CMSC 131
Object-Oriented Programming I

Review, Public Tests, Comments

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

Eclipse Projects / Java Packages

- So far we have created projects using the option:
  "Use project folder as root for sources and class files"
- The second choice is:
  "Create separate folders for sources and class files"
- The second choice allow us to organize .java file and .class files separately
- We can also organize our code better by using Java Packages. So far we have used the default package. A package can be seen as folder; it helps us organize code better. For example, you can have a package for tests and a package for other classes
- **Example:** See the ClassesIntroVICode distribution
- When you create a class you create a class in a package. You will see the package directive at the top of the class

Review

- Let's go over the following topics:
  - Private/public
  - toString
  - The concept of the current object
  - equals method
  - static fields
  - static methods
  - How to use private methods to avoid code duplication

Public Tests

- Many of the class projects include public tests with the code distribution
- They are called JUnit Public Tests
  - JUnit is the framework used to write them
- A class named PublicTests.java represents the public tests for a project
  - Each method in the class represents a Test
- You don't need to know (for now) how to write them; just how to run them and read the results
- See the PublicTests.java file in the code distribution for an example and additional information

Comments

- Should make it easy for someone to look at your code and very quickly understand how it works
- Good style translates to less commenting is necessary
- Comment areas
  - Comment at the top of class → describes class purpose
  - Comment above every method → describes the contract (pre/post conditions) associated with the method and algorithms used (if any). These comments are usually very detailed
    - Pre-conditions → what must be true for the method to work
    - Post-conditions → given pre-conditions are met what will be the result of executing the method

Comments Styles

- */* block comments *\/*
  - Usually above methods or big chunks of code
- *// one line comments*
  - Usually in the middle of methods or for instance variables
- */** Javadoc *\/*
  - Notice two ** at the beginning
  - Represents comments to be processed by an utility called javadoc
  - Descriptions of classes/methods in projects usually generated using javadoc