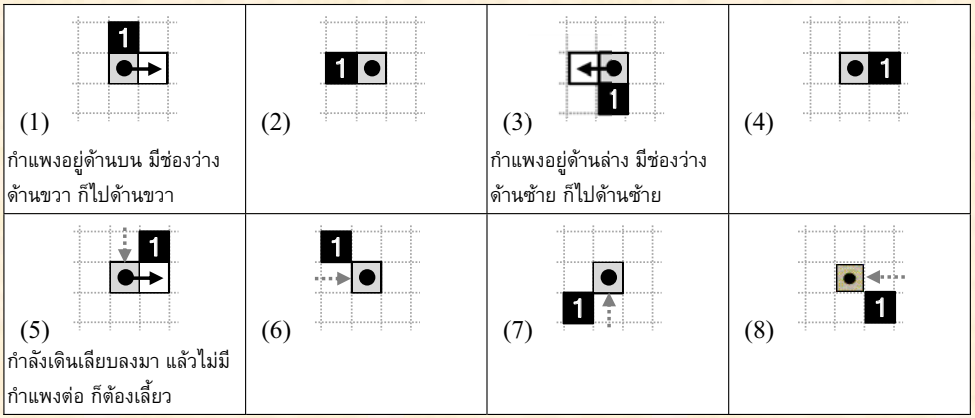
```
int[][] findPath(DWindow w, int[][] m)
```

โดยที่อาเรย์ **m** แทนห้อง **findPath** เติมทางเดินเลียบกำแพงในอาเรย์ **m** โดยเติมค่า 2 ในช่องที่เป็นทางเดิน แล้วคืนอาเรย์ของห้องนี้กลับคืนเป็นผลลัพธ์ สิ่งที่เขียนให้แล้วใน **findPath** คือ วงวนสำหรับการเติมทางเดินรอบละหนึ่งช่อง (เติมช่องไหน อย่างไร ต้องเขียนเอง) ในแต่ละรอบจะเรียกเมท็อด **show(w,m)** เพื่อแสดงทางเดินในวินโดว์ **w** ทำให้เห็นความก้าวหน้าของการเติมทางเดิน

ข้อแนะนำ

ตัวแปร **x** และ **y** ใน **findPath** เก็บตำแหน่งปัจจุบันของการเดินเลียบกำแพง (เริ่มที่ 2,2) ดังนั้น **m[x][y]** ต้องมีค่าเป็น 2 เพราะได้เดินมาถึง (x, y) เนื่องจากเป็นการเดินเลียบกำแพง จึงต้องมีอย่างน้อยหนึ่งช่องที่ติดกับตำแหน่ง (x, y) เป็นกำแพง, นั่นคือ หนึ่งหรือมากกว่าของค่า **m[x-1][y-1]**, **m[x+1][y]**, **m[x-1][y+1]**, **m[x][y-1]**, **m[x][y+1]**, **m[x+1][y-1]**, **m[x+1][y]** หรือ **m[x+1][y+1]** เป็น 1 ภาระที่ต้องทำในวงวนแต่ละรอบคือ จะเปลี่ยน **x** หรือ **y** อย่างไรหนึ่งตำแหน่ง (นั่นคือจะเป็น **x--**, **x++**, **y--** หรือ **y++**) แทนการเดินอีกหนึ่งช่องนั่นเอง

แล้วจะรู้ได้อย่างไร ? สิ่งที่ต้องเน้นตรงนี้ก่อนคือ เราเลือกที่จะเดินแบบตามเข็ม คือเริ่มที่ (2,2) เดินเลียบไปทางตะวันออก นั่นคือ ถ้ามีกำแพงอยู่ด้านบน ก็เลือกเดินไปทางขวา ถ้ากำแพงอยู่ด้านล่าง ก็เลือกเดินไปทางซ้าย ถ้าพิจารณาให้รอบคอบ จะพบว่ามีทั้งหมด 8 กรณี แสดงด้วยรูปข้างล่างนี้ ช่องปัจจุบันแทนด้วย ● กำแพงแทนด้วย 1 และเส้นลูกศรทึบแทนทิศทางการเดินไปยังช่องถัดไปที่แทนด้วย □ กรณีที่ (1) ถึง (4) เป็นกรณีที่ควรตรวจสอบก่อน ในขณะที่ 4 กรณีล่างแทนกรณีเดินเลี้ยว เช่น กรณีที่ (1) แทนกรณีที่มีกำแพงด้านบน และช่องทางขวาว่าง ก็เดินไปทางขวา สำหรับกรณีที่ (5) เป็นกรณีที่ด้านบนของช่องปัจจุบันคงไม่ใช่กำแพง (ถ้าใช่ย่อมตรงกับกรณีที่ (1)) น่าจะมาจากกรณีที่เดินมาจากด้านบน (มาจากกรณีที่ (4)) เลียบกำแพงมาเรื่อย ๆ แล้วไม่พบกำแพงทางขวา จึงต้องเลี้ยว ลองเติมรายละเอียดในแต่ละกรณีที่เหลือ



เมื่อเข้าใจกรณีต่าง ๆ ครบถ้วนแล้ว ก็เปลี่ยนแต่ละกรณีไปเป็นคำสั่งทดสอบว่า ช่องปัจจุบันตรงตามกรณีใด เช่น กรณีที่ (1) ก็น่าจะเปลี่ยนเป็นอะไรทำนองนี้ (เขียนเป็นคำสั่งภาษาจาวาเอง)

ถ้าช่องด้านบนและช่องทางขวาของช่องที่อยู่ตอนนี้เป็นกำแพงและช่องว่างตามลำดับ ก็ให้เดินไปทางขวา (ซึ่งคือการเพิ่ม x ไปหนึ่งตำแหน่ง)

อ่านเพิ่มเติม

- "เริ่มเรียนเขียนโปรแกรม" บทที่ 4 (คำสั่ง if, if-else) และบทที่ 7 (อาเรย์ 2 มิติ)

รหัสต้นฉบับเริ่มต้น

```
import jlab.graphics.DWindow;
import java.util.*;

public class Maze {
  static boolean testMode = false;
  public static void main(String[] args) {
    int width = 40;
    int height = 30;
    DWindow w = new DWindow(width * 8, height * 8);
    w.setRepaintDuringSleep(true);
    int[][] maze = createMaze(width, height);
    findPath(w, maze);
    System.out.println("Done.");
  }
  //------------------------------------------------------------------
  // เดินทางเดินเลียบผนังแบบตามเข็มนาฬิกา เริ่มที่ (2,2) เดินจนวนกลับมาที่จุดเริ่มต้น
  //
  public static int[][] findPath(DWindow w, int[][] m) {
    int x = 2, y = 2;
    m[x][y] = 2;
    show(w, m);
    while (!(x == 2 && y == 3)) {
     // เขียนคำสั่งเพื่อเพิ่มหรือลดค่าของ x หรือ y ไปหนึ่ง แทนทางเดินช่องถัดไป ในแต่ละรอบ
```

```
      m[x][y] = 2;  // เติมเป็นทางเดิน
      show(w, m);   // แสดงภาพของห้องใหม่พร้อมทางเดิน
    }
    return m;
  }
```

```
  //-----------------------------------------------------------------
  // แสดงห้องในวินโดว์ w   ผนังสีดำ (1)  ,  ทางเดินสีแดง (2) ,  ที่ว่างที่ยังไม่เดินผ่านสีขาว (0)
  //
  private static void show(DWindow w, int[][] m) {
    if (testMode) return;
    w.clearBackground();
    int width = w.getWidth() / m.length;
    for (int x = 0; x < m.length; x++) {
      for (int y = 0; y < m[x].length; y++) {
        if (m[x][y] == 1)
          w.fillRect(DWindow.BLACK, x * width, y * width, width, width);
        if (m[x][y] == 2)
          w.fillRect(DWindow.RED, x * width, y * width, width, width);
      }
    }
    w.sleep(1);
  }
  //-----------------------------------------------------------------
  // สร้างห้องขนาด width x height ที่มีผนังกั้นสี่ด้าน มีผนังสุ่ม ๆ ภายในห้อง
  // ทุกผนังมีความกว้างอย่างน้อย 2 และทุกช่องว่าง (ที่ให้เดินได้) มีความกว้างอย่างน้อย 2 เช่นกัน
  //
  public static int[][] createMaze(int width, int height) {
    // ขอสร้าง maze ขนาดเป็นครึ่งหนึ่งของที่ต้องการ แล้วค่อยขยายทีหลัง
    // เพื่อให้ทางเดินมีขนาดสองช่องเสมอ จะเดินเดินไปและกลับได้
    int w2 = width / 2, h2 = height / 2;
    int[][] m = new int[w2][h2];
    for (int y = 0; y < h2; y++)
      m[0][y] = m[w2 - 1][y] = 1; // เติมผนังด้านซ้ายและขวา
    for (int x = 0; x < w2; x++)
      m[x][0] = m[x][h2 - 1] = 1; // เติมผนังด้านบนและล่าง
    int t = Math.max(width, height) / 3; // จำนวนการเติมผนังในห้อง
    for (int k = 0; k < t; k++) {
      int n = (int) (Math.max(w2, h2)/3 * Math.random());     // ความยาวผนัง
      int x = (int)(w2 * Math.random());              // ตำแหน่งเริ่มต้น
      int y = (int)((h2 - n) * Math.random());
      for (int i = 0; i < n; i++) m[x][y + i] = 1;  // เติมผนังแนวตั้ง
      // เติมผนังแนวนอนในห้อง
      x = (int)((w2 - n) * Math.random());            // ตำแหน่งเริ่มต้น
      y = (int)(h2 * Math.random());
      for (int i = 0; i < n; i++) m[x + i][y] = 1;  // เติมผนังแนวนอน
    }
    m[1][1] = 0; // เติมช่องว่างที่มุมซ้ายบน (จุดเริ่ม)
    //-----------------------------------------
    // สร้างห้องเท่าของจริง ย้ายข้อมูลจากห้องเล็กมาเติมในห้องใหญ่
    //
    int[][] maze = new int[width][height];
    for (int x = 0; x < width; x++) {
      for (int y = 0; y < height; y++) {
        maze[x][y] = m[x / 2][y / 2];
      }
    }
    return maze;
  }
}
```

