Lecture 10: Static Methods and Variables

Last time:
1. Project #2 Hints
2. Aliasing
3. Constructors, Accessors, Mutators
4. Equality
5. Printing an object
6. for loops

Today:
1. for loops (from the previous set of slides)
2. Static variables and methods
Static Variables and Methods

- We have seen how to declare:
  - Instance variables in classes:
    ```java
    public int tokenLevel = 3;
    ```
  - Methods in classes:
    ```java
    public void setName (String nameDesired) {
      ...
    }
    ```
- Objects created from a class receive their own copies of instance variables and methods
- Java also has static variables and methods, which are shared by all objects in a class
Why Have Static Variables / Methods?

- Sometimes info needs to be shared data among all objects of a specific class type
  - e.g. How many objects in a class have been created?
  - A constant that needs to be the same for all objects of that type
- Sometimes it is useful to have methods that are in a class that can be invoked without first creating objects of that type
- Static components help for these types of things
Declaring Static Methods (and variables and constants)

- Static methods
  ```java
  public static void main (...) { ... }
  public static void drawLetter(DrawingGrid grid, char letter, Color color) { ... }
  ```

- How do we call static methods?
  ```java
  LetterMaker.drawLetter(grid, choice.charAt(0), color)
  ```

- Can have static variables and constants too
  ```java
  public static int numStudents = 0;
  public static final int MAX_ENROLLMENT = 0;
  ```

- How do we use static variables and constants? (see next example)
  ```java
  StudentRoster.numStudents
  StudentRoster.MAX_ENROLLMENT
  ```
Example: Object Counting

public class StudentRoster {
    public static int numStudents = 0;
    public static final int MAX_ENROLLMENT = 50;
    public static int getNumStudents () {
        return numStudents;
    }
}

...
What Is Printed?

- Student s1 = new Student ("John Doe");
  Student s2 = new Student ("Mary Roe");
  System.out.println (StudentRoster.getNumStudents());
  2

- Student s3 = new Student ("Eduardo Duhalde");
  System.out.println (StudentRoster.getNumStudents());
  3

- System.out.println (StudentRoster.MAX_ENROLLMENT);
  50
Class Access to Static Variables and Methods

- If `Ct` is a class, `sv` is a static variable, and `sm` is a static method ...
- Then `sv`, `sm` can be accessed via:
  - `Ct.sv`
  - `Ct.sm`
- I.e. no object of type `Ct` needs to be created at all, but if they do exist that is OK too
When To Use Static Variables?

- Class-wide constants
  ```java
  static final int MAX_ENROLLMENT = 50;
  ```

- Class-wide aggregate data
  ```java
  static int numStudents = 0;
  ```
When To Use Static Methods?

- When a method should be invocable without object creation
- When a method should not change instance variables
  - A static method can only change static variables
  - Instance variables can only be changed by non-static methods
Default Values

- Static and instance variables
  - initialized
    - most types to 0
    - char to value 0 (non-printable character)
    - strings are assigned to null
- Local variables do not have a default value and you get an error from eclipse
Calling one method from another – static and non-static

- **non-static methods**
  - when running they are associated with a specific instance
  - do have a “current object”
  - are called with: `objectName.methodName()`

- **static methods**
  - when running they are NOT associated with a specific instance
  - do NOT have a “current object”
  - are called with: `ClassName.methodName()`