Discussion 1: Analyze the Difference Between Objects and Classes with Examples

Matthew D Bullis

National University

CSC272: Advanced Programming in Java

**Doctor Shahzad Aslam-Mir**

07/10/2025

Discussion 1: Analyze the Difference Between Objects and Classes with Examples

Classes and objects are essential to OOP, allowing programmers to easily store information about functionally similar items separately. A class can be simple or complex, but it is how it interacts with other classes and instances of itself (objects) that demonstrates two main components of OOP: encapsulation ensures that you or a user of the program does not need to know how an object stores or works with data (Horstmann, 2016, p. 376), and inheritance as a concept allows code to be used across classes (Horstmann, 2016, pp. 437-438).

**Classes**

It is important to understand classes before describing objects in detail. Java programmers do not “implement” objects at all, they simply use a class (Horstmann, 2016, pp. 376). All class instances are objects, and the attributes of a class instance are dependent on the methods assigned to the class. To create a class, define a group of public methods and the private variables that will store the information of a class instance. The user or another programmer doesn’t need to understand how a method works, just that it does (p. 376). This is known as encapsulation. To better demonstrate these concepts, I’ll make a scratcher game. The Scratcher class will handle the entirety of this simple scratcher game while the main class will be used to create instances of the class.

A computer code with text

AI-generated content may be incorrect.

Here I’ve created necessary variables for methods within the Scratcher class.

A screen shot of a computer code

AI-generated content may be incorrect.

These are a few of the accessor methods. Accessor methods return the value of instance variables. Like the previous example, instance variables store the data for the instance of a class (p. 379).

A screen shot of a computer program

AI-generated content may be incorrect.

Here are a few mutator methods within the Scratcher class. All methods declared beforehand are used within the startGame() method. The game is fully functional with a game loop!

**Objects**

The Scratcher class is a success! In order to put it to use, the main class needs an instance of the class to invoke methods on.

A screenshot of a computer screen

AI-generated content may be incorrect.

In main, a new instance of Scratcher is created. First, I print the return values of accessor methods before initiating the game loop.

A screenshot of a computer

AI-generated content may be incorrect.

As you can see, the user of the program invokes methods through an input stream using an object of the Scanner class “in” (refer to GitHub for source code). Lucky me, I won!s

References

Horstmann, C. S. (2016). Big Java Late Objects (2nd ed.). Wiley Global Education US. https://nu.vitalsource.com/books/9781119321071