- กรณีต้องการทดสอบด้วยตนเอง : กดปุ่ม F5 เพื่อสั่งโปรแกรมทำงานเริ่มที่เมท็อด **main** โปรแกรมที่เขียนไว้จะสร้างห้องพร้อมกำแพงแบบสุ่ม แสดงทางจอภาพ แล้วเข้าสู่วงวนหาทางเดิน โดยเมื่อเดินทางเดินหนึ่งช่อง ภาพที่แสดงห้องก็จะเปลี่ยนแปลงตาม ทำให้เห็นทางเดินที่หาได้ยาวขึ้น ๆ หากเขียนคำสั่งได้ถูกต้อง ก็จะวนกลับมาที่จุดเริ่มต้น เป็นอันสิ้นสุดการทำงาน ดังตัวอย่างภาพการเปลี่ยนแปลงข้างล่างนี้ (ดูตามลำดับซ้ายไปขวาทีละบรรทัด)

- กรณีต้องการให้ระบบตรวจให้คะแนนอัตโนมัติ : กดปุ่ม F6 ระบบจะสั่งให้โปรแกรมตัวตรวจทำงาน โดยตัวตรวจจะสร้างห้องที่มีขนาดสุ่มและเติมกำแพงแบบสุ่มด้วย จากนั้นเรียกเมท็อด **findPath** ให้หาทางเดิน ได้ผลกลับมา ก็ตรวจสอบว่า ถูกต้องหรือไม่ โดยไม่ได้แสดงภาพของห้องให้ดู ตัวตรวจกระทำการทดสอบเช่นนี้เป็นจำนวน 10 ห้อง แล้วรายงานคะแนนที่ได้
  หากระหว่างการตรวจปรากฏว่ามีข้อความสีแดงแสดงความผิดพลาดว่า **<<--- TIME OUT --->>** โดยไม่มีผลบอกว่า ok หรือ X แต่รายงานว่าได้ 0 คะแนน อาการเช่นนี้เกิดจากการที่เมท็อดที่เขียนใช้เวลาการทำงานมากผิดปกติ (ตั้งไว้ไม่ให้เกิน 5 วินาที) ระบบจึงตัดการทำงาน เหตุการณ์เช่นนี้มักมีสาเหตุมาจากการเดินทางเดินแล้วไม่สามารถวกกลับมาที่จุดเริ่มต้นเลย ทำให้การทำงานของ **findPath** ไม่สามารถออกจากวงวนได้ ซึ่งคงจะมีข้อผิดพลาดจากเงื่อนไขการตรวจสอบ และการเปลี่ยนค่าของ **x** และ **y** ที่ไม่ถูกต้อง

แบบฝึกหัดเพิ่มเติม

1. ปรับเปลี่ยนเงื่อนไขในเมท็อด **`findPath`** เพื่อให้เดินเลียบกำแพง แบบวนทวนเข็ม

2. หาก **`findPath`** รับพารามิเตอร์ **`x0`**, **`y0`** ที่ระบุตำแหน่งเริ่มต้นในห้อง จงปรับ **`findPath`** ให้หาทางเดินไปจนพบ (2,2)

ปฏิบัติการที่ 8 : ปริศนา 15 แผ่น

ผลการเรียนรู้

- การใช้อาเรย์ 1 และ 2 มิติ
- การใช้วงวนซ้อนกัน เพื่อประมวลผลอาเรย์ทั้ง 1 และ 2 มิติ

เนื้อหา

8	10	7	11
	5	4	14
1	13	12	6
2	9	3	15

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	

1	2	3	4
5	6	7	8
9	10	11	12
13	15	14	

หวังว่าคงเคยเล่นเกม 15 puzzle กันมาก่อน เป็นแผ่นพลาสติกรูปสี่เหลี่ยมจัตุรัส ภายในประกอบด้วยแผ่นพลาสติกย่อยเล็กๆ (สี่เหลี่ยมจัตุรัสเหมือนกัน) จำนวน 15 แผ่น (แต่ละแผ่นมีตัวเลขกำกับตั้งแต่ 1 ถึง 15) วางเรียงกันเป็นตาราง 4×4 โดยมีช่องว่างหนึ่งอยู่ภายใน สามารถเลื่อนแผ่นต่าง ๆ ไปมาได้ในแนวนอนแนวตั้งเมื่ออยู่ติดกับช่องว่าง

จุดประสงค์ของเกมก็คือ ให้เลื่อนแผ่นสี่เหลี่ยมภายในไปมา เพื่อให้แผ่นสี่เหลี่ยมเหล่านี้ เรียงเป็นระเบียบไล่ไปเรื่อย ๆ 1 ถึง 15 (จากซ้ายไปขวา จากบนลงล่าง) ดังตัวอย่างที่แสดง จากรูปบน ให้หาวิธีเลื่อนจนได้ดังรูปล่าง

ประเด็นปัญหาที่เราสนใจในที่นี้คือ มันไม่แน่เสมอไปว่า จากแผ่นเริ่มต้นที่ให้มา (ซึ่งมีรูปแบบที่เป็นไปได้ทั้งสิ้น 15! มากกว่าหนึ่งล้านล้านแบบ) จะมีวิธีเลื่อนกลับให้เป็นระเบียบตามที่ต้องการได้ ตัวอย่างเช่น ถ้าให้แผ่นเริ่มต้นเป็นดังรูปขวานี้ (สังเกตว่า แผ่นนี้ต่างกับแผ่นเป้าหมาย เพียงแค่สลับ 14 กับ 15 เท่านั้น) จะไม่มีทางเลื่อนกลับได้

สิ่งที่ต้องเขียน

จงเขียนเมท็อด **`public static boolean solvable15Puzzle( int [][] b )`** ในคลาส **`FifteenPuzzle`** ซึ่งรับตาราง **`b`** เป็นอาเรย์สองมิติขนาด 4×4 โดยที่ **`b[i][j]`** เก็บหมายเลขของแผ่นสี่เหลี่ยมที่อยู่บนแถวแนวนอนที่ **`i`** และแถวแนวตั้งที่ **`j`** (ช่องว่างจะแทนด้วยหมายเลข 0) ถ้า **`b`** เป็นตารางที่ไม่มีวิธีเลื่อนกลับได้ ให้คืน **`false`** แต่ถ้ามีหนทางเลื่อนกลับได้ให้คืน **`true`**

1	2	3	4
5	6	7	8
9	10	12	15
13	14		11

จะรู้ได้อย่างว่า เลื่อนได้เลื่อนไม่ได้ ? จะบอกวิธีให้ โดยไม่ต้องลองเลื่อน ดังนี้

- นำแผ่นสี่เหลี่ยมเล็กๆ มาวางเรียงเป็นแถวเดียว ไล่จากซ้ายไปขวา บนลงล่าง เช่น จากรูปทางขวานี้ เรียงได้ 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 12, 15, 13, 14, 11
  ข้อสังเกต : เราไม่นำช่องว่าง มาวางเรียงในแถว
- พิจารณาตัวเลขในแถวเป็นคู่ ๆ ทุกคู่ มี 15 ตัว จึงมีทั้งหมด $\binom{15}{2} = (15\times14)/2$ คู่ เพื่อนับว่า มีอยู่กี่คู่ที่ตัวทางซ้ายมากกว่าตัวทางขวา (เรียกว่า *กลับลำดับ*) เช่น จากตัวอย่างข้างบนนี้ คู่ที่กลับลำดับคือ (12,11), (15,13), (15,14), (15,11), (13,11) และ (14,11) จึงมีอยู่ 6 คู่ เรียกจำนวนนี้ว่า L
- ให้ B คือ หมายเลขแถวที่ช่องว่างอยู่ (แถวบนสุดคือแถวที่ 1 ไล่ลงมา)
- ถ้า $L+B$ เป็นจำนวนคู่ แสดงว่า เราสามารถเลื่อนแผ่นสี่เหลี่ยมต่างๆ ไปสู่เป้าหมายได้ แต่ถ้า $L+B$ เป็นจำนวนคี่ แสดงว่า เลื่อนไม่ได้ (จากตัวอย่าง L=6, B=4, $L+B$ = 10 เป็นจำนวนคู่ สรุปว่า เลื่อนได้)
- ขอไม่พิสูจน์ว่า วิธีตรวจสอบข้างบนนี้ใช้ได้ผลจริงได้อย่างไร (ถ้าสนใจ ก็ลองค้นอินเทอร์เน็ต)

ข้อแนะนำ

- การใช้วงวน **for** สองวงซ้อนกัน เพื่อเข้าใช้ข้อมูลในอาเรย์สองมิติทุกช่อง เป็นรูปแบบที่พบบ่อยมากในการประมวลผลอาเรย์สองมิติ (ดูหนังสือหน้าที่ 179) จึงสมควรฝึกเขียนให้คล่อง อย่าลืมว่า
  - จำนวนแถวแนวนอนของอาเรย์สองมิติ **d** คือ **d.length**
  - จำนวนแถวแนวตั้งของอาเรย์สองมิติ **d** คือ **d[0].length** (เมื่อทุกแถวแนวนอนมีข้อมูลเท่ากัน)
- การใช้วงวน **for** สองวงซ้อนกัน เพื่อพิจารณาข้อมูลทุกคู่ที่เก็บในอาเรย์หนึ่งมิติ ก็เป็นอีกรูปแบบหนึ่งที่พบบ่อยมากในการประมวลอาเรย์หนึ่งมิติ จึงควรฝึกเขียนให้คล่องเช่นกัน สมมติว่า อาเรย์มีขนาด 4 ช่อง เราต้องเขียนวงวนเพื่อแจกแจงเลขตำแหน่งของอาเรย์ให้ได้ (4×3)/2 = 6 กรณี ดังนี้

  (0,1), (0,2), (0,3), (1,2), (1,3) และ (2,3)

  ถ้าอาเรย์มี 5 ช่อง ก็ต้องแจกแจงให้ได้ (5×4)/2 = 10 กรณี ดังนี้

  (0,1), (0,2), (0,3), (0,4), (1,2), (1,3), (1,4), (2,3), (2,4), (3,4)

  ส่วนของโปรแกรมข้างล่างนี้ ยังแจกแจงทุกคู่ในอาเรย์ **d** ได้ไม่ถูกต้อง

```
for (int i=0; i<d.length; i++) {
  for (int j=0; j<d.length; j++) {
    ...
  }
}
```

