Lab 2

Implementing Encapsulation and Constructors

Objectives

Upon completion of this lab, you should be able to:

- Complete review questions
- Use encapsulation in a Java technology class
- Implement constructors in a class

Lab Overview

In this lab, you complete the review questions and three exercises.

- In the first exercise, you implement encapsulation in a Java technology class.
- In the second exercise, you access encapsulated attributes of a class.
- In the third exercise, you implement constructors in a Java technology class.

Completing Review Questions

Complete the following questions:

1. Which of the following statements defines an interface
   a. The way a class completes its tasks within a method
   b. The operations and attributes of an object
   c. The way other objects interact with an object
   d. The declaration of private attributes
2. State whether the following statements are true or false:
   a. The private members of an object can be accessed through public methods
   b. Constructors cannot be overloaded
3. The scope of a variable refers to:
   a. The lifetime of the variable
   b. Where a variable can be used within a program, also known as, the extent of a variable.
   c. The way other objects interact with an object
   d. Whether the variable is `private` or `public`
4. What is the default constructor for the following class?

```
public class Penny {
   String name = "lane";
}
```

   a. `Penny()`
   b. `public Penny()`
   c. `class()`
   d. `String()`

Exercise 1: Writing Encapsulated Classes

The objective of this exercise is to write a class that uses encapsulation to hide data from other objects.

In this exercise, you create a class called `DateOne` and modify its attributes to implement encapsulation and examine the output.

This exercise consists of two tasks:

- "Task 1 – Implementing Encapsulation in a Class"
- "Task 2 – Accessing Encapsulated Attributes of a Class"

Preparation

Ensure that the `DateOneTest.java`, `DateTwoTest.java`, and `DateThreeTest.java` files exists in the `SL110/exercises/09_encapconstr/exercise1` directory. This is your working directory.

Task 1 – Implementing Encapsulation in a Class

In this task, you create a class containing `private` attributes and try to access them in another class.

Follow these steps to write your test class:

1. Go to the working directory and open an editor.
2. Create a class called `DateOne` that contains three member variables of type `int` named: `day`, `month`, and `year`. Give `public` access to all the member variables. Save the class with the name `DateOne.java`.
3. Open the `DateOneTest.java` file provided in the working directory and write the following in its `main` method:
   a. Create and initialize an object of type `DateOne`.
   b. Assign different numeric values to the member variables of the `DateOne` instance.
   c. Display the value of the member variables of the `DateOne` instance.
4. Save and compile `DateOneTest.java` file.

5. Execute the `DateOneTest` class file and examine the output.
6. Create a new class called `DateTwo` similar to `DateOne` with three member variables and save it as `DateTwo.java`.
7. Modify the access specifier of the three member variables of `DateTwo` to `private`.
8. Open the `DateTwoTest.java` file provided in the working directory and perform the same steps as in Step 3. However, in this case create an instance of `DateTwo` class instead of `DateOne` class. The other lines of code will remain the same.
9. Save and compile `DateTwoTest.java.`
10. Examine the compilation errors and identify the reason of the compilation errors.

Task 2 – Accessing Encapsulated Attributes of a Class

In this task, you create a class with private attributes and enable them to be manipulated from another class.

Complete the following steps:

1. Create a class called `DateThree` and save it with `DateThree.java` file name.
2. `DateThree` class has the same three private member variables as `DateTwo` class.
3. Add the following methods in the `DateThree` class:
   a. `get` and `set` methods for all three member variables. For example:

```
public void setDay(int d) {
    day = d;
   }
public int getDay() {
    return day;
  }
```

b. The `setDate` method – It accepts three arguments of type `int` and assigns the value of the arguments to the `day`, `month`, and `year` member variables of the `DateThree` class, respectively. Before assigning the parameters to the member variables validate them such that the day is within 30, 31, 28, 29, and month is between 1 and 12 and year is between 1000 and 10000. For example February 31 should not be accepted. Error messages should be displayed whenever the validation fails.

Note – Use `switch case` statements and `if else` statements for performing the validations.

- The `displayDate` method – Prints the date in the following form: `Today's Date is: 10/25/2006`

4. Save and compile `DateThree.java`
5. Open the `DateThreeTest.java file provided in the working directory` and perform the following in the `main` method:
   a. Create and initialize and instance of `DateThree` class
   b. Invoke the `setDay`, `setMonth`, and `setYear` methods and pass valid values as arguments to these methods.
   c. Display the values of the `DateThree` member variables using its `getDay`, `getMonth`, and `getYear` methods. For example:
   ```
   System.out.println(" The date is " +
   dateObj.getMonth()+ " / " dateObj.getDay() +" /" +
   dateObj.getYear());
   ```
   ```
   // dateObj is an instance of DateThree class
   ```
6. Save and compile the `DateThreeTest.java` file.
7. Execute the `DateThreeTest` class and examine the output.
8. Reopen the `DateThreeTest.java` file.
9. In the `main` method, add code at the end which will do the following:
   a. Invoke the `setDate` method using the instance of `DateThree` class and pass valid values as arguments to the method.
   b. Invoke the `displayDate` method using the instance of `DateThree` class.

