

CS1632, Lecture 6



Wonsun ahn

Software tends not to break much on the “happy path”

- Happy Path: a case where user inputs valid, usual values; expected usage
- It breaks on the unexpected cases.
 - Corner cases.
 - Systems going down.
 - Malicious users.
 - When you're off in the wilderness.

Logic Errors:

The logic of the program is incorrect

- Requirement: Driving while drunk shall get a student a DUI.
- Code:

```
if (student.isDrunk() || student.isDriving()) {  
    student.setDUI(true);  
} else {  
    student.setDUI(false);  
}
```

- ☛ Test multiple interior and boundary values
- ☛ When multiple parameters, test different combinations

OFF-by-one Errors: A subset of logic errors where values are specified incorrectly by one unit

- Requirement: The minimum drinking age for student shall be 21.
- Code:

```
if (student.getAge() > 21) {  
    student.setCanDrink(true);  
} else {  
    student.setCanDrink(false);  
}
```

- ☛ Pay special attention to boundary values

Null pointer error:

The program dereferences a null pointer.

- Code:

```
String cs1632 = null;  
cs1632 = cs1632.toUpperCase();
```

```
System.out.printf("Welcome to " + cs1632);
```

- ☛ When whitebox-testing, test cases where objects are not instantiated
- ☛ When blackbox-testing, think of scenarios where above may be true
 - Missing database or network connection
 - Missing textbox entry in a form

Rounding / Floating Point Errors:

Errors due to Floating Point Imprecision

```
double oneVal = 1.0 / 857.0;  
double total = oneVal * 857.0;
```

```
System.out.println("Total = " + total);
```

```
$ cd sample_code/breaking_software  
$ javac FloatingPointError.java  
$ java FloatingPointError  
Total = 0.9999999999999999
```

- Is this a serious problem? After all, it's a miniscule difference ...

Rounding / Floating Point Errors:

Errors due to Floating Point Imprecision

- Patriot Missile Defense failure of 1991
 - Failed to detect an incoming Scud missile during the Gulf War
 - Caused 28 deaths and around 100 injuries
- Why? Due to accumulated floating point precision error
 - Missile trajectory calculation involves millions of floating point operations
 - Precision errors tend to accumulate, significantly impacting final result
- ☛ Test long calculations where errors have a chance to accumulate

Integration Errors:

Errors at boundaries between systems/subsystems

Subsystem 1:

```
public class Spacecraft
    public void setDistance(int distanceInMiles) {
        ...
    }
}
```

Subsystem 2:

```
int startDistanceInKilometers = 14;
spacecraft.setDistance(startDistanceInKilometers);
```


Integration Errors:

Errors at boundaries between systems/subsystems

Subsystem 1:

```
OutputFile.write(TAB_DELIMITED) ;
```

Subsystem 2:

```
InputFile.read(COMMA_DELIMITED) ;
```

- ☛ Always test system after integration, even if subsystems are tested

Missing Data Errors:

Error due to missing data

```
public static void main(String[] args) {  
    System.out.println(args[3]);  
}
```

☛ If system expects some data, try testing without that data!

Bad Data Errors:

Improperly formatted or invalid data

```
Enter two numbers to divide: 7 0
```

```
Exception in thread "main"
```

```
    java.lang.ArithmeticException: / by zero
```

- ☛ If system expects certain values or certain formatting of values, try testing unexpected values or unexpected formatting!
- ☛ Later, we will learn how to randomly generate bad data when we talk about *fuzz testing*

Display Errors:

Data is correct but not displayed properly.

- Requirement: System shall display value of PI up to two decimals.
- Code:

```
double pi = Math.PI;
```

```
System.out.printf("Pi is equal to %.1f", pi);
```

- ☛ If data is required to be displayed in a certain way, testing value of data is not sufficient; need to also test data display
- ☛ Especially important class of errors for GUI applications

I/O Errors:

Unexpected state of disk, network, or other I/O

- Code

```
try {  
    // read in file  
} catch (FileNotFoundException e) {  
    // AAARGH WHAT DO I DO  
    System.out.println("Panic! Emergency protocol!");  
}
```

☛ If you see the above code, try running without the file 😊

Configuration error: System not configured to work correctly

- Requirement: *my.pitt.edu* shall warn user if cookies not enabled
- Expected behavior: Page displays “Website requires cookies.”
- Observed behavior:



If system expects certain software, or software with a specific version, or software configured in a specific way,

🔑 Test with both correct *and* incorrect configurations

The list goes on...

- Data type errors
 - Error arising due to incorrect implicit / explicit data type conversion
- Permission errors
 - No permission to access a required resource (file, database table, etc.)
- Version mismatch errors
 - Library version not the same version software was intended to be used with
- Distributed system errors
 - Error while communicating between different parts of distributed system
 - E.g. Error in data marshalling / demarshalling between client / server
- Interface errors
 - Error arising from developer misunderstanding behavior of an API

Now Please Read Textbook Chapter 7