Programming Methodology

Testing

เนื้อหา

- Assertion
- Testing คืออะไร
- Test-Driven Development (TDD)
- JUnit
- JUnit Testing in Eclipse

assert

- assertion – เงื่อนไขที่ต้องเป็นจริง ณ จุดใดจุดหนึ่งของโปรแกรม
- เมื่อเงื่อนไขเป็นเท็จ (false) จะเรียกว่า assertion นั้น "fail"

```
assert Expression1 ;
assert Expression1 : Expression2 ;
```

เมื่อ "fail" จะ throw AssertionError

(1) ไม่มีรายละเอียด

(2) แสดงรายละเอียดตาม *Expression*$_2$

การใช้ assert (1)

- Internal Invariants

```
if (i % 3 == 0) {
    ...
} else if (i % 3 == 1) {
    ...
} else {
    // We know (i % 3 == 2)
    ...
}
```

```
if (i % 3 == 0) {
    ...
} else if (i % 3 == 1) {
    ...
} else {
    assert i % 3 == 2 : i;
    ...
}
```

การใช้ assert (2)

- Control-Flow Invariants
- ใช้ในจุดที่คาดว่าจะไปไม่ถึง (will not be reached)

```
public void foo() {
    for (...) {
        if (...)
            return;
    }
    // Execution should never reach this point!!!
}
```

```
public void foo() {
    for (...) {
        if (...)
            return;
    }
    assert false; // Execution should never reach this point!
}
```

การใช้ assert (3/1)

- Preconditions/postconditions
- public จะไม่ใช้ assert แต่ให้ใช้ exception (เดี๋ยวได้เรียน)

```
/**
* Sets the refresh rate. *
* @param rate refresh rate, in frames per second.
* @throws IllegalArgumentException if rate <= 0 or
* rate > MAX_REFRESH_RATE.
*/
public void setRefreshRate(int rate) {
    // Enforce specified precondition in public method
    if (rate <= 0 || rate > MAX_REFRESH_RATE)
        throw new IllegalArgumentException("Illegal rate: " + rate);
    setRefreshInterval(1000/rate);
}
```

การใช้ assert (3/2)

- Preconditions/postcondition
- private จะใช้ assert เพื่อตรวจสอบ

```
/**
* Sets the refresh interval (which must correspond to a
* legal frame rate).
*
* @param interval refresh interval in milliseconds.
*/
private void setRefreshInterval(int interval) {
   // Confirm adherence to precondition in nonpublic method
   assert interval > 0 &&
          interval <= 1000/MAX_REFRESH_RATE : interval;
    ...
   // Set the refresh interval
}
```

Assertion

- อย่าใช้ assert เพื่อ
  - ตรวจสอบ argument ของ public method
  - ทำงานในคำสั่งที่โปรแกรมต้องทำ
    - assert name.remove(null); // assert can be disabled.

    แก้เป็น
    - boolean nullsRemoved = name.remove(null);
    - assert nullsRemoved;

- เพราะเรา disable assertion ได้

Enabling and Disabling Assertions (1)