  เพราะถ้า **d** มีขนาด 4 ช่อง ค่าของ **i** และ **j** ที่แจกแจงได้ภายในวงวน จะเป็น

  (0,0), (0,1), (0,2), (0,3), (1,0), (1,1), (1,2), (1,3), (2,0), (2,1), (2,2), (2,3), (3,0), (3,1), (3,2), (3,3)

  ซึ่งมีจำนวนมากเกินไป ถ้าดูให้ดีจะพบว่า คู่ **i** และ **j** ที่มีขีดเส้นใต้ข้างบนนี้ไม่ต้องแจง (ทำไม ?) ลองปรับปรุงวงวน **for** สองวงข้างบนนี้ เพื่อให้แจกแจงข้อมูลครบทุกคู่ คู่ละครั้ง

  การเขียนวงวนแจกแจงทุกคู่ของข้อมูลในอาเรย์ จึงมีข้อควรคำนึงถึงดังนี้
  - คู่ที่ประกอบด้วยตัวเดียวกัน (เช่น คู่ **d[3]** กับ **d[3]**) ต้องนำมาพิจารณาหรือไม่
  - พิจารณาเกินความจำเป็นหรือไม่
    - ที่พบบ่อยมากคือ กรณีที่พิจารณาคู่เดียวกันซ้ำกัน เช่น พิจารณา คู่ **d[2]** กับ **d[3]** แล้วยังมาพิจารณาคู่ **d[3]** กับ **d[2]** อีก
    - อีกกรณีหนึ่งคือ การเข้าใช้ข้อมูลในอาเรย์ที่ตำแหน่งนอกขอบเขต เช่น อาเรย์ **d** มี 10 ช่อง แต่ไปใช้ **d[10]** จะทำให้เกิด **ArrayIndexOutOfBoundsException** ที่แจ้งให้ทราบถึงสิ่งผิดปกติของการทำงาน
  - พิจารณาไม่ครบทุกคู่หรือไม่ กรณีเช่นนี้มักเกิดกับการเขียนเงื่อนไขของวงวนผิด โปรแกรมจะทำงานได้โดยไม่เกิดสิ่งปกติ จำเป็นต้องทดสอบให้รอบคอบ อาจใช้วิธีแสดงค่าตำแหน่งของคู่ข้อมูลออกมาดู สำหรับกรณีที่อาเรย์มีขนาดไม่มาก ดูผลด้วยตาก็พอตรวจสอบเพิ่มความมั่นใจได้

อ่านเพิ่มเติม

- "เริ่มเรียนเขียนโปรแกรม" บทที่ 7

รหัสต้นฉบับเริ่มต้น

```
public class FifteenPuzzle {
  //-----------------------------------------------------
  public static boolean solvable15Puzzle(int[][] b) {
  }
  //-----------------------------------------------------
  public static void main(String[] a) {
    int[][] ok  = { { 1,  2,  3,  4},
                    { 5,  6,  7,  8},
                    { 9, 10, 12, 15},
                    {13, 14, 11,  0} };
    int[][] nok = { { 1,  2,  3,  4},
                    { 5,  6,  7,  8},
                    { 9, 10, 11, 12},
                    {13, 15, 14,  0} };
    System.out.println( solvable15Puzzle(ok) );
    System.out.println( solvable15Puzzle(nok) );
  }
}
```

- กรณีต้องการทดสอบด้วยตนเอง : กดปุ่ม F5 เพื่อสั่งโปรแกรมทำงานเริ่มที่เมท็อด **main** ชุดคำสั่งที่เขียนไว้ที่ **main** มีตัวอย่างตารางเริ่มต้นสองแบบ แบบหนึ่งเลื่อนได้ และอีกแบบเลื่อนไม่ได้ หากต้องการแบบใหม่ ๆ ก็สามารถสร้างได้ด้วยการสุ่มสร้างตาราง จากนั้นสลับสองหมายเลขที่ไม่ใช่ 0 ที่อยู่ติดกัน จะต้องมีแบบหนึ่งเลื่อนได้ อีกแบบเลื่อนไม่ได้แน่ๆ ก็สามารถทดสอบความถูกต้องของโปรแกรมได้หลายกรณีมากขึ้น (ถ้าเป็นกรณีที่สลับกับ 0 ที่อยู่ติดกันบนแถวแนวนอนเดียวกัน ต้องได้ผลเหมือนกัน) ชุดคำสั่งข้างล่างนี้แสดงตัวอย่างการทดสอบสุ่ม 1000 กรณีด้วยวิธีข้างต้น (ต้องเขียนเมท็อด **swap(d,i,j)** เอง เพื่อสลับข้อมูลตัวที่ **i** และ **j** ในอาเรย์ **d**) ลองศึกษาการทำงานดูเอง

```
int[] d = {0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15};
boolean correct = true;
for (int c=0; correct && c<1000; c++) { // ทดสอบมากสุด 1000 แบบ
  for (int i=0; i<d.length; i++) {
     int j = (int)(16*Math.random());   // สุ่มสลับข้อมูลใน d
     swap(d, i, j);                     // เขียนเมท็อด swap เอง
  }
  int[][] b1 = new int[4][4];
  int[][] b2 = new int[4][4];
  for (int i=0, k=0; i<b1.length; i++) {
     for (int j=0; j<b1[0].length; j++) {
        b1[i][j] = b2[i][j] = d[k++];   // สร้างสองตารางสุ่มเหมือนกัน
     }
  }
  int i = (int)(4*Math.random());       // สุ่มแถว 0 ถึง 3
  int j = (int)(3*Math.random());       // สุ่มคอลัมน์ 0 ถึง 2
  swap(b2[i], j, j+1);                  // สลับ b[i][j] กับช่องข้าง

  boolean r1 = solvable15Puzzle(b1);
  boolean r2 = solvable15Puzzle(b2);
  if (b2[i][j] != 0 && b2[i][j + 1] != 0) {
    if (r1 == r2) correct = false;      // ผิด ถ้าไม่มี 0 และผลเหมือน
  } else {
    if (r1 != r2) correct = false;      // ผิด ถ้ามี 0 แต่ผลไม่เหมือน
  }
}
System.out.println(correct);            // true แสดงว่าถูกต้อง
```

- กรณีต้องการให้ระบบตรวจให้คะแนนอัตโนมัติ : กดปุ่ม F6 ระบบจะสั่งให้โปรแกรมตัวตรวจทำงาน โดยตัวตรวจจะสุ่มสร้างตารางขนาด 4×4 จำนวน 10 แบบ ข้อมูลในตารางมีค่า 0 ถึง 15 หนึ่งค่าหนึ่งช่องแน่นอน ไม่มีซ้ำ ไม่มีขาด แล้วเรียกเมท็อด **solvable15Puzzle** นำผลผลลัพธ์กลับมาตรวจสอบว่าถูกต้องหรือไม่ กระทำการทดสอบเช่นนั้น หลาย ๆ ครั้ง แล้วรายงานคะแนนที่ได้ และเนื่องจากเมท็อดที่ให้เขียนนี้ได้ผลแค่ **true** หรือ **false** ตัวตรวจยังทดสอบกรณีที่เขียนชุดคำสั่งแบบสุ่มตอบ หรือแบบเลือกตอบด้วยค่าคงที่ตลอด ซึ่งถือว่าผิด

แบบฝึกหัดเพิ่มเติม

1. จงเขียนเมท็อด **`isValid( int[][] b )`** เพื่อทดสอบว่า อาเรย์ **`b`** ที่ได้รับแทนตารางของปริศนา 15 แผ่นที่ถูกต้องหรือไม่ โดยตารางที่ถูกต้องคือตารางที่มีขนาด 4×4 บรรจุตัวเลข 0 ถึง 15 อย่างละหนึ่งตัว ไม่ขาด ไม่เกิน และต้องเป็นตารางที่เลื่อนกลับไปสู่เป้าหมายที่ต้องการของปริศนา 15 แผ่นได้

2. เราสามารถมองการเลื่อนแผ่นพลาสติกในเกมปริศนา 15 แผ่นให้ขึ้น ลง ซ้าย หรือขวา เป็นการเลื่อนช่องว่างให้ลง ขึ้น ขวา หรือซ้ายก็ได้ จงเขียนเมท็อด **`randomlyMoveBlank( int[][] b, int k )`** ที่รับพารามิเตอร์เป็นอาเรย์สองมิติ (แทนตารางของเกมปริศนา 15 แผ่น) และจำนวนเต็ม **`k`** เพื่อเลื่อนช่องว่างในตารางที่ได้รับนี้สุ่มๆ จำนวน **`k`** ครั้ง (การสุ่มเลื่อนช่องว่างในที่นี้ คือการสุ่มว่าจะเลื่อนขึ้น ลง ซ้าย หรือขวา) ถ้าตารางเริ่มต้นที่ได้รับเป็นตารางที่เลื่อนไปหาเป้าหมายสุดท้ายได้ ตารางที่เลื่อนช่องว่างสุ่มนี้ ย่อมได้ผลที่เลื่อนกลับได้ด้วย

ปฏิบัติการที่ 9 : ภาพซ้อนภาพ

ผลการเรียนรู้

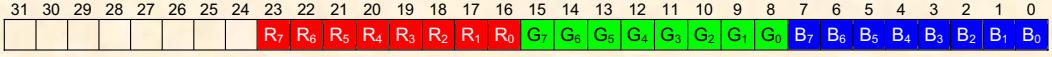
- การประมวลผลอาเรย์สองมิติ
- การประมวลผลภาพแบบพื้นฐาน

เนื้อหา

- ถ้า `w` คือวินโดว์ที่สร้างด้วย `DWindow` เราสามารถอ่านภาพมาแสดงได้ด้วยบริการ `w.loadImage(q)` โดย `q` คือตำแหน่งของแฟ้มภาพ หรือจะเป็น URL ที่เก็บภาพในอินเทอร์เน็ต เช่น วินโดว์ที่แสดงรูปทางขวานี้ได้มาจากคำสั่งข้างล่างนี้

```
DWindow w = new DWindow();
w.loadImage("http://www.eng.chula.ac.th/files/u1/Building2.jpg");
```

