Lab #6
CSE110 - Arizona State University

Topics

- Writing/Testing classes - Chapter 8

Coding Guidelines

- Give identifiers semantic meaning and make them easy to read (examples numStudents, grossPay, etc).
- Keep identifiers to a reasonably short length.
- Use upper case for constants. Use title case (first letter is upper case) for classes. Use lower case with uppercase word separators for all other identifiers (variables, methods, objects).
- Use tabs or spaces to indent code within blocks (code surrounded by braces). This includes classes, methods, and code associated with ifs, switches and loops. Be consistent with the number of spaces or tabs that you use to indent.
- Use white space to make your program more readable.
- Use comments after the ending brace of classes, methods, and blocks to identify to which block it belongs.

Assignment/Lab Documentation

At the beginning of each programming assignment you must have a comment block with the following information:

```
/*-------------------------------------------------------------------------
// AUTHOR: your name
// FILENAME: title of the source file
// SPECIFICATION: description of the program
// FOR: CSE 110- Lab #6
// TIME SPENT: how long it took you to complete the assignment
//-----------------------------------------------------------------------*/
```

References

- **(Page 362)** Writing classes
- **(Page 371)** Instance variables
- **(Page 372)** Writing methods inside the class
- **(Page 375)** Constructors
- **(Page 380)** Testing a class

Getting Started

Create a class called `Lab6` and a class called `SuperHero`. Be sure to import the `Scanner` class in `Lab6`. Be sure to name your files `Lab6.java` and `SuperHero.java`, respectively. Also, make sure the files are saved in the same directory.

Task Overview

Remember that creating by creating a class, we are defining a Java Object. These objects are often used to model various real-world objects. The overall goal of this lab is to create a class to model a superhero. This `SuperHero` class will then be tested using the `Lab6` driver class.

Part 1: Defining the Class

The first part of creating any class is to declare it as a class. The overall set up for the `SuperHero` class when the lab is done should look something like the code below. Copy it to your `SuperHero.java` file.

```
public class SuperHero {
    // instance variables go below here

    // the two constructors go below here

    // getNumberOfHeroes() goes below here

    // recordSave() goes below here

    // the second recordSave method goes here

    // killHero() goes below here

    // printSuperHeroRecord() goes below here
}
```

Part 2: Defining the Instance Variables

Instance variables are used to define different traits about an object. In the `SuperHero` class, define the following instance variables in the appropriate place:

- `numberOfHeroes`, a `static int`
- `heroName`, a `String`
- `secretIdentity`, a `String`
- `numberOfLifeChances`, an `int`
- `numberOfPeopleSaved`, an `int`

Notes & Hints

- Be sure to declare all of your instance variables as `private`, as we do not want other programs to have direct access to them.
- `numberOfHeroes` needs to be a `static` variable, because its value needs to be the same between all instances of the `SuperHero` class. You can read about `static` instance variables on page 400 or see the video example.

- As an example for the other three, the `heroName` instance variable is written below:
  `private String heroName;`

The section on instance variables starts on page 371.

Part 3: Creating the Constructors

The purpose of a constructor is to define the instance variables of the object upon its creation. By allowing for arguments to be passed into the constructor, these instance variables can be easily customized. For example, the following constructor will allow for the `heroName`, `secretIdentity`, and `numberOfPeopleSaved` to be customized:

```
public SuperHero(String initHeroName, String initSecretIdentity, int initPeopleSaved) {
    numberOfHeroes++; // one more hero created
    numberOfLifeChances = 2; // start the hero with two lives
    heroName = initHeroName;
    secretIdentity = initSecretIdentity;
    numberOfPeopleSaved = initPeopleSaved;
}
```

Include this code in the constructor section of your `SuperHero` class. Notice how `numberOfHeroes` and `numberOfLifeChances` are explicitly defined in the constructor, while the values of the other instance variables depend on what was passed into the constructor.

Now, create another constructor that will only take one parameter for the hero's name. The constructor should contain the following (you can place it below the first one):

- Only allow the `heroName` as a parameter for the constructor
- Use the same code for defining the `numberOfHeroes` and the `numberOfLifeChances`
- Set the `heroName` attribute to the parameter passed into the constructor
- Set the `secretIdentity` to `"unknown"`
- Set `numberOfPeopleSaved` to 0

When this part is completed, you will have a `SuperHero` class with two constructors. The following calls will be valid calls to one of the constructors:

```
SuperHero hero1 = new SuperHero("Superman", "Clark", 1000);
SuperHero hero2 = new SuperHero("Batman");
```

The section on constructors start on page 375.

Part 4: Creating the Other Methods

Now that we have the instance variables and constructors made, we need to define some other methods to make the class useful.