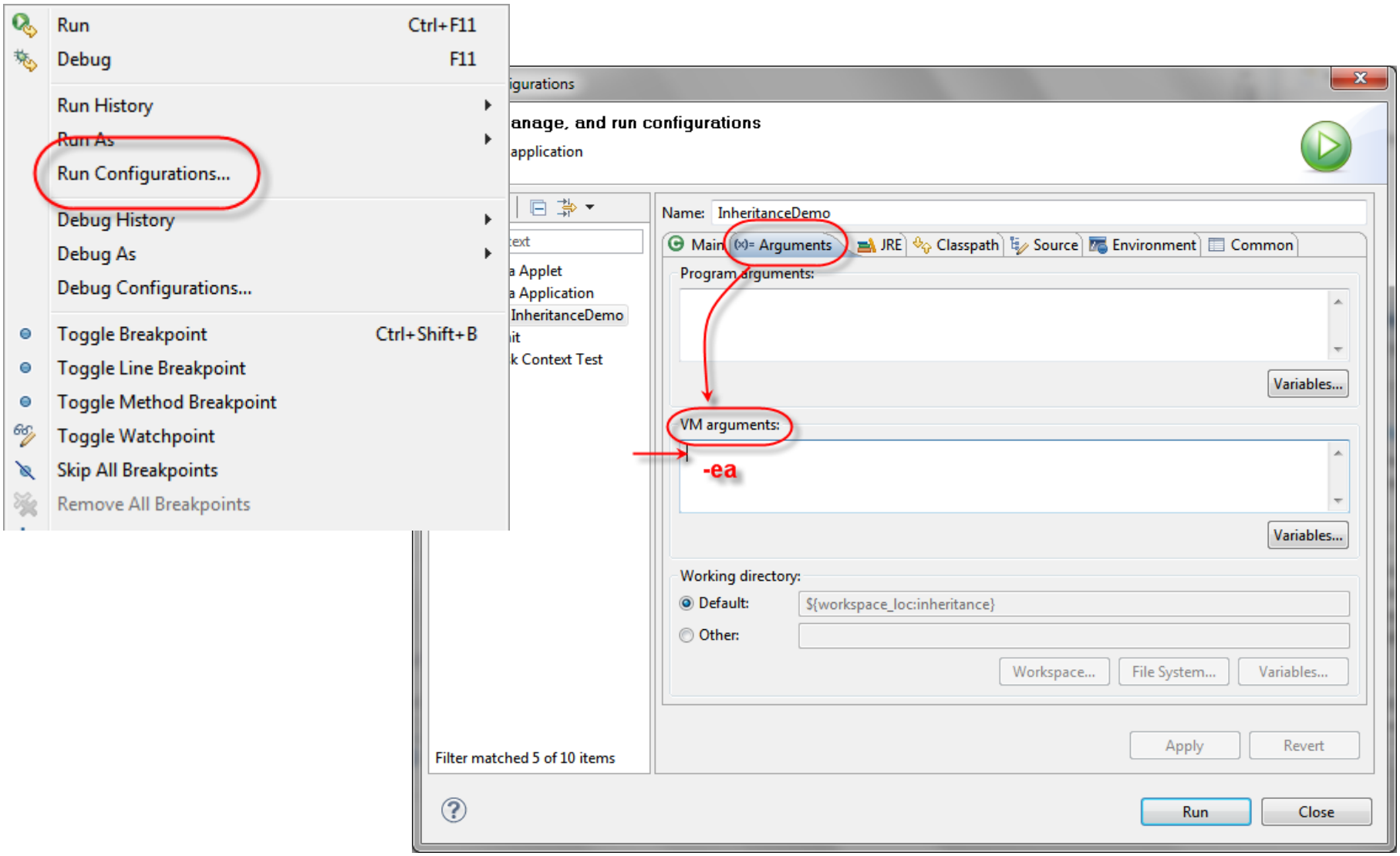
- By default, assertions are disabled at runtime.
- To enable, use the -enableassertions, or -ea, switch.
- To disable, use the -disableassertions, or -da, switch.

- no arguments - Enables or disables assertions in all classes except system classes.

Enabling and Disabling Assertions (2)

- packageName... - Enables or disables assertions in the named package and any subpackages.
- ... - Enables or disables assertions in the unnamed package in the current working directory.
- className - Enables or disables assertions in the named class
- ตัวอย่าง runs a program, *BatTutor*, with assertions enabled in only package *com.wombat.fruitbat* and its subpackages:
  - java -ea:com.wombat.fruitbat... BatTutor

Enabling Assertion in Eclipse



Testing คืออะไร?