- ภาพหนึ่งภาพประกอบด้วยจุดภาพ (pixel) มากมาย เรียงกันเป็นแถว ๆ จากบนลงล่าง
- แต่ละจุดภาพเปล่งแสงมีสีที่ประกอบด้วยองค์ประกอบสี แดง เขียว และน้ำเงิน เรียกย่อว่า R, G, B
- แต่ละสีมีความเข้ม 256 ระดับ เช่น สีแดงระดับ 255 คือ แดงสดสุด ๆ แต่ถ้าระดับ 0 คือไม่มีสีแดงเลย ถ้าเป็นสีม่วงสดสุด ๆ คือ R=255, G=0, B=255
- เราสามารถแทนสีของจุดภาพหนึ่งจุดด้วยจำนวนเต็มแบบ `int` หนึ่งตัว ดังนี้



  - เนื่องจาก $2^8 = 256$ ถ้าต้องการเก็บจำนวนเต็ม 0 ถึง 255 จึงใช้เนื้อที่เพียง 1 ไบต์ ดังนั้น จุดภาพแต่ละจุดใช้เนื้อที่เก็บ 3 ไบต์ (1 ไบต์แทน 1 สี)
  - เนื่องจาก `int` หนึ่งตัวมีขนาด 4 ไบต์ ดังนั้น 1 pixel จึงแทนได้ด้วย `int` หนึ่งตัว โดยกำหนดให้
    - ระดับของสีน้ำเงินใช้บิตที่ 0 ถึงบิตที่ 7 (ดูรูปข้างบนประกอบ)
    - ระดับของสีเขียวใช้บิตที่ 8 ถึงบิตที่ 15
    - ระดับของสีแดงใช้บิตที่ 16 ถึงบิตที่ 23
    - ส่วนบิตที่เหลือทางซ้ายมีไว้แทนระดับความทึบของจุดภาพ, ถ้าเป็น 255 แสดงว่าจุดภาพนี้ทึบ แต่ถ้าเป็น 0 แสดงว่าจุดภาพนี้โปร่งใสมองไม่เห็น (ปฏิบัติการนี้ไม่ได้ใช้ส่วนนี้)
  - การแทนสีของจุดภาพเช่นนี้ แต่ละจุดจึงมีสีได้แตกต่างกันถึง $2^8 2^8 2^8 = 2^{24} = 16{,}777{,}216$ สี
- ภาพหนึ่งภาพจึงแทนได้ด้วยอาเรย์สองมิติของ `int` เรียกว่า *pixel map* (เรียกสั้น ๆ ว่า *pixmap*) โดยสีของจุดภาพที่พิกัด (x,y) ถูกเก็บในอาเรย์ที่ช่อง `[x][y]` เราสามารถดึงจุดภาพของภาพที่แสดงใน `DWindow` ออกมาเป็นอาเรย์สองมิติได้ด้วยบริการ `getPixmap` เช่น

```
DWindow w = new DWindow();
w.loadImage("http://www.eng.chula.ac.th/files/u1/Building2.jpg");
int[][] p = w.getPixmap();
```

  โดย `p.length` แทนความกว้างของภาพ และ `p[0].length` แทนความสูงของภาพ ดังนั้น `p[0][0]` แทนสีของจุดภาพมุมซ้ายบนและ `p[p.length-1][p[0].length-1]` แทนสีของจุดภาพมุมขวาล่าง
- อาจสงสัยว่า ทำไม `p.length` แทนความกว้าง ในเมื่อ `p.length` คือ จำนวนแถวของอาเรย์ ซึ่งน่าจะเป็นความสูงของภาพ ต้องขอเน้นตรงนี้ว่า นี่เป็นความตั้งใจของผู้เขียนเมท็อด `getPixmap` ที่ให้เป็นเช่นนั้น

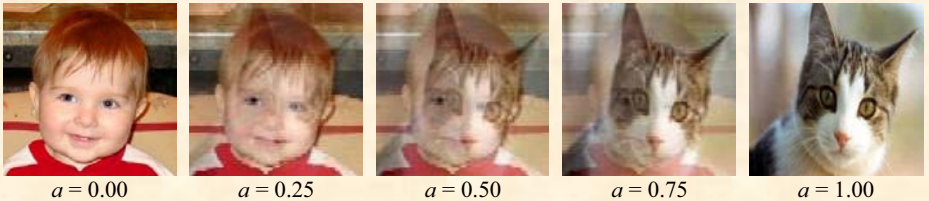
เนื่องจากจะทำให้การใช้จุดภาพที่พิกัด (x, y) เขียนด้วย **p[x][y]** ซึ่งดูคุ้นตามากกว่าการเขียน **p[y][x]**

- เราสามารถนำจุดภาพมาประมวลผล หรือปรับเปลี่ยนสีของจุดภาพในอาเรย์ได้ด้วยบริการของคลาส **DWindow** เพื่อแยกและรวมองค์ประกอบสี ดังนี้
  - **DWindow.getR(c)** คืนค่าขององค์ประกอบสีแดงของสี **c** (มีค่าระหว่าง 0 ถึง 255)
  - **DWindow.getG(c)** คืนค่าขององค์ประกอบสีเขียวของสี **c** (มีค่าระหว่าง 0 ถึง 255)
  - **DWindow.getB(c)** คืนค่าขององค์ประกอบสีน้ำเงินของสี **c** (มีค่าระหว่าง 0 ถึง 255)
  - **DWindow.mixRGB(r,g,b)** คืนค่าสีที่ได้จากการผสมองค์ประกอบสีแดง เขียว น้ำเงิน
- ขอเน้นตรงนี้ว่า การเปลี่ยนค่าของอาเรย์ที่แทนแผนที่จุดภาพนั้น จะยังไม่เปลี่ยนภาพให้เห็นในวินโดว์ เนื่องจากอาเรย์ที่ได้มา ไม่ใช่เนื้อที่ที่ใช้แสดงภาพในวินโดว์ หากต้องการแสดงภาพใหม่ด้วยแผนที่จุดภาพ **p** ต้องเรียกใช้บริการ **setPixmap** ดังตัวอย่างข้างล่างนี้ (ทำอะไร ? ลองศึกษาดู)

```
String url = "http://www.eng.chula.ac.th/files/u1/Building2.jpg";
DWindow w = new DWindow();
w.loadImage( url );
int[][] p = w.getPixmap();
int width = p.length;
int height = p[0].length;
for (int x = 20; x < width - 20; x++) {
  for (int y = 20; y < height - 20; y++) {
    int r = DWindow.getR(p[x][y]);
    int g = DWindow.getG(p[x][y]);
    int b = DWindow.getB(p[x][y]);
    r = 255 - r; g = 255 - g; b = 255 - b;
    p[x][y] = DWindow.mixRGB(r, g, b);
  }
}
w.setPixmap(p); // นำแผนที่จุดภาพ p ออกแสดงในวินโดว์ w
```

สิ่งที่ต้องเขียน

- เมท็อด **int[][] blend(int[][] b1, int[][] b2, double a)** ในคลาส **Blend** เมท็อดนี้นำสีของแต่ละจุดภาพของ b_1 และของ b_2 ที่ตำแหน่งเดียวกันมาผสมกัน โดยนำความเข้ม a ส่วนของ b_1 มาผสมกับ $(1-a)$ ส่วนของ b_2 ด้วยสูตร $ab_1 + (1-a)b_2$ โดยที่ $0 \le a \le 1$ ภาพข้างล่างนี้ คือ ผลการสร้างภาพซ้อนของภาพแมว (b_1) กับภาพเด็ก (b_2) ด้วยค่าของ a ที่ต่างๆ กัน เมื่อ $a = 0$ ได้ภาพผลลัพธ์คือภาพเดียว กับ b_2 และเมื่อ $a = 1$ ได้ภาพผลลัพธ์คือภาพเดียวกับ b_1 อนึ่ง การนำสีของจุดภาพมารวมกันนั้น อย่านำสีรวมของสองภาพมาผสมกัน ต้องแยกองค์ประกอบสีทั้งสาม (R, G, B) ของภาพทั้งสองออกแยกผสมกัน ได้สีทั้งสามแล้วจึงนำมาผสมเป็นสีผลลัพธ์



$a = 0.00$ $a = 0.25$ $a = 0.50$ $a = 0.75$ $a = 1.00$

- เมท็อด **int[][] chromaKey(int[][] fg, int[][] bg, int c)** ในคลาส **ChromaKey** เมท็อดนี้นำภาพหน้าฉาก (**fg**) มาวางซ้อนกับภาพหลังฉาก (**bg**) เช่น ตัวอย่างที่แสดงข้างล่างนี้ ภาพหน้าฉากคือนก ส่วนภาพหลังฉากคือท้องฟ้า เมื่อรวมกันแล้ว จะได้ภาพด้านขวาสุด

รูป + สีเขียวเป็น key

รูปพื้นหลัง

รูปหลังการซ้อนภาพ

กลวิธีที่ใช้ในการซ้อนภาพนี้เรียกว่า chroma key อาศัยสีพิเศษในภาพหน้าฉาก (เรียกว่า key) เพื่อระบุว่าให้นำจุดภาพของหลังฉากมาแทนจุดภาพของหน้าฉากที่มีตำแหน่งเดียวกัน และเป็นสีพิเศษ สำหรับเมท็อด chromaKey ที่ให้เขียน รับพารามิเตอร์ `c` ที่ระบุสีพิเศษที่ว่านี้ (ในตัวอย่างข้างบนนี้ สีพิเศษนี้ก็คือ สีเขียว)

ข้อแนะนำ

ลองศึกษาการทำงานของโปรแกรมข้างล่างนี้ว่า ทำอะไร ถ้ามองไม่ออก ก็ลองป้อนแล้วสั่งทำงานโปรแกรมนี้ดู (อนึ่งแนะนำให้เขียนเมท็อด **`chromaKey`** ก่อน เพราะน่าจะง่ายกว่า)

```
import jlab.graphics.DWindow;
public class Mirror {
  public static void main(String[] args) {
    String image = "http://www.eng.chula.ac.th/files/u1/Building2.jpg";
    DWindow in = new DWindow();
    in.loadImage(image);
    DWindow out = new DWindow(in.getWidth(), in.getHeight());
    in.setLocation(10,10);                    // ย้ายวินโดว์ in มาที่ตำแหน่ง 10,10
    out.setLocation(50+in.getWidth(),10);     // ย้ายวินโดว์ out ไว้ทางขวาของ in
    int[][] b = in.getPixmap();
    int w = b.length;
    int h = b[0].length;
    //---------------------------------
    for (int x = 0; x < w / 2; x++) {
      for (int y = 0; y < h; y++) {
        b[w - x - 1][y] = b[x][y];
      }
    }
    //---------------------------------
    out.setPixmap(b);
  }
}
```

