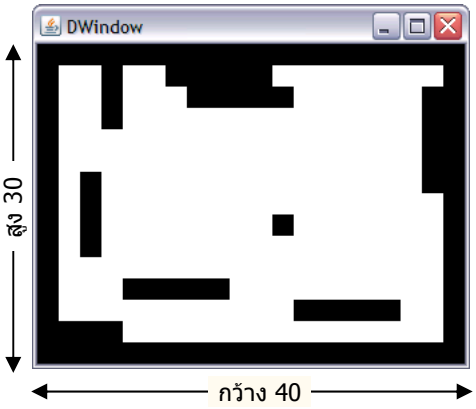
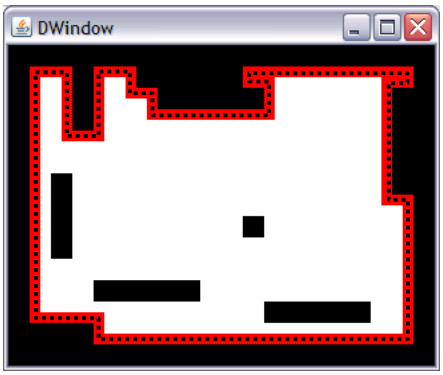
# ปฏิบัติการครั้งที่ 8 : เดินเลียบกำแพง

| | | | | | | | | 2 | 1 |
|---|---|---|---|---|---|---|---|---|---|

ปฏิบัติการครั้งนี้ให้เขียนโปรแกรมทางเดินเลียบกำแพงในห้องที่มีผนังล้อมรอบสี่ด้านและมีกำแพงขวางในห้องแบบสุ่ม ดังรูปทางขวา (สีดำแทนกำแพง สีขาวแทนที่ว่าง) เราแทนห้องนี้ด้วยอาเรย์สองมิติแบบ **int[][]** ขนาด w x h แทน กว้าง x สูงของห้อง (รูปทางขวานี้เป็นอาเรย์ **int[40][30]** แทนห้องกว้าง (แนวนอน) 40 และ สูง (แนวตั้ง) 30 สมมติว่าอาเรย์นี้ชื่อ **m** จะได้ว่า **m[0][0]** แทนตำแหน่งมุมซ้ายบน และ **m[40][30]** แทนตำแหน่งมุมขวาล่าง ให้สังเกตว่าเราใช้ **m[x][y]** อ้างอิงตำแหน่ง **(x,y)** ของห้อง (อย่าคิดว่า เมื่อนำแต่ละช่องในอาเรย์วางทาบบนห้องแล้วตำแหน่งในอาเรย์จะตรงกับของห้อง เพราะถ้าเป็นเช่นนั้น จะต้องอ้างอิงตำแหน่งที่ **(x,y)** ของห้องด้วย **m[y][x]** ของอาเรย์ ขอเน้นอีกครั้งว่า เราใช้ **m[x][y]** อ้างอิงตำแหน่ง **(x,y)** ของห้อง)



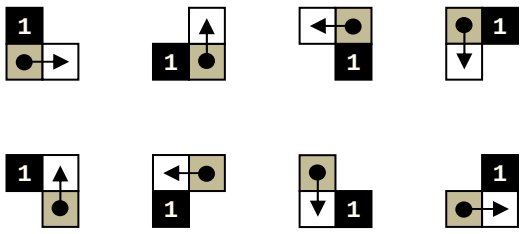

เมท็อด **createMaze** รับผิดชอบการสร้างห้องโดยประกันว่า กำแพงและช่องว่างทั้งหลายจะกว้างเป็นจำนวนคู่เสมอ (เพื่อให้การเดินเข้าซอกใด ๆ ในห้องนี้ เมื่อเดินเข้าได้ จะสามารถเดินสวนออกมาได้เสมอ โดยไม่ทับทางเดิม) ดังนั้นเส้นดำ ๆ ที่เห็นในรูปมีความหนา 2 ช่อง ข้อมูลในอาเรย์นี้มีค่าเป็น 0, 1, หรือ 2 โดยที่ 0 แทนที่ว่าง, 1 แทนกำแพง และ 2 แทนทางเดิน (ที่เราต้องเติม)