- กระบวนการในการเปรียบเทียบระหว่าง “ผลที่ได้” กับ “สิ่งที่ควรจะเป็น”

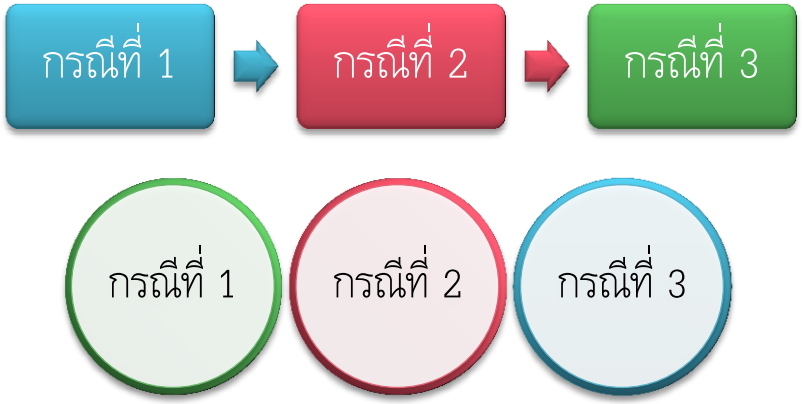
- จาก IEEE Standard 610.12-1990, "IEEE Standard Glossary of Software Engineering Terminology“
  - "The process of operating a system or component under specified conditions, observing or recording the results, and making an evaluation of some aspect of the system or component."

Testing

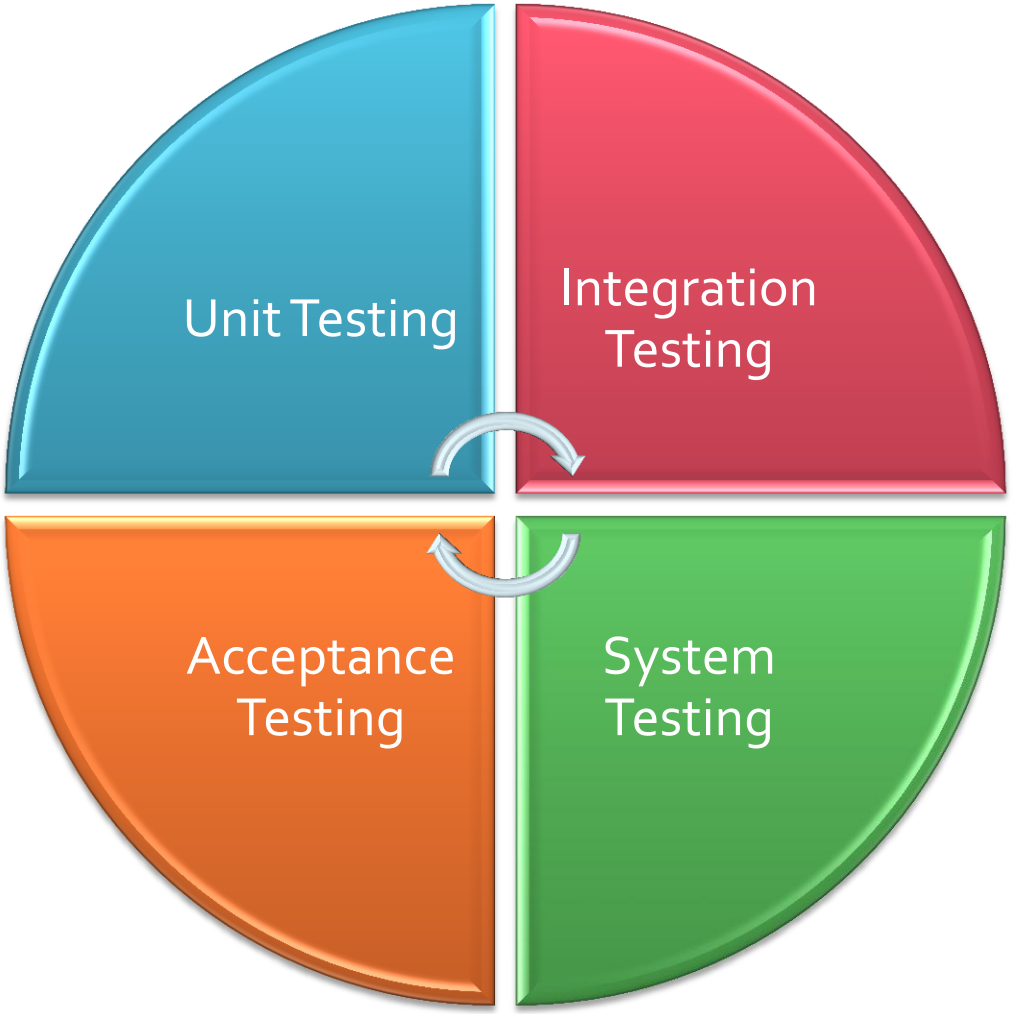
- Engineer ทำอะไรจะต้อง
  - มีการวางแผน
  - ทำการออกแบบ
  - ตามมาตรฐาน
  - ลงมือทำ และ
  - มีแผนทดสอบ
- ประโยชน์
  - รับประกันความถูกต้อง
  - เข้าใจปัญหา