อ่านเพิ่มเติม

- "เริ่มเรียนเขียนโปรแกรม" บทที่ 7

รหัสต้นฉบับเริ่มต้น

```
import jlab.graphics.DWindow;

public class ChromaKey {

  public static void main(String[] args) {
    DWindow fg = new DWindow();
    fg.loadImage("bird.gif");
    int w = fg.getWidth();
    int h = fg.getHeight();

    DWindow bg = new DWindow();
    bg.loadImage("sky.jpg",w, h);
    DWindow out = new DWindow(w, h);

    fg.setLocation(10, 10);
    bg.setLocation(10 + fg.getWidth(), 10);
    out.setLocation(10 + 2*fg.getWidth(), 10);

    out.setPixmap(chromaKey(fg.getPixmap(), bg.getPixmap(),
                  DWindow.mixRGB(0, 255, 0));
  }
  //-----------------------------------------------------------------
  public static int[][] chromaKey(int[][] fg, int[][] bg, int c) {
    int w = fg.length;
    int h = fg[0].length;
    int[][] b = new int[w][h];
```

```
    return b;
  }
}
```

รหัสต้นฉบับเริ่มต้น

```
import jlab.graphics.DWindow;

public class Blend {
  public static void main(String[] args) {
    int w = 300, h = 300;
    DWindow pic1 = new DWindow(w, h);
    DWindow pic2 = new DWindow(w, h);
    DWindow out = new DWindow(w, h);
    pic1.loadImage("cat.jpg", w, h);
    pic2.loadImage("baby.jpg", w, h);
    pic1.setLocation(10, 10);
    pic2.setLocation(10 + w, 10);
    out.setLocation(10 + 2 * w, 10);
    while (true) {
      for (double a = 0; a <= 1; a += 0.1) {
        int[][] b = blend(pic1.getPixmap(), pic2.getPixmap(), a);
        out.setPixmap(b);
      }
      for (double a = 1; a >= 0; a -= 0.1) {
        int[][] b = blend(pic1.getPixmap(), pic2.getPixmap(), a);
        out.setPixmap(b);
      }
    }
  }
  //-----------------------------------------------------------------------
  public static int[][] blend(int[][] b1, int[][] b2, double a) {
    int w = b1.length;
    int h = b1[0].length;
    int[][] b3 = new int[w][h];
```

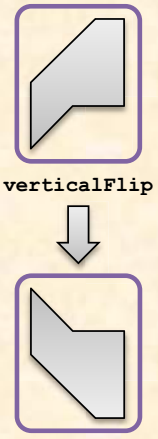
```
    return b3;
  }
}
```

การตรวจ

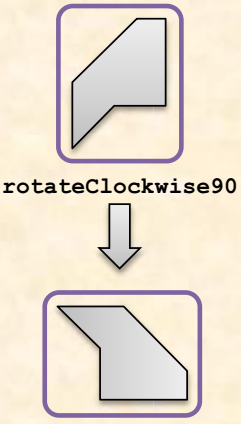
- กรณีต้องการทดสอบด้วยตนเอง : กดปุ่ม F5 เพื่อสั่งโปรแกรมทำงานเริ่มที่เมท็อด **`main`** เมท็อด **`main`** ของทั้งสองคลาส (**`Blend`** และ **`ChromaKey`**) ที่ให้มานั้น มีคำสั่งเปิดวินโดว์ อ่านภาพ ซ้อนภาพแล้วแสดงผลลัพธ์ ในคลาส **`Blend`** มีวงวนการทำงานไม่สิ้นสุด เพื่อซ้อนภาพด้วยการปรับค่า α ให้เพิ่มจาก 0 ไปเป็น 1 และลดจาก 1 กลับเป็น 0 ทำไปไม่สิ้นสุด ทำให้เห็นภาพค่อย ๆ เปลี่ยนไปมาระหว่างภาพเด็กกับภาพแมว ส่วนในคลาส **`ChromaKey`** อ่านภาพนกเป็นหน้าฉาก ท้องฟ้าเป็นหลังฉาก ซ้อนภาพ และแสดงผลลัพธ์ในอีกวินโดว์
- กรณีต้องการให้ระบบตรวจให้คะแนนอัตโนมัติ : กดปุ่ม F6 ระบบจะสั่งให้โปรแกรมตัวตรวจทำงาน โดยตัวตรวจจะสร้างอาเรย์ของจุดภาพที่มีสีสุ่ม ๆ ป้อนให้กับเมท็อดที่ต้องการทดสอบ จากนั้นอ่านผลลัพธ์กลับมาตรวจสอบว่า ถูกต้องหรือไม่ กระทำการทดสอบเช่นนี้เมท็อดละ 5 ครั้ง แล้วรายงานคะแนนที่ได้

แบบฝึกหัดเพิ่มเติม

1. จงเขียนเมท็อดประมวลผลภาพ **`int[][] verticalFlip(int[][] p)`** คืนแผนที่จุดภาพที่เป็นผลจากการพลิกภาพ **`p`** ตามแนวดิ่ง



2. จงเขียนเมท็อดประมวลผลภาพ **`int[][] rotateClockwise90(int[][] p)`** คืนแผนที่จุดภาพที่เป็นผลจากหมุนภาพ **`p`** ตามเข็มไป 90 องศา



3. จงเขียนเมท็อดประมวลผลภาพ **`int[][] brightness(int[][] p, double v)`** คืนแผนที่จุดภาพที่เป็นผลจากการคูณองค์ประกอบสีทั้งสามสีของจุดภาพทุกจุดด้วย **`v`**

ปฏิบัติการที่ 10 : เรื่องของอ็อบเจกต์

ผลการเรียนรู้

- การเขียนตัวสร้าง (constructor) ให้กับคลาส
- การสร้าง การเข้าใช้และเปลี่ยนแปลงข้อมูลภายในอ็อบเจกต์
- การสร้างอาเรย์ของอ็อบเจกต์
- การประมวลผลข้อมูลแบบอ็อบเจกต์ด้วยเมท็อดประจำคลาส

เนื้อหา

- มีแฟ้มข้อมูลชื่อ members.txt ภายในประกอบด้วย
  - บรรทัดแรกเก็บจำนวนเต็มที่ระบุจำนวนสมาชิกที่เก็บในแฟ้มนี้ คนละหนึ่งบรรทัด ตั้งแต่บรรทัดที่สองเป็นต้นไป
  - แต่ละบรรทัดประกอบด้วยข้อมูล 6 ส่วน แต่ละส่วนคั่นด้วยช่องว่าง ดังนี้
    - เลขประจำตัว
    - หมู่เลือด
    - วันที่เกิด
    - เดือนเกิด
    - ปีเกิด (ปี พ.ศ.)
    - ชื่อ สกุล (ส่วนนี้คือส่วนเหลือทั้งหมดของบรรทัด)

```
4
IO1-1029381 A 12 3 2531 ศรีสมร
AB9-1019291 O 11 4 2521 อรอนงค์
AB8-1029211 O 31 5 2521 องอาจ
XW0-1029112 AB 21 5 2521 กนก
```

- การนำข้อมูลย่อยที่มีความสัมพันธ์กันในลักษณะเช่นนี้เข้ามาประมวลผล ควรจะนำมาเก็บในรูปของอ็อบเจกต์
- สำหรับกรณีนี้ เราเขียนคลาส **Member** ที่มีรูปแบบดังนี้

```
public class Member {
  public String id;
  public String name;
  public String bloodGroup;
  public Date birthDate;
}
```

มีไว้สร้างอ็อบเจกต์เพื่อเก็บข้อมูลของสมาชิก หนึ่งอ็อบเจกต์เก็บสมาชิกหนึ่งคน
ให้สังเกตว่า **Member** มีสมาชิกชื่อ **birthDate** เป็นอ็อบเจกต์ของคลาส **Date** ซึ่งมีรูปแบบดังนี้

```
public class Date {
  public int d;
  public int m;
  public int y; // พ.ศ.
}
```

สิ่งที่ต้องเขียน

- คลาส **Member**
  - เขียนตัวสร้าง **public Member(String i, String n, String bg, Date bd)**
- คลาส **Date**
  - เขียนตัวสร้าง **public Date(int d0, int m0, int y0)**
  - เขียนเมท็อด **public static int compare(Date d1, Date d2)**
    เพื่อเปรียบวันที่ **d1** กับ **d2**
    - ถ้า **d1** มาก่อน **d2** ให้คืน **-1**
    - ถ้า **d1** คือวันเดียวกับ **d2** ให้คืน **0**
    - ถ้า **d1** มาหลัง **d2** ให้คืน **1**
- คลาส **MyMain**
  - เขียนเมท็อด **public static void sortByBirthDate(Member[] m)**
    เพื่อเรียงลำดับข้อมูลในอาเรย์ **m** ที่เก็บสมาชิก โดยเรียงตามวันเกิดสมาชิกจากอายุมากไปอายุน้อย

ข้อแนะนำ

ปกติข้อมูลที่เรานำมาประมวลผลมักเป็นกลุ่มของข้อมูลที่มีความสัมพันธ์กัน หรือมองในอีกมุมหนึ่งว่า ข้อมูลที่เรานำมาประมวลผลมักประกอบด้วยข้อมูลย่อยต่างๆ ที่มีความสัมพันธ์กัน เช่น วันที่ ประกอบด้วย วัน เดือน และปี ข้อมูลสามตัวนี้ต้องไปด้วยกันเสมอเพื่อประกอบกันเป็นวันที่ หากเราเจอสภาพของข้อมูลที่ต้องการประมวลผลในลักษณะเช่นนี้ ก็เหมาะมากที่จะนิยามลักษณะของข้อมูลนั้นด้วยคลาส นั่นคือ ใช้คลาสเพื่อนิยามข้อมูลย่อยต่างๆ ที่ประกอบกันเป็นข้อมูลที่เราสนใจ และเมื่อใดที่นิยามประเภทข้อมูลด้วยคลาส ก็ต้องเขียนตัวสร้างกำกับคลาสเพื่อให้ผู้ใช้สร้างอ็อบเจกต์ของคลาส ในปฏิบัติการนี้ เราต้องเขียนตัวสร้างของ **Member** และ **Date**