10. Save, compile, and execute the `DateThreeTest.java` file.
11. Examine the output and see if any error messages are displayed.
12. In the `DateThreeTest.java` file, modify the value of the arguments passed in the `setDate` method, repeat Step 10 and Step 11.

Discussion – You can discuss the possible error conditions while setting values for the month, day, and year in the `setDate` method. Ask the students to test the `DateThree` class with all of them and verify the error messages.

Exercise 2: Using Constructors

The objective of this exercise is to implement constructors in a class.

In this exercise, you create overloaded constructors in a class and use them to initialize objects of that class.

You have two tasks in this exercise:

- "Task 1 – Creating and Using Constructors"
- "Task 2 – Creating Constructors to Initialize Objects"

Preparation

Ensure that the `DateFourTest.java` and `RectangleTest.java` file exists in the `SL110/exercises/09_encapconstr/exercise2` directory. This is your working directory.

Task 1 – Creating and Using Constructors

In this task, you write a `DateFour` class with constructors and create objects of the class to use the constructors appropriately.

Follow these steps to write your class:

1. Go to your working directory and open an editor.
2. Create a class called `DateFour` similar to `DateThree` as in Exercise 1 and save the file as `DateFour.java` in the working directory.
3. Add a constructor, with no arguments, to the `DateFour.java` file that assigns the value of `month` to 1, the `day` to 1, and the `year` to 2007.
4. Rename the `setDate` method to be a constructor that accepts, verifies, and sets the `month`, `day`, and `year`.
5. Save and compile the `DateFour.java` file.
6. Open the `DateFourTest.java` file. In the `main` method create `DateFour` objects `d1` and `d2`, where `d1` is created using the no-argument constructor and `d2` is created using the three-argument constructor.

7. In your `DateFourTest` class, invoke the `display` method to display the `day`, `month`, and `year` values for `d1` and `d2`.
8. Compile the `DateFourTest.java` file, execute it, and verify the output.

   The output from the `DateFourTest` class should be similar to the following:

```
Today's Date is: 1/1/2007
Today's Date is: 3/9/1967
```

Task 2 – Creating Constructors to Initialize Objects

In this task, you create a class and use constructors to initialize objects. Follow these steps to create your class:

1. Create a `Rectangle` class and name the file as `Rectangle.java`.
2. Create two private member variables of type `int` and name them as `width` and `height`.
3. Add the following constructors in `Rectangle.java`:
   - A constructor with no arguments that prints the message "Default rectangle created: width=25, height=10" and sets the `width` to 25 and the `height` to 10.
   - A constructor that takes two integer arguments `w` and `h` and sets the `width` to `w` and the `height` to `h` only if both the `w` and `h` are greater than 0 and less than 30. An appropriate error message is printed if `w` or `h` is out of range. In addition, a message is printed that a rectangle was created with `width=w` and `height=h`.
4. Create a `getArea` method that calculates and returns the area of the rectangle. The area is calculated as `height* width`.
5. Create a `draw` method and perform the following in the method:
   a. Create a nested `for` loop to draw a rectangle using asterisks (*).
   b. The number of asterisks used to represent a rectangle should match with the height and width of the rectangle.
6. Save and compile the `Rectangle.java` file.
7. Modify the provided `RectangleTest.java` file to create `Rectangle` instances `r1` and `r2`, such that:
   - `r1` is created with the constructor with no arguments.
   - `r1` is drawn immediately after it is created.

- `r2` is created by using the constructor with arguments.
- `r2` is drawn, and the area is printed.

8. Save and compile the `RectangleTest.java` file and test your rectangle code.

   The output from the `RectangleTest` class should be similar to the following:

```
Default rectangle created: width=25, height=10
This is the first rectangle:
*************************
*************************
*************************
*************************
*************************
*************************
*************************
*************************
*************************
*************************
Rectangle created: width=20, height=7
This is the second rectangle:
********************
********************
********************
********************
********************
********************
********************
```

The area is 140.

Exercise Summary

Take a few minutes to identify what experiences, issues, or discoveries you had during the lab exercise.

- Experiences
- Interpretations
- Conclusions
- Applications