Test cases (กรณีทดสอบ)

- ต้องคิดให้ดีๆ
- Inputs คืออะไร มีอะไรบ้าง
- Outputs ที่คาดหวังของ input แต่ละตัว
- Order of execution
  - cascading test cases
  - independent test cases

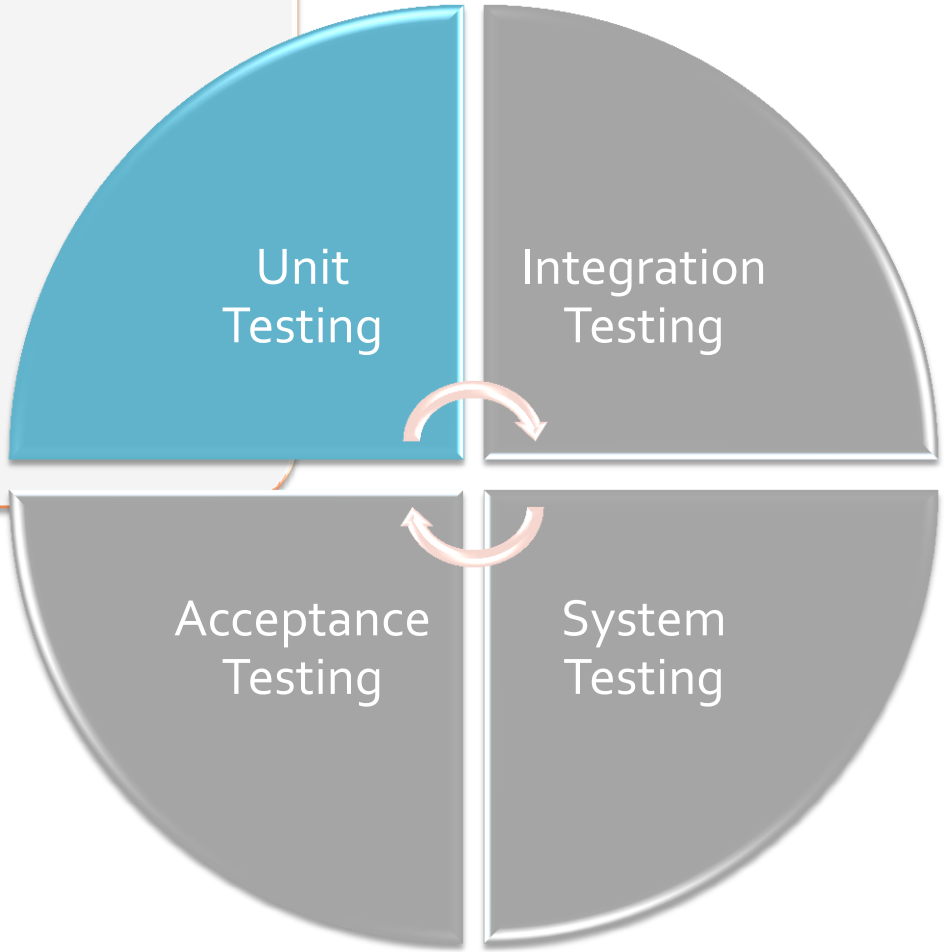


ระดับของการทดสอบ



Unit Testing

- ทดสอบหน่วยที่เล็กที่สุดของโปรแกรม
  - C++, Java → คลาส
  - C → function
- ทำไปพร้อมกับการพัฒนาโดยผู้พัฒนา
- เพื่อความมั่นใจในโปรแกรม
- ต้องผ่าน 100%



มีการทดสอบแบบไหนบ้าง?