รหัสต้นฉบับของคลาส **Member**, **Date** และ **MyMain** มีตัวอย่างการใช้อ็อบเจกต์ การสร้างอ็อบเจกต์ และการสร้างอาเรย์เพื่อเก็บอ็อบเจกต์ให้ศึกษากัน อยากแนะนำให้ศึกษาดูว่าแต่ละเมท็อดทำงานอย่างไร ดังนี้

- คลาส **Date** มีเมท็อด **equals(d1,d2)** ที่รับอ็อบเจกต์ของ **Date** สองตัว **d1** และ **d2** เพื่อเปรียบเทียบว่า ข้อมูลภายในทุกตัวของ **d1** และ **d2** เหมือนกันหรือไม่
- คลาส **Member** มีเมท็อด **equals(m1,m2)** ที่รับอ็อบเจกต์ของ **Member** สองตัว **m1** และ **m2** เพื่อเปรียบเทียบว่า ข้อมูลภายในทุกตัวของ **m1** และ **m2** เหมือนกันหรือไม่ ให้สังเกตว่า มีการใช้เมท็อด **equals** ของสตริง และเมท็อด **equals** ของคลาส **Date** เพื่อเปรียบเทียบวันเกิดด้วย
- คลาส **MyMain** มีเมท็อด **readMemberFile(file)** อ่านข้อมูลสมาชิกจากแฟ้มมาสร้างอาเรย์ของ-อ็อบเจกต์สมาชิก และมีเมท็อด **showMembers(members)** ที่รับอาเรย์ของสมาชิกมาแสดง

สำหรับการเรียงลำดับสมาชิกตามอายุมากไปน้อยนั้น สามารถใช้การเรียงแบบเลือก (selection sort ดูหนังสือหน้าที่ 170) นำมาปรับให้ใช้กับอาเรย์ของอ็อบเจกต์ และใช้ **Date.compare** ในการเปรียบเทียบวันเกิด

อ่านเพิ่มเติม

- "เริ่มเรียนเขียนโปรแกรม" บทที่ 7 (การเรียงลำดับข้อมูลในอาเรย์) และบทที่ 8 (คลาสและอ็อบเจกต์)

รหัสต้นฉบับเริ่มต้น

```
public class Date {
  public int d;
  public int m;
  public int y; // พ.ศ.

  public Date(int d0, int m0, int y0) {
    // เพื่อความง่าย ไม่ต้องตรวจสอบว่า วัน เดือน ปี ที่ได้รับเป็นจำนวนที่ถูกต้องหรือไม่
```

```
  }
  //---------------------------------------------------------
  public static int compare(Date d1, Date d2) {
    // -1 เมื่อ d1 เป็นวันที่ที่มาก่อน d2
    //  0 เมื่อ d1 เป็นวันที่เดียวกับ d2
    // +1 เมื่อ d1 เป็นวันที่ที่อยู่หลัง d2
```

```
  }
  //---------------------------------------------------------
  public static boolean equals(Date d1, Date d2) {
    return d1 != null && d2 != null &&
           d1.y == d2.y && d1.m == d2.m && d1.d == d2.d;
  }
}
```

รหัสต้นฉบับเริ่มต้น

```
public class Member {
  public String id;
  public String name;
  public String bloodGroup;
  public Date birthDate;

  public Member(String i, String n, String bg, Date bd) {
    // เพื่อความง่าย ไม่ต้องตรวจสอบว่า ข้อมูลทั้งหลายที่ได้รับถูกต้องหรือไม่
```

```
  }
  //-------------------------------------------------------------
  public static boolean equals(Member m1, Member m2) {
    return m1 != null && m2 != null &&
           m1.id.equals(m2.id) &&
           m1.name.equals(m2.name) &&
           m1.bloodGroup.equals(m2.bloodGroup) &&
           m2.birthDate != null && Date.equals(m1.birthDate, m2.birthDate);
  }
}
```

รหัสต้นฉบับเริ่มต้น

```
import java.util.Scanner;
import java.io.*;

public class MyMain {
  public static void main(String[] args) throws IOException {
    Member[] members = readMemberFile("members.txt");
    sortByBirthDate(members);
    showMembers(members);
  }
  //---------------------------------------------------------------------------
  public static void sortByBirthDate(Member[] m) { //เรียงจากอายุมากไปอายุน้อย
```

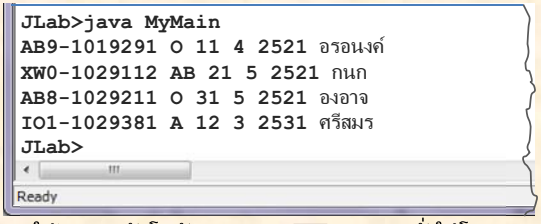
```
  }
  //---------------------------------------------------------------------------
  public static Member[] readMemberFile(String file) throws IOException {
    Scanner f = new Scanner(new File(file));
    int n = f.nextInt();
    Member[] members = new Member[n];
    for (int i = 0; i < n; i++) {
      String id = f.next();
      String bloodGroup = f.next();
      int d = f.nextInt();
      int m = f.nextInt();
      int y = f.nextInt();
      String name = f.nextLine();
      members[i] = new Member(id, name, bloodGroup, new Date(d, m, y));
    }
    f.close();
    return members;
  }
  //---------------------------------------------------------------------------
  public static void showMembers(Member[] members) {
    for (int i = 0; i < members.length; i++)
      System.out.println(members[i].id + " " +
                         members[i].bloodGroup + " " +
                         members[i].birthDate.d + " " +
                         members[i].birthDate.m + " " +
                         members[i].birthDate.y + " " +
                         members[i].name);
  }
}
```

การตรวจ

- กรณีต้องการทดสอบด้วยตนเอง : กดปุ่ม F5 เพื่อสั่งโปรแกรมทำงานเริ่มที่เมท็อด **main** โปรแกรมที่เขียนไว้ จะอ่านแฟ้ม members.txt มาสร้างอ็อบเจกต์เก็บในอาเรย์ แล้วส่งไปเรียงลำดับตามวันเกิด ถ้าแฟ้ม members.txt มีตัวอย่างข้อมูลข้างล่างนี้

```
4
IO1-1029381 A 12 3 2531 ศรีสมร
AB9-1019291 O 11 4 2521 อรอนงค์
AB8-1029211 O 31 5 2521 องอาจ
XW0-1029112 AB 21 5 2521 กนก
```

จะได้ผลลัพธ์ข้างล่างนี้ เรียงลำดับตามวันเกิด 11/4/2521, 21/5/2521, 31/5/2521 และ 12/3/2531



- กรณีต้องการให้ระบบตรวจให้คะแนนอัตโนมัติ : กดปุ่ม F6 ระบบจะสั่งให้โปรแกรมตัวตรวจทำงาน โดยตัวตรวจจะทดสอบตัวสร้างและเมท็อดดังนี้ตามลำดับ
  - ตัวสร้างของคลาส **Date**
  - เมท็อด **compare** ของคลาส **Date**
  - ตัวสร้างของคลาส **Member**
  - เมท็อด **sortByBirthDate** ของคลาส **MyMain**

  โดยต้องเขียนให้ถูกต้องตามลำดับที่แสดงข้างบนนี้ เพราะตัวสร้างหรือเมท็อดข้างล่างต้องใช้ของข้างบน

แบบฝึกหัดเพิ่มเติม

1. ตัวสร้างสำเนา (copy constructor) คือ ตัวสร้างที่รับพารามิเตอร์ที่เป็นอ็อบเจกต์ของคลาสเดียวกับตัวสร้าง มีหน้าที่ตั้งค่าเริ่มต้นของข้อมูลภายในต่างๆ ให้เหมือนกับของพารามิเตอร์ที่รับมา จงเขียนตัวสร้างสำเนาของคลาส **Date** และ **Member**

```
public class Date {
  ...
  public Date( Date d ) {









  }
  ...
```

```
public class Member {
  ...
  public Member( Member m ) {









  }
  ...
```

2. จงเขียนเมท็อด **`Member[] getMembers(Member[] m, String bloodGroup)`** ที่คืนอาเรย์ที่เก็บเฉพาะสมาชิกที่เลือกจาก **`m`** ที่มีหมู่เลือดเหมือนที่ระบุในพารามิเตอร์ **`bloodGroup`**

3. จงเขียนเมท็อด **`Member[] getMembers(Member[] m, int maxAge, Date currentDate)`** ที่คืนอาเรย์ที่เก็บเฉพาะสมาชิกที่เลือกจาก **`m`** ที่อายุไม่เกิน **`maxAge`** โดยวันนี้คือ **`currentDate`** (อายุคิดคร่าว ๆ จากเลขปีก็พอ)

ปฏิบัติการที่ 11 : ตัดเกรด

ผลการเรียนรู้

- การเขียนเมท็อดประจำอ็อบเจกต์

เนื้อหา

เพื่ออำนวยความสะดวกให้กับครูในการประเมินผลการเรียนของนักเรียน เราต้องการออกแบบคลาส **Grader** ที่ผลิตตัวตัดเกรดที่ทำตัวเสมือนถังเก็บคะแนน ให้บริการตัดเกรดตามเกณฑ์ และหาค่าสถิติต่างๆ ของคะแนน ก่อนจะดูรายละเอียดของคลาสนี้ มาดูวิธีการใช้งานกันก่อนดีกว่า

```
Grader gd = new Grader(20, 80, 70, 60, 50);
gd.add("5310012121", 89);
gd.add("5310214821", 80);
gd.add("5310312321", 71);
gd.add("5310391221", 54);
gd.add("5310442921", 63);
gd.add("5310437621", 75);
System.out.println("Average = " + gd.average());
System.out.println("Stdev   = " + gd.stdev());
gd.printSortedByID();
```