ปฏิบัติการนี้ต้องการให้หาทางเดินเริ่มจากตำแหน่ง **(2,2)** เดินเลียบกำแพงไปเรื่อย ๆ จนวกกลับมาที่ตำแหน่งตั้งต้น (เปรียบเสมือนการเริ่มต้นที่ **(2,2)** ใช้มือซ้ายแตะกำแพง หน้าหันไปด้านทิศตะวันออก แล้วเดินไปเรื่อย ๆ โดยมือซ้ายแตะกำแพงไปตลอดการเดินทาง) ดังตัวอย่างในรูปขวา แสดงทางเดินด้วยสีเทาและเส้นจุดไข่ปลา

ต้องเขียนคำสั่งการเติมทางเดินในเมท็อด **int[][] findPath(DWindow w, int[][] m)** ที่รับอาเรย์ **m** บรรยายห้อง และคืนอาเรย์ของห้องนี้ที่ได้จากการเติมทางเดินเลียบกำแพงที่ต้องการ (ด้วยค่า 2 ในอาเรย์) สิ่งที่เขียนให้แล้วในเมท็อด **findPath** คือวงวนที่ทำการเติมทางเดินหนึ่งรอบหนึ่งช่อง (เติมอย่างไร ต้องเขียนเอง) เมื่อเติมทางเดินช่องใหม่ได้แล้ว จะเรียกเมท็อด **show(w, m)** เพื่อแสดงทางเดินในวินโดว์ จะได้เห็นความก้าวหน้าของการเติมทางเดิน

ข้อแนะนำ : ช่องถัดไป ☐ ของการเดินสามารถหาได้จากช่องปัจจุบัน ● และลักษณะของช่องข้าง ๆ แบ่งได้เป็น 8 กรณีดังนี้ โดยต้องพิจารณาสี่กรณีบนก่อนสี่กรณีล่าง (รูป **1** ที่แสดงคือกำแพง)



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

public class Maze {
  public static void main(String[] args) {
    int width = 30, height = 40;
    DWindow w = new DWindow(width * 8, height * 8);
    w.setRepaintDuringSleep(true);
    int[][] maze = createMaze(width, height);
    findPath(w, maze);
    System.out.println("Done.");
  }
  //-----------------------------------------------------------------
  // เติมทางเดินเลียบกำแพงในห้องจากตำแหน่ง (xs, ys) ถึง (xt, yt)
  // ในกรณีที่ไม่มีทางเดินไปสู่จุดหมาย ให้จบเมื่อออกนอกห้อง
  public static int[][] findPath(DWindow w, int[][] m) {
    int x = 2, y = 2; // เริ่มที่ตำแหน่ง (2,2)
    m[x][y] = 2; show(w, m);
    while (!(x == 2 && y == 3)) { // เมื่อวกกลับมาที่ตำแหน่งก่อนเริ่มต้นก็จบ

     // เติมคำสั่งที่นี่ (คงใช้คำสั่งมากกว่าที่เนื้อที่ที่เว้นให้ตรงนี้)


      m[x][y] = 2; show(w, m); // แสดงภาพใหม่ทุกครั้งที่เติมทางเดิน
    }
    return m;
  }
  //-----------------------------------------------------------------
  // แสดงห้องในวินโดว์ w  กำแพงสีดำ (1) , ทางเดินสีแดง (2), ที่ว่างที่ยังไม่เดินผ่านสีขาว (0)
  private static void show(DWindow w, int[][] m) {
    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 ที่มีกำแพงกั้นสี่ด้าน และมีกำแพงสุ่ม ๆ วางกระจายในห้อง
  // กำแพงทุกแผงมีความกว้างอย่างน้อย 2 และช่องว่างทุกช่อง (ที่ให้เดินได้) มีความกว้างอย่างน้อย 2 เช่นกัน
  public static int[][] createMaze(final int width, int height) {
    // ก่อนอื่นขอสร้างห้องขนาดเป็นครึ่งหนึ่งของที่ต้องการ แล้วค่อยขยายทีหลัง
    // เพื่อให้ทางเดินมีขนาดสองช่องเสมอ จะได้เดินไปและกลับได้
    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; // เติมกำแพงด้านซ้ายและขวา
    for (int x = 0; x < w2; x++) m[x][0] = m[x][h2 - 1] = 1; // เติมกำแพงด้านบนและล่าง
    int t = Math.max(width, height) / 4; // จำนวนการเติมกำแพงในห้อง
    for (int k = 0; k < t; k++) {
      int n = (int) (8 * Math.random());             // ความยาวกำแพง
      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;  // เติมกำแพงแนวตั้ง
      // 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;  // เติมกำแพงแนวนอน
    }
    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;
  }
}
```