Black box

- based on requirements and specifications
- ไม่ต้องรู้รายละเอียดภายในของโปรแกรมที่จะทดสอบ

White box

- ทดสอบ internal path, structure และ implementation
- ต้องการความสามารถทางด้าน Programming สูง

Grey box

- แอบดูนิดนึง จนพอเข้าใจการ implement แล้วก็ทำตาม black box

โปรแกรมตัดเกรด

- จะต้องทดสอบอะไรบ้าง?
- ต้องทดสอบคะแนนไหนบ้าง?

Test Cases มีกี่แบบ?

- Positive test case
  - Valid input → valid result
  - Functionality test
- Negative test case
  - Invalid input/condition
  - Robustness test
- จำนวน negative อาจจะมากกว่า positive

Identifying Test Cases

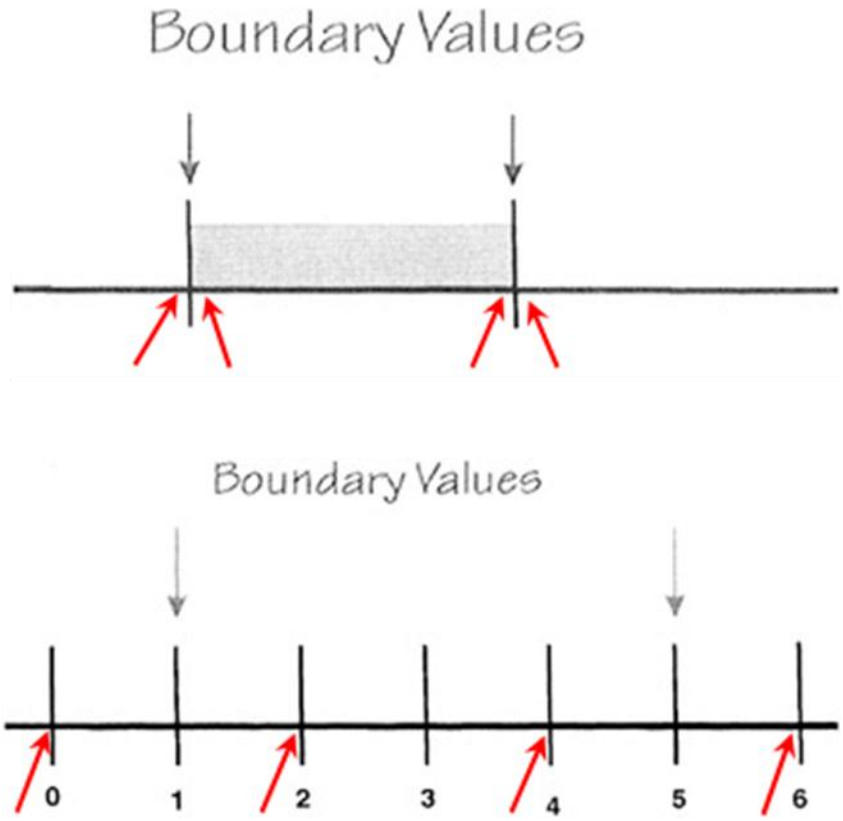
- Boundary Value Analysis
- Equivalence Partitioning/Class

- มีอีกหลายวิธี (แต่ไม่สอน) เช่น
  - logic coverage
  - all-pair testing
  - cause-effect
  - state transition
  - ฯลฯ

- ทดสอบทั้งหมด ทำไม่ไหว
- เปลืองแรง เวลา เงิน และทรัพยากรอื่นๆ

Boundary Value Analysis

- boundary condition
  - อยู่ตรงขอบเขตพอดี
  - อยู่สูงกว่าขอบเขต 1 จุด
  - อยู่ต่ำกว่าขอบเขต 1 จุด
- ของ input และ output ของ equivalence class



