CMSC 131
Object-Oriented Programming I

Math, Break, Continue

Dept of Computer Science
University of Maryland College Park

This material is based on material provided by Ben Bederson, Bonnie Dorr, Fawzi Emad, David Mount, Jan Plane

Code Duplication Avoidance

- For the current project and future projects we will be grading for comments and code duplication avoidance
- Code Duplication
  - If there is a fragment of code that appears in several sections of your program, that code should be factored out and placed in an auxiliary method
- **Example:**
  - ExperimentWithCodeDup.java
  - ExperimentNoCodeDup.java

Abstraction (Technique)

- Abstraction
  - Provide high-level model of activity or data
- Procedural abstraction
  - Specify what actions should be performed
  - Hide algorithms
- Data abstraction
  - Specify data objects for problem
  - Hide representation

Break from Loops

- break can also be used to exit immediately from any loop
  - while
  - do-while
  - for
- e.g. “Read numbers from input until negative number encountered”
  - **Example:** Break.java
  - Loop only terminates when break executed
  - This only happens when value < 0
  - Always breaks to first enclosing loop
  - Notice that break only works in loops and in the switch statement
- How about breaking from nested loops?
  - You can have a flag in both loops that controls when to stop

Warning About Breaks

- Undisciplined use of break can make loops impossible to understand
  - Termination of loops without break can be understood purely by looking while, for parts
  - When break included, arbitrary termination behavior can be introduced
- Rule of thumb: use break only when loop condition is always true
  - i.e. break is only way to terminate loop
- When you use it, make sure it has a good comment explaining what is happening

continue Statement

- **continue** can also be used to affect loops
  - while
  - do-while
  - for
  - continue jumps to bottom of loop body
- Following code prints even numbers between 0 and 10

```
for (int i = 0; i <= 10; i++){
    if (i % 2 == 1) {
        continue;
    }
    System.out.println(i);
}
```

- Effect of continue statement is to jump to bottom of loop immediately when i is odd
- This bypasses println!
- **Example:** Continue.java

continue Statement

- **continue** should be avoided
  - Confusing
  - Easy equivalents exist (e.g. if-else)
  - Included in Java mainly for historical reasons
- When you use it, make sure it has a good comment explaining what is happening

String API & Math API

- Java API
  - http://docs.oracle.com/javase/8/docs/api/
  - You should have this in your bookmarks
  - You can even download it to your computer
  - Extremely helpful
- String API
  - http://docs.oracle.com/javase/8/docs/api/java/lang/String.html
  - Implements lots of string functions
- Math API
  - http://docs.oracle.com/javase/8/docs/api/java/lang/Math.html
  - Implements lots of Math functions

String

- Keep in mind that concatenation of strings is not cheap
  - We will see an alternative called StringBuffer later on
- Let's explore some String methods through examples
  - length()
  - equalsIgnoreCase
  - compareTo
  - compareToIgnoreCase
  - charAt
  - toLowerCase → creates new object
  - toUpperCase → creates new object
  - Others …
- **Example:** StringExamples.java

Math

- Remember, part of java.lang
- Note: there is a separate package called java.math
- For java.lang.Math
  - Provides “static services”
  - We do not create a Math object (let’s try ☺)
- Let’s explore some methods
  - Math.abs
  - Math.ceil
  - Math.floor
  - Math.pow
  - Math.random → random value between 0 and less than 1
- **Example:** MathExamples.java