บรรทัดแรกสร้างอ็อบเจกต์ตัดเกรดด้วยตัวสร้างที่พารามิเตอร์ตัวแรกระบุจำนวนนักเรียนมากสุด (ของจริงอาจน้อยกว่าก็ได้) ตามด้วยพารามิเตอร์อีก 4 ตัวที่ระบุเกณฑ์การตัดเกรด (ขอมีแค่ 5 เกรด จากตัวอย่างคือ ถ้าได้คะแนนตั้งแต่ 80 ขึ้นไป ได้ A, ถ้าได้คะแนนตั้งแต่ 70 แต่ไม่ถึง 80 ได้ B, ถ้าได้คะแนนตั้งแต่ 60 แต่ไม่ถึง 70 ได้ C, ถ้าได้คะแนนตั้งแต่ 50 แต่ไม่ถึง 60 ได้ D และถ้าน้อยกว่า 50 ได้ F) หกบรรทัดต่อมาใส่คะแนนของนักเรียนหกคนให้กับตัวตัดเกรด ตามด้วยสองคำสั่งที่แสดงค่าเฉลี่ยและค่าเบี่ยงเบนมาตรฐาน ปิดท้ายด้วยการแสดงรายงานผลการตัดเกรดของนักเรียนทั้งหมดเรียงลำดับตามเลขประจำตัว

เนื่องจากตัวตัดเกรดมีบริการรายงานผล **printSortedByID** ดังนั้นภายในอ็อบเจกต์แบบ **Grader** จึงต้องจำข้อมูลต่าง ๆ ของนักเรียนทุกคนไว้ด้วย วิธีที่ง่ายสุดในการจำข้อมูลดังกล่าวก็คือ ใช้อาเรย์เก็บ โดยสร้างอาเรย์ที่มีขนาดเท่ากับพารามิเตอร์ตัวแรกของตัวสร้าง แต่ขนาดนี้เป็นปริมาณมากสุด ข้อมูลจริงอาจมีน้อยกว่าได้ จึงต้องมีตัวแปรอีกตัวจำจำนวนนักเรียนไว้ด้วย คลาสที่ออกแบบจึงมีลักษณะดังนี้

```
public class Grader {
  Student[] students;        // เก็บข้อมูลของนักเรียน
  int size;                  // เก็บจำนวนนักเรียน
  double a, b, c, d;         // เก็บเกณฑ์การตัดเกรด

  public Grader(int n, double a0, double b0, double c0, double d0) {
    students = new Student[n];       // สร้างอาเรย์เตรียมไว้เก็บนักเรียน
    size = 0;                        // เริ่มต้นยังไม่มีนักเรียน จำนวนนักเรียนเป็น 0
    a = a0; b = b0; c = c0; d = d0;
  }
  ...
}
```

สิ่งที่ต้องเขียน

เขียนเมท็อดประจำอ็อบเจกต์ให้กับคลาส **Grader** ดังต่อไปนี้

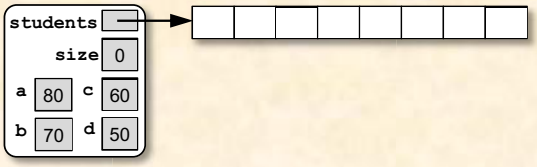
- **String grade(double point)** คืนเกรดที่คิดจากคะแนนที่ได้รับ
- **void add(String id, double point)** เพิ่มข้อมูลของนักเรียนคนใหม่พร้อมคำนวณเกรดให้
- **double average()** คืนค่าเฉลี่ยของคะแนนทั้งหมด
- **double stdev()** คืนค่าเบี่ยงเบนมาตรฐานของคะแนนทั้งหมด
- **void printSortedByID()** แสดงข้อมูลของนักเรียนทุกคนทางจอภาพ โดยแสดงเรียงตามลำดับของเลขประจำตัวจากน้อยไปมาก

ข้อแนะนำ

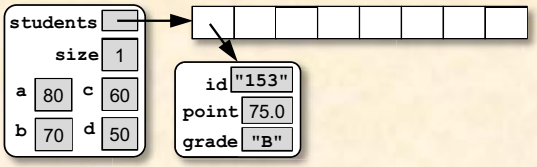
- คลาส **Grader** ผลิตอ็อบเจกต์ที่ภายในเก็บเกณฑ์การตัดเกรด มีอาเรย์ที่แต่ละช่องเก็บอ็อบเจกต์ **Student** กับตัวแปรที่จำจำนวนนักเรียน อาเรย์มีขนาดตามที่ผู้สร้างกำหนด สมมติว่าเราสร้างด้วยคำสั่งข้างล่างนี้

```
Grader g = new Grader(8,80,70,60,50);
```

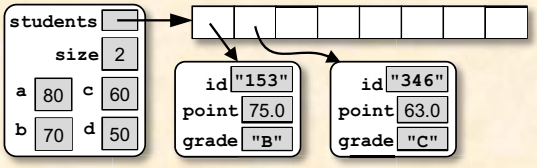
จะได้อ็อบเจกต์ดังรูป



อย่าลืมว่า การ **new Students[8]** ได้แค่อาเรย์ 8 ช่อง ไว้เก็บตัวอ้างอิงอ็อบเจกต์นักเรียน ไม่ได้สร้างอ็อบเจกต์นักเรียนใดๆ เมื่อใดที่ **add** ข้อมูล จึงจะสร้างอ็อบเจกต์นักเรียนใหม่เพิ่มในอาเรย์ เช่น หลังทำคำสั่ง **g.add("153",75)** จะได้



ให้สังเกตว่า การ **add** ครั้งแรกนี้ได้สร้างและเก็บอ็อบเจกต์นักเรียนในช่องที่ 0 ของอาเรย์ ค่าของตัวแปร **size** ถูกเพิ่มขึ้นอีก 1 ถ้าเพิ่มนักเรียนอีกคนด้วย **g.add("346",63)** คราวนี้อ็อบเจกต์ใหม่จะถูกเก็บต่อจากตัวหลังสุด โดยตำแหน่งของช่องที่เราจะเก็บก็คือ ค่าของตัวแปร **size** (ซึ่งตอนนี้เก็บ 1) คือเก็บในช่อง 1 ของอาเรย์ ดังแสดงข้างล่างนี้ และตัวแปร **size** ก็เปลี่ยนเป็น 2



อ่านเพิ่มเติม

- "เริ่มเรียนเขียนโปรแกรม" บทที่ 8

รหัสต้นฉบับเริ่มต้น

```
public class Student implements Comparable {
  String id;
  double point;
  String grade;

  public Student(String i, double p, String g) {
    id = i;
    point = p;
    grade = g;
  }
  public int compareTo(Object obj) {
    Student s = (Student) obj;
    return id.compareTo(s.id);
  }
  public String toString() {
    return "id=" + id + ", point=" + point + ", grade=" + grade;
  }
  public boolean equals(Object obj) {
    if (!(obj instanceof Student)) return false;
    Student s = (Student) obj;
    return id.equals(s.id) && point == s.point && grade.equals(s.grade);
  }
}
```

รหัสต้นฉบับเริ่มต้น

```
public class Grader {
  Student[] students;        // เก็บข้อมูลของนักเรียน
  int size;                  // เก็บจำนวนนักเรียน
  double a, b, c, d;         // เก็บเกณฑ์การตัดเกรด

  public Grader(int n, double a0, double b0, double c0, double d0) {
    students = new Student[n];  // สร้างอาเรย์เตรียมไว้เก็บนักเรียน
    size = 0;                   // เริ่มต้นยังไม่มีนักเรียน จำนวนนักเรียนเป็น 0
    a = a0; b = b0; c = c0; d = d0;
  }
  //-------------------------------------------------------
  // คืนเกรดที่ได้จากคะแนนและเกณฑ์การตัดเกรดที่ได้ตั้งไว้ตอนสร้างอ็อบเจกต์
  public String grade(double point) {









  }
```

```
  //-------------------------------------------------------
  // เพิ่มเลขประจำตัวและคะแนนของนักเรียน
  public void add(String id, double point) {
    // ตัดเกรดให้ก่อน จากนั้นสร้างอ็อบเจกต์ Student เพิ่มเก็บในอาเรย์
```

```
  }
  //-------------------------------------------------------
  // คืนค่าเฉลี่ยของคะแนน
  public double average() {
```

$$\bar{x} = \frac{1}{N}\sum_{i=1}^{N} x_i$$

```
  }
  //-------------------------------------------------------
  // คืนค่าเบี่ยงเบนมาตรฐานของคะแนน
  public double stdev() {
```

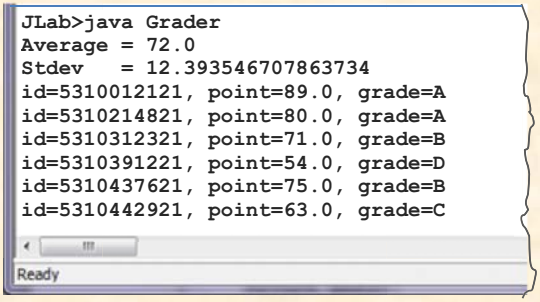
$$s = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(x_i - \bar{x})^2}$$

```
  }
```

```
  //-------------------------------------------------------
  // แสดงข้อมูลของนักเรียนทุกคนทางจอภาพ โดยแสดงเรียงตามลำดับของเลขประจำตัวจากน้อยไปมาก
  public void printSortedByID() {
    // เมื่อต้องการแสดงนักเรียน students[i] สามารถเขียน
    //   System.out.println( students[i] );
    // ได้เลย เพราะคลาส Student มีเมท็อด toString เขียนไว้เรียบร้อย
    // สำหรับการเปรียบเทียบ id นั้น จะใช้เครื่องหมาย < หรือ > ไม่ได้เพราะเราเก็บ id แบบสตริง
    // การเปรียบเทียบสตริงต้องใช้ compareTo เช่น s1.compareTo(s2) ได้ผลสามแบบ
    // ถ้าน้อยกว่า 0 แสดงว่า s1 "น้อยกว่า" s2 ถ้ามากกว่า 0 แสดงว่า s1 "มากกว่า" s2
    // แต่ถ้าเป็น 0 แสดงว่าเท่ากัน

  }
  //-------------------------------------------------------
  public static void main(String[] args) {
    Grader gd = new Grader(20, 80, 70, 60, 50);
    gd.add("5310012121", 89);
    gd.add("5310214821", 80);
    gd.add("5310312321", 71);
    gd.add("5310391221", 54);
    gd.add("5310442921", 63);
    gd.add("5310437621", 75);
    System.out.println("Average = " + gd.average());
    System.out.println("Stdev   = " + gd.stddev());
    gd.printSortedByID();
  }
}
```