แบบฝึกหัด ๑

- ปรับจุดแบ่ง equivalence class
- หา boundary value ของทั้งหมด

-1, 0, 1, 58, 59, 60,
~~59~~, ~~60~~, 61, 68, 69, 70,
~~69~~, ~~70~~, 71, 78, 79, 80,
~~79~~, ~~80~~, 81, 88, 89, 90,
~~89~~, ~~90~~, 91, 99, 100, 101

Equivalence Partitioning (1)

- ลดจำนวนกรณีทดสอบลง โดยยังครอบคลุมทั้งหมด
  - แบ่งเป็น partition/class หรือ กลุ่ม
  - แต่ละกลุ่มทดสอบสิ่งเดียวกัน
  - ถ้ามีกรณีทดสอบไหนเจอ bug กรณีอื่นในกลุ่มเดียวกันก็จะเจอเหมือนกัน
  - ถ้าไม่เจอ กรณีอื่นก็จะไม่เจอ

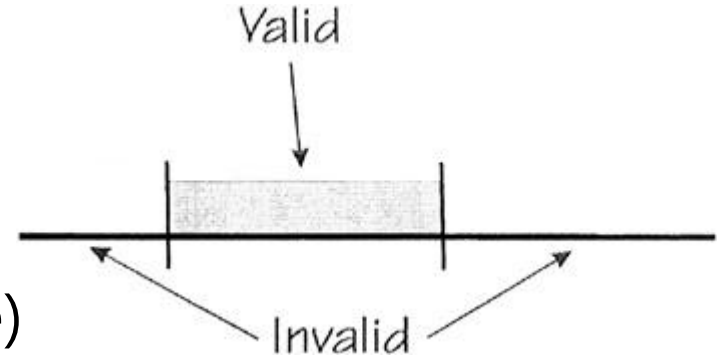
Equivalence Partitioning (2)

- จะทดสอบกรณีไหนบ้าง
  - เฉพาะ valid input หรือ
  - ทั้ง valid และ invalid input
- Design by contract
  - preconditions/postconditions
  - ทดสอบเฉพาะ valid input
  - ต.ย. มี 5 ส่วน
- Defensive design
  - ทดสอบหมด
  - ต.ย. มี 5 + 1 ส่วน

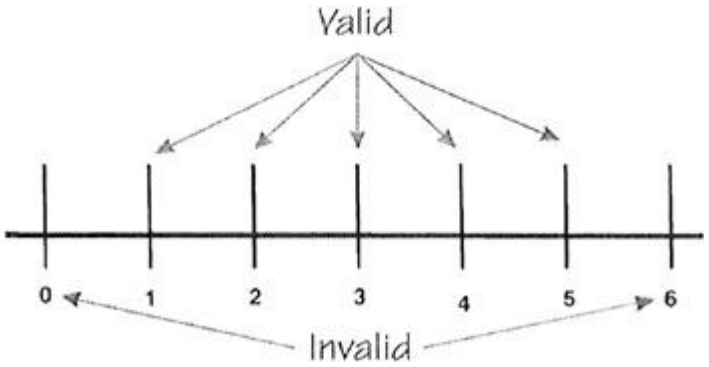
invalid input
- อาจจะแบ่งเป็น equivalence partition ย่อยๆ ได้อีก

การหา equivalence partition (1)

- แตกต่างกันตามชนิดของ input (สำหรับ defensive design)
- continuous range
  - 1 valid + 2 invalids (below/above)



- discrete values within a range
  - 1 valid + 2 invalids



การหา equivalence partition (2)

- แตกต่างกันตามชนิดของ input (สำหรับ defensive design)
- set of input value
  - 1 valid + 1 invalid
- “must be” situation
  - e.g. “first character must be a letter”
  - 1 valid + 1 invalid

แบบฝึกหัด ๒

- หา equivalence partition/class ของ
  - combineRank(Card a) เป็นการรวมค่า rank เข้ากับไพ่อีกใบ โดยให้มีค่ารวมไม่เกิน 21 (เมธอดนี้รีเทิร์น int) ตัว rank ที่เป็นอักษร ถือว่ามีค่าเท่ากับสิบ ยกเว้น A ที่มีค่าเท่ากับ 11 ก็ได้ หรือ 1 ก็ได้
    - ตัวอย่างเช่น แต่ละใบเป็นตัวเลข (this เป็นตัวเลข และ a เป็นตัวเลข)
    - ลองหา equivalence class ที่เหลือ