Method: getNumberOfHeroes

First, we will create an accessor method to return the number of heroes. To do this, simply fill in the body of this method:

```
public static int getNumberOfHeroes() {
    // Put something here (only 1 line is required)
}
```

Method: recordSave

Next, we will create a method to keep track of when the hero saves someone. This method will simply increment the `numberOfPeopleSaved` instance variable by 1. This method is given to you below. Include it in your lab.

```
public void recordSave() {
    numberOfPeopleSaved++;
}
```

Now, create another method called `recordSave`. This time, require an integer, called `num`, as input. Inside of the method, increment the number of people saved by this value. This allows a hero to save multiple people at once. An unfinished version of this is given below:

```
public void recordSave(<define a parameter, "int num", here>) {
    // Increment numberOfPeopleSaved by num
}
```

Method: killHero

If a hero is defeated by his/her enemy, they should lose a life. This is represented by decreasing `numberOfLifeChances` by 1. However, the hero cannot possibly die if they have no lives. So, if the hero has no lives left, display a message stating as such. The code is partially given to you below:

```
public void killHero() {
    if (<check if the hero has lives, using numberOfLifeChances>) {
        // Decrease numberOfLifeChances by 1
    } else {
        // Print a message saying the hero is dead
    }
}
```

Method: printSuperHeroRecord

This method will simply print out the different attributes of the hero. The output should look like the following:

```
Name: Superman
Secret Identity: Clark
Status: Alive
People Saved: 1000
```

Note that if the hero has no lives left, the status should be "Dead".

Hints for this method

- Use what you know about printing messages and variables - use the instance variables where appropriate inside of your print statements.
- You can use the same logic as the `killHero()` method to detect whether the hero is alive. Thus, you will need to use an `if-else` statement inside of this class.
- Since you will be printing all relevant output to the console, you do need to return anything.

The section on writing methods in classes starts on page 372.

Part 5: Create a Test SuperHero

Now that we have the `SuperHero` class created, we need to make a class to test it. This is the `Lab6.java` file, as explained in the Getting Started section. Make sure you've imported the `Scanner` class. All of the following instructions should be implemented inside of the `main` method of the `Lab6` class. Copy the following code into your `Lab6` main method. We will be modifying this code.

```
// Create a Scanner object for later use
Scanner scan = new Scanner(System.in);
// Create a superhero called Spider-Man
System.out.println("Creating Spider-Man.......");
SuperHero spiderman = new SuperHero("Spider-Man");
// Ask the user to enter a superhero name
System.out.println("\nWhat is the name of your superhero?");
String heroName = scan.nextLine(); // This is line 10
System.out.println("What is his secret identity?");
/** 13: Read in the identity */
System.out.println("Creating your super hero.......");
/** 16: Create the hero called yourHero, who saved 10 people */
System.out.println("\nSpider-Man just saved 100 lives!");
/** 19: Call recordSave on spiderman with 100 as the input */
System.out.println("Oops, Spider-Man was shot dead twice!");
/** 22: Kill spiderman twice */
System.out.print("\nYour hero saved a kidnapped kid ");
System.out.println("but was shot once");
/** 26: Kill your hero once */
/** 27: Add 1 to your hero's lives saved */
System.out.println("\n---- Superhero information ----");
/** 30: Store the number of heroes in an int called numHeroes */
System.out.println("There are " + numHeroes + " known superheroes.");
spiderman.printSuperHeroRecord();
System.out.println();
/** 33: print the record of yourHero */
```

Replace the `"/**<comment>*"` comments with the appropriate code. Each of these comments contains its line number for reference in the below hints.

Hints

- Line 13: This is very similar to what was done on line 10, just with a different variable name.
- Line 16: You will be using the constructor given to you in the lab. Use the the variables from lines 10 and 13 as the `name` and `identity` and the value 10 for the `numberOfPeopleSaved`.
- Line 19: For example, `spiderman.recordSave()` will call the `recordSave()` method for one person.
- Line 22: Simply call the `killHero()` method twice on the `spiderman` object.
- Line 26: Simply call the `killHero()` method once on your `hero` object.
- Line 27: Simply call the `recordSave()` method once on your `hero` object.
- Line 30: You will need to use the `getNumberOfHeroes()` method with the class name, like:

  `SuperHero.getNumberOfHeroes()`

  and store the returned value in the `numHeroes` variable you have to create.

- Line 33: Call the appropriate method of your `hero` object, similar to as done above for the `spiderman` object.

The section on testing classes starts on page 380.

Sample Output

Below is an example of what your output should roughly look like when this lab is completed. All text in bold represents user input.

Sample Run:
What is your super hero's name?
Superman
What is his secret identity?
Clark
Creating your super hero.....

Spider-Man just saved 100 lives!
Oops, Spider-Man was shot dead twice!
Your hero saved a kidnapped kid but was shot once.

—- Superhero information —-
There are 2 known superheroes.
Name: Spider-Man
Secret Identity: unknown
Status: Dead
People Saved: 100

Name: Superman
Secret Identity: Clark
Status: Alive
People Saved: 11

Submission

Submit your `Lab6.java` and `SuperHero.java` to the Submission Server. Go to the Submission Server site located on the course website, login, then click on Lab Submissions in the left frame. Choose Lab6 from the dropdown box, click on the browse button and find where you saved your `Lab6.java` and `SuperHero.java` on your computer. Upload the files to the site and then click on the Submit button.

Your file will be submitted and a screen will show up displaying if your program compiled and what your output is when run on some sample input.

You should then check to make sure that the actual file submitted properly and is readable to the grader. To do so click on Grades in the frame on the left of the page and then click on the 0 underneath Lab6. You will again see that your program compiled and the sample output, but you should scroll down to the bottom of the screen and make sure your file is readable as well.

Important Note: You may resubmit as many times as you like until the deadline, but we will only mark your last submission.