การตรวจ

- กรณีต้องการทดสอบด้วยตนเอง : กดปุ่ม F5 เพื่อสั่งโปรแกรมทำงานเริ่มที่เมท็อด **main** โปรแกรมที่เขียนไว้มีเมท็อด **main** สร้างอ็อบเจกต์ของ **Grader** เพิ่มข้อมูลนักเรียน 6 คน จากนั้นแสดงค่าเฉลี่ย ค่าเบี่ยงเบนมาตรฐาน และรายละเอียดนักเรียนทุกคนเรียงตามเลขประจำตัว หากการทำงานทุกอย่างถูกต้อง การทำงานของ **main** ควรได้ผลดังแสดงข้างล่างนี้



- กรณีต้องการให้ระบบตรวจให้คะแนนอัตโนมัติ : กดปุ่ม F6 ระบบจะสั่งให้โปรแกรมตัวตรวจทำงาน โดยตัวตรวจสร้างข้อมูลของนักเรียนพร้อมคะแนนแบบสุ่ม แล้วทดสอบการทำงานของแต่ละเมท็อดที่ให้เขียน แนะนำว่า ให้เขียนเมท็อด **grade** ก่อน ตามด้วย **add** แล้วจึงเขียน **average**, **stdev** และ **printSortedByID** การทำงานของทุกๆ เมท็อดยกเว้น **add** ต้องไม่เปลี่ยนข้อมูลใด ๆ ในอ็อบเจกต์ เพราะเมท็อดทั้งสี่ล้วนเป็นเมท็อดที่นำข้อมูลในอ็อบเจกต์มาวิเคราะห์ ไม่มีการปรับเปลี่ยนข้อมูลแต่อย่างไร (ยกเว้น **add** ที่ต้องเพิ่มอ็อบเจกต์ของ **Student** เข้าเก็บในอาเรย์) ตัวตรวจจะตรวจกรณีดังกล่าวนี้ด้วย

แบบฝึกหัดเพิ่มเติม

1. จงเขียนเมท็อด **public Student get(String id)** ให้กับคลาส **Grader** เพื่อค้นและคืนอ็อบเจกต์ **Student** ที่มีเลขประจำตัวตรงกับ **id** ที่ได้รับ ในกรณีที่หา **id** ไม่พบ ให้คืนค่า **null**

2. จงเขียนเมท็อด **`public void regrade(double a, double b, double c, double d)`** ให้กับคลาส **`Grader`** เพื่อตั้งเกณฑ์การให้เกรดใหม่ (อย่าลืมให้เกรดใหม่กับนักเรียนที่เก็บไว้ด้วย)

3. ข้อด้อยของเมท็อด **`add`** ที่เขียนมาคือ จะเพิ่มได้เท่ากับขนาดของอาเรย์ที่สร้างขึ้นตอนสร้างอ็อบเจกต์ **`Grader`** หากเพิ่มเกินจากที่กำหนดไว้ จะเกิดข้อผิดพลาด วิธีแก้ปัญหานี้คือ ตรวจว่าถ้าเก็บเต็มอาเรย์แล้ว ให้สร้างอาเรย์ใหม่ที่มีขนาดใหญ่กว่า (เช่น เป็น 1.5 เท่าของของเดิม) ทำสำเนาข้อมูลอาเรย์เดิมไปเก็บในอาเรย์ใหม่ แล้วเปลี่ยนตัวแปร **`students`** ให้อ้างอิงอาเรย์ใหม่นั้น เพียงเท่านี้ เราจะเพิ่มนักเรียนคนใหม่ได้ จงเขียนเมท็อด **`add`** ใหม่ที่เพิ่มความสามารถดังกล่าว

ภาพหน้าปก

รูปต้นไม้ของหน้าปกถูกสร้างด้วยโปรแกรมข้างล่างนี้ โปรแกรมนี้ใช้หลักการทำงานแบบเรียกซ้ำ ผู้สนใจสามารถอ่านรายละเอียดการวาดรูปในลักษณะนี้ได้ในหัวข้อการวาดสาทิสรูป บทที่ 6 หนังสือ “เริ่มเรียนเขียนโปรแกรม : ฉบับวาจาจาวา”

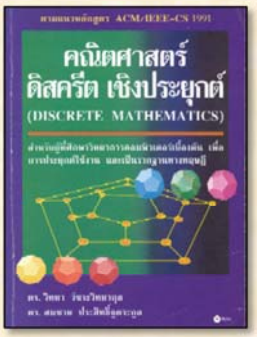
ต้นไม้

```
import jlab.graphics.*;
import java.util.Scanner;
import java.awt.*;

public class DigitalTree {
  public static void main(String[] args) {
    DWindow w = new DWindow(500, 600);
    drawTree(w, 250, 590, 150, 90, 7);
  }

  public static void drawTree(DWindow w, double x0, double y0,
                              double len, double a, int depth) {
    double x1 = x0 + len * cos(a);
    double y1 = y0 - len * sin(a);
    DRectangle r = new DRectangle(x0, y0, len, 1.2 * depth); // สร้างสี่เหลี่ยมผืนผ้าแทนกิ่ง
    r.setColor(Color.DARK_GRAY, Color.DARK_GRAY);            // ให้สีเทาเข้ม
    r.rotate(-a, x0, y0);                                    // หมุนไป a องศา
    w.draw(r);                                               // วาด
    if (depth <= 0) {
      if (Math.random() < 0.03)
        double r2 = 40 * Math.random();{                     // วาดใบเป็นวงกลม
        w.fillEllipse(Util.getRandomColor(), x1, y1, r2, r2);// ให้สีสุ่ม ๆ
      }
    } else {
      len *= 0.75;
      drawTree(w, x1, y1, len, a, depth - 1);
      double fac = 0.5 + 0.4 * Math.random();
      x1 = x0 + fac * len * cos(a);
      y1 = y0 - fac * len * sin(a);
      drawTree(w, x1, y1, fac * len, a + 80 * Math.random(), depth - 1);
      fac = 0.5 + 0.4 * Math.random();
      x1 = x0 + fac * len * cos(a);
      y1 = y0 - fac * len * sin(a);
      drawTree(w, x1, y1, fac * len, a - 80 * Math.random(), depth - 1);
    }
  }
  private static double sin(double a) {
    return Math.sin(Math.toRadians(a));
  }
  private static double cos(double a) {
    return Math.cos(Math.toRadians(a));
  }
}
```

ประวัติผู้เขียน

สมชาย ประสิทธิ์จูตระกูล จบการศึกษาปริญญาตรีสาขาวิศวกรรมคอมพิวเตอร์ (เกียรตินิยมอันดับหนึ่งเหรียญทอง) จากจุฬาลงกรณ์มหาวิทยาลัย เมื่อปี พ.ศ. ๒๕๒๖ ได้รับพระราชทานทุนมูลนิธิ “อานันทมหิดล” เพื่อศึกษาต่อในปี พ.ศ. ๒๕๒๘ จบการศึกษาปริญญาโทและปริญญาเอกสาขาวิทยาศาสตร์คอมพิวเตอร์ เมื่อปี พ.ศ. ๒๕๓๐ และ พ.ศ. ๒๕๓๔ ตามลำดับจากมหาวิทยาลัยอิลลินอยส์ ณ เมืองเออร์บานา-แชมเปน สหรัฐอเมริกา เข้ารับราชการเป็นอาจารย์ที่ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัยตั้งแต่ปี พ.ศ. ๒๕๒๗ จนถึงปัจจุบัน มีผลงานด้านตำราวิชาการดังนี้ (http://www.cp.eng.chula.ac.th/~somchai/books)

- แบบฝึกปฏิบัติการเขียนโปรแกรม ฉบับวาจาจาวา, พ.ศ. ๒๕๕๔
- เริ่มเรียนเขียนโปรแกรม : ฉบับวาจาจาวา, พ.ศ. ๒๕๕๒
- โครงสร้างข้อมูล : ฉบับวาจาจาวา, พ.ศ. ๒๕๕๐
- การออกแบบและวิเคราะห์อัลกอริทึม, พ.ศ. ๒๕๔๔
- วิยุตคณิต (ภินทนคณิตศาสตร์), พ.ศ. ๒๕๔๔
- คณิตศาสตร์ดีสครีตเชิงประยุกต์ (เขียนร่วมกับ ดร. วิทยา วัชระวิทยากุล), พ.ศ. ๒๕๓๖



การเขียนโปรแกรม
แบบฝึกปฏิบัติ : ฉบับวาจาจาวา

การเขียนโปรแกรมเป็นความสามารถที่ต้องลงมือฝึกปฏิบัติด้วยตนเอง เหมือนกับทักษะอื่น ๆ ทางวิศวกรรมที่จำเป็นต้องฝึก ๆ ๆ จึงจะเห็นผล ไม่สามารถได้มาด้วยการอ่าน ๆ ๆ

แบบฝึกปฏิบัติการเขียนโปรแกรมเล่มนี้ถูกจัดทำขึ้น เพื่อให้นิสิตได้ศึกษาเนื้อหาและเตรียมตัวก่อนเข้าเขียนโปรแกรมในห้องปฏิบัติการที่จัดขึ้นเป็นกิจกรรมเสริมการเรียนรายสัปดาห์ โดยแบบฝึกปฏิบัติการชุดต่าง ๆ มีตัวตรวจที่ใช้กับซอฟต์แวร์ JLab เพื่อตรวจสอบความถูกต้องของโปรแกรมอย่างอัตโนมัติ นิสิตจะได้ฝึกเขียน แก้ปัญหา หาที่ผิด ตรวจสอบความถูกต้องของโปรแกรมที่ได้พัฒนาขึ้น นอกจากนี้ แบบฝึกปฏิบัติการเล่มนี้ยังมีแบบฝึกหัดเพิ่มเติมให้นิสิตได้ฝึกทำโจทย์เสริมอื่น ๆ

ISBN 978-616-551-428-6