JUnit Testing Framework

- เป็นเครื่องมือที่ใช้ช่วยในการทำ unit test สำหรับภาษา Java
- open source (www.junit.org)
- Framework มาตรฐานสำหรับการทำ unit test สำหรับภาษา Java
- ทำการทดสอบแบบ black box

ใช้ JUnit อย่างไร? (1)

- สร้างคลาสที่จะเป็นตัวทดสอบ
- ภายในคลาสจะประกอบด้วยเมท็อดต่างๆ มากมาย
- แต่ละเมท็อดก็จะทำการทดสอบ 1 กรณีตามที่ได้หาจาก boundary value หรือ equivalent class
- ตั้งชื่อให้สื่อความหมาย
- เติม @Test หน้าชื่อเมท็อด แต่ต้องมี import org.junit.Test ที่ต้นไฟล์

- ถ้าเป็น cascading test ก็มี test method ที่เรียก test ย่อยตามลำดับ
- ถ้าเป็น independent test ก็เขียน 1 test method สำหรับ 1 test case

ใช้ JUnit อย่างไร? (2)

- เมื่อต้องการทดสอบค่าต่างๆ ก็จะใช้ เมท็อด assertXXXX(...) ทำการทดสอบ
- เติม import org.junit.Assert.*; ที่ต้นไฟล์ จะได้ไม่ต้องพิมพ์ชื่อยาว
- แต่ละ assert จะมีเงื่อนไขที่จะเป็นจริงแตกต่างกันออกไป
- ถ้าไม่ผ่าน จะเกิด AssertionError

```
assertEquals   assertFalse    assertNotNull  assertNotSame

assertNull     assertSame     assertTrue     fail
```

ใช้ JUnit อย่างไร? (3)

Assertion	ผ่านเมื่อ
assertTrue	พารามิเตอร์เป็น true
assertFalse	พารามิเตอร์เป็น false
assertNull	พารามิเตอร์เป็น null
assertNotNull	พารามิเตอร์ไม่เป็น null
assertEquals	พารามิเตอร์ทั้งคู่เท่ากัน ตามการเรียกใช้ equals
assertSame	พารามิเตอร์ทั้งคู่คือ object เดียวกัน
assertNotSame	พารามิเตอร์ทั้งคู่เป็นคนละ object
fail	Fail เสมอ

ใช้ JUnit อย่างไร? (4)

- เมื่อต้องการทดสอบว่ามี exception ที่คาดหวังเกิดขึ้นหรือไม่
- ตอนเพิ่ม @Test ให้เพิ่มรายละเอียดดังนี้
- @Test (expected=YourExpectedException.class)
- ตอนนี้ยังไม่อธิบาย เอาไว้ตอนเรียน exception

Java Unit Testing in Eclipse

- Setup JUnit โดยการ add JUnit library ใน Java Build Path ของโปรเจ็ค
- Demo คลาส Grade.java
- Card.java
  - combineRank(Card a)

References

- http://www.ibm.com/developerworks/java/library/j-junit4/index.html [Access July 5, 2011].
- Johannes Link and Peter Frohlick, Unit Testing in Java: How tests Drive the Code, Morgan Kaufmann Publishers, 2003.
- Russell Gold, Thomas Hammell and Tom Snyder, Test Driven Development: A J2EE Example, Apress, 2005.
- Lee Copeland, A Practitioner's Guide to Software Test Design, Artech House, 2004.
- JavaTM 2 SDK, Standard Edition Documentation Version 1.4.2, Programming With Assertions, Available: http://java.sun.com/j2se/1.4.2/docs/guide/lang/assert.html [Access July 23, 2008].
- http://pmd.sourceforge.net/rules/junit.html#UseAssertEqualsInsteadOfAssertTrue [Access July 5, 2011].
- http://junit.sourceforge.net/doc/cookbook/cookbook.htm [Access July 5, 2011].