Defensive Programming
Which is more important

- When writing code for a moon rover

A) Correctness
B) Robustness
Which is more important

- When writing code for a media player app (e.g., Pandora)

A) Correctness
B) Robustness
Which is more important

- When writing code for a high-frequency stock trading firm

A) Correctness
B) Robustness

Which is better?

- accepting user input
- turning off assert

A

```java
public static void main(String[] args) {
    assert args.length >= 2 && args.length <= 3 :
        "This program takes 2 or 3 arguments";
    ...
    \textit{pre-condition}
}
```

B

```java
public static void main(String[] args) {
    if (args.length < 2 || args.length > 3) {
        printUsage();
        return;
    }
    ...
}
```

C  Both are fine

D  Both are problematic
Sources of Invalid Inputs

■ Where do bad inputs come from?

- Users
  - Gracefully handle error
- Files
  - Graceful handle error
- Environmental problems
- Programming errors
- Attackers
  - Ignore the attack
  - Likelyhood error
  - Intermittent

■ How should we address them?

- Graceful error: warn, but fall back on default
- Previous
- Graceful error: warn, but fall back on default
- Defensive: radioactive particles
- Crash
- Assert
- Program instead of corrupting data
What is wrong with this code?

```java
public static ArrayList<Integer> getDataFromFile(String filename) {
    ArrayList<Integer> parsedInts;
    assert (parsedInts = new ArrayList<Integer>()) != null;
    try {
        File file = new File(filename);
        Scanner scanner = new Scanner(file);
        while (scanner.hasNext()) {
            parsedInts.add(scanner.nextInt());
        }
    } catch (FileNotFoundException e) {
        return parsedInts;
    }
    return parsedInts;
}
```

- The code contains a null pointer exception because the `assert` statement is checking the assignment of a null `ArrayList` to `parsedInts`, which is not valid. The `assert` statement should be used to check the conditions before the assignment.
- The `FileNotFoundException` is not handled properly. The code should return `parsedInts` instead of `null` to indicate that the file was not found.
- The use of `assert` is not recommended in production code as it can lead to unexpected behavior and is generally not used in production environments.
What is wrong with this code

```java
public static int Sum(int[] array){
    int sum = 0;
    try {
        for (int i = 0; true; i++) {  
            sum += array[i];
        }
    } catch (ArrayIndexOutOfBoundsException e) {
        return sum;
    }
}
```
Exceptions: Usage rules

- only for exceptional situations
- handling errors in the appropriate place
  - but, handle in the smallest scope possible
- throw exceptions of the proper abstraction
- avoid empty catch blocks
Data Structure Integrity

*E.g.,* For a doubly-linked list

```java
public class LinkedList {
    private LinkedList next;
    private LinkedList prev;

    public void checkIntegrity() {
        if (next != null) {
            assert next.prev == this :
            "prev pointer doesn't point to predecessor";
            next.checkIntegrity();
        }
    }
}
```
Barricade / Firewall

User
input

check

check

check

find errors
ASAP

trusted

untrusted

assert

assert

assert

assert

programming
error
To Dos for Tuesday

- Read Ch. 11 (Power of Variable Names)
- Read Ch. 12 (Fundamental Data Types)
- Read Ch. 13.1 and 13.2 (Unusual Data Types)
  - 13.2 is optional given that it doesn’t really relate to Java

- No new code to write.
- Just revisit the two projects that you’ve written and refactor as necessary so that they conform to what you learned in these readings.
- Be ready for code reviews next week.
  - Read rubrics that will be out later today.