UCT Department of Computer Science

Computer Science 1015F

Testing and Debugging

Hussein Suleman
<hussein@cs.uct.ac.za>
March 2009

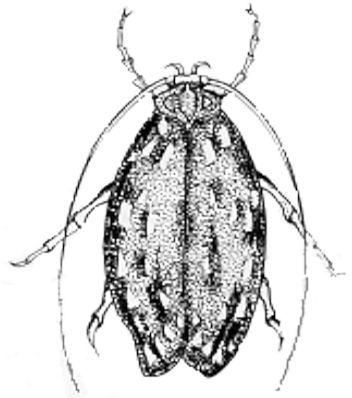
Errors and testing

- Quick Poll

- In a typical hour spent programming, how many minutes do you spend fixing errors?

Errors

- What is an error?
  - When your program does not behave as intended or expected.
- What is a bug?
  - "…a bug crept into my program …"
- Debugging
  - the art of removing bugs



Types of Errors

- Compile-time Error
  - Discovered by Java when you hit “compile”.
  - Improper use of Java language.
  - e.g., `int x + 1;`
- Run-time Error
  - Program compiles but does not execute as expected.
  - e.g., `int x=0, y = 15/x;`

Types of Errors II

- Logic Error
  - Program compiles and runs but produces incorrect results - because of a flaw in the algorithm or implementation of algorithm.

```
int a = Keyboard.readInt();
int b = Keyboard.readInt();
int maximum;
if (a < b) { maximum = a; }
      else { maximum = b; }
```


Testing Methods

- Programs must be thoroughly tested for all possible input/output values to make sure the programs behave correctly.
- But how do we test for all values of integers?

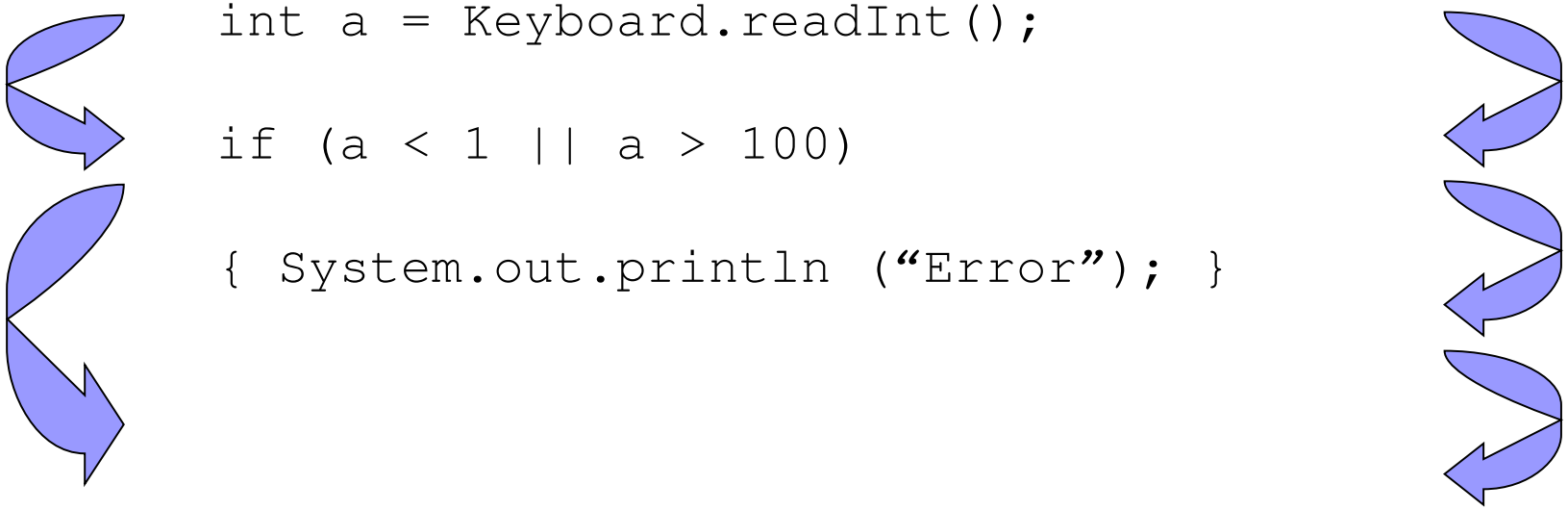
```
int a = Keyboard.readInt();
if (a < 1 || a > 100)
{ System.out.println (“Error”); }
```


Equivalence Classes

- Group input values into sets with similar expected behaviour and choose candidate values
  - e.g., -50, 50, 150
- Choose values at and on either side of boundaries (*boundary value analysis*)
  - e.g., 0, 1, 2, 99, 100, 101

Path Testing

- Create test cases to test every path of execution of the program at least once.



```
int a = Keyboard.readInt();

if (a < 1 || a > 100)

{ System.out.println (“Error”); }
```

Path 1: a=35

Path 2: a=-5

Statement Coverage

- What if we had:

```
if (a < 25)
{ System.out.println (“Error in a”); }
else
{ System.out.println (“No error in a”); }
if (b < 25)
{ System.out.println (“Error in b”); }
else
{ System.out.println (“No error in b”); }
```

- Rather than test all paths, test all statements at least once.
  - e.g., (a,b) = (10, 10), (50, 50)

Glass and Black Boxes

- If you can create your test cases based on only the problem specification, this is *black box testing*.
- If you have to look at the code, this is *glass box testing*.
- Which categories do these fall in:
  - Equivalence classes/boundary values
  - Path coverage
  - Statement coverage

Test Cases Example

```
// Software Testing sample
// hussein suleman
// CSC1015F
// 19 March 2007

import java.util.Scanner;

class SoftwareTesting
{
   public static void main ( String [] args )
    {
       // get a number
       Scanner scan = new Scanner (System.in);
       int input = scan.nextInt();
        input += 10;

       // check if it is small or large
       if (input < 20)
          System.out.print ("small number");
       else
          System.out.print ("large number");

       // check if it is divisible by 5
       if (input % 5 == 0)
          System.out.println (" divisible by 5");
    }
}
```

equivalence classes:

small multiple of 5: 5
small non-multiples of 5: 3
large multiple of 5: 25
large non-multiples of 5: 23

boundary values:

4, 5, 6, 9, 10, 11, 14, 15, 16

statement coverage:

5, 14

path coverage:

5, 7, 13, 20

Quick Poll

- So, which of these is the best approach to determine test values?

1. Exhaustive testing of all values
2. Equivalence classes and boundary values
3. Path testing
4. Statement coverage

Debugging

- **Debugging** is the process of finding **errors** or **bugs** in the code.
- A **debugger** is a tool for executing an application where the programmer can carefully control execution and inspect data.
- Features include:
  - step through code one instruction at a time
  - viewing variables
  - insert and remove breakpoints to pause execution

Assertions

- In Java a programmer can specify conditions that must always be satisfied at particular points (**invariants**) or the program produces an error. This is an **assertion**.

- Example:

```
assert (input > 0);
```


Tracing

- Insert temporary statements into code to output values during calculation.
- Very useful when there is no debugger!

- Example:

```
int x = y*y*2;
int z = x+5;

System.out.println (z);
if (z == 13)
{
   …
}
```