codecademy

Access, Encapsulation, and Static Methods

The public and private keywords

In Java, the keywords `public` and `private` define the access of classes, instance variables, constructors, and methods.

`private` restricts access to only the class that declared the structure, while `public` allows for access from any class.

Encapsulation

Encapsulation is a technique used to keep implementation details hidden from other classes. Its aim is to create small bundles of logic.

The private Keyword

In Java, instance variables are encapsulated by using the `private` keyword. This prevents other classes from directly accessing these variables.

```
public class CheckingAccount{
  // Three private instance variables
  private String name;
  private int balance;
  private String id;
}
```

Accessor Methods

In Java, accessor methods return the value of a `private` variable. This gives other classes access to that value stored in that variable. without having direct access to the variable itself.

Accessor methods take no parameters and have a return type that matches the type of the variable they are accessing.

```
public class CheckingAccount{
  private int balance;

  //An accessor method
  public int getBalance(){
    return this.balance;
  }
}
```

Mutator Methods

In Java, mutator methods reset the value of a `private` variable. This gives other classes the ability to modify the value stored in that variable without having direct access to the variable itself.
Mutator methods take one parameter whose type matches the type of the variable it is modifying. Mutator methods usually don't return anything.

```
public class CheckingAccount{
  private int balance;

  //A mutator method
  public void setBalance(int newBalance){
    this.balance = newBalance;
  }
}
```

Local Variables

In Java, local variables can only be used within the scope that they were defined in. This scope is often defined by a set of curly brackets. Variables can't be used outside of those brackets.

```
public void exampleMethod(int
exampleVariable){
  // exampleVariable can only be used
inside these curly brackets.
}
```

The this Keyword with Variables

In Java, the `this` keyword can be used to designate the difference between instance variables and local variables. Variables with `this.` reference an instance variable.

```
public class Dog{
  public String name;

  public void speak(String name){
    // Prints the instance variable named
name
    System.out.println(this.name);

    // Prints the local variable named
name
    System.out.println(name);
  }
}
```

The this Keyword with Methods

In Java, the `this` keyword can be used to call methods when writing classes.

```
public class ExampleClass{
  public void exampleMethodOne(){
    System.out.println("Hello");
  }
```

```
  public void exampleMethodTwo(){
    //Calling a method using this.
    this.exampleMethodOne();
    System.out.println("There");
  }
}
```

Static Methods

Static methods are methods that can be called within a program without creating an object of the class.

```
// static method
public static int getTotal(int a, int b) {
  return a + b;
}

public static void main(String[] args) {
  int x = 3;
  int y = 2;
  System.out.println(getTotal(x,y)); //
Prints: 5
}
```

Calling a Static Method

Static methods can be called by appending the dot operator to a class name followed by the name of the method.

```
int largerNumber = Math.max(3, 10); //
Call static method
System.out.println(largerNumber); //
Prints: 10
```

The Math Class

The `Math` class (which is part of the java.lang package) contains a variety of static methods that can be used to perform numerical calculations.

```
System.out.println(Math.abs(-7.0)); //
Prints: 7

System.out.println(Math.pow(5, 3)); //
Prints: 125.0

System.out.println(Math.sqrt(52)); //
Prints: 7.211102550927978
```

The static Keyword

Static methods and variables are declared as static by using the `static` keyword upon declaration.

```
public class ATM{
  // Static variables
  public static int totalMoney = 0;
  public static int numATMs = 0;

  // A static method
  public static void averageMoney(){
    System.out.println(totalMoney /
numATMs);
  }
```

Static Methods and Variables

Static methods and variables are associated with the class as a whole, not objects of the class. Both are used by using the name of the class followed by the `.` operator.

```
public class ATM{
  // Static variables
  public static int totalMoney = 0;
  public static int numATMs = 0;

  // A static method
  public static void averageMoney(){
    System.out.println(totalMoney /
numATMs);
  }

  public static void main(String[] args){

    //Accessing a static variable
    System.out.println("Total number of
ATMs: " + ATM.numATMs);

    // Calling a static method
    ATM.averageMoney();
  }

}
```

Static Methods with Instance Variables

Static methods cannot access or change the values of instance variables.

```
class ATM{
// Static variables
  public static int totalMoney = 0;
  public static int numATMs = 0;
```

```
  public int money = 1;

  // A static method
  public static void averageMoney(){
    // Can not use this.money here because
a static method can't access instance
variables
  }

}
```

Methods with Static Variables

Both non-static and static methods can access or change the values of static variables.

```
class ATM{
// Static variables
  public static int totalMoney = 0;
  public static int numATMs = 0;
  public int money = 1;

 // A static method interacting with a
static variable
  public static void staticMethod(){
    totalMoney += 1;
   }

  // A non-static method interactingwith a
static variable
  public void nonStaticMethod(){
    totalMoney += 1;
  }
}
```

Static Methods and the this Keyword

Static methods do not have a `this` reference and are therefore unable to use the class's instance variables or call non-static methods.

```
public class DemoClass{

  public int demoVariable = 5;

  public void demoNonStaticMethod(){

  }
```

```
  public static void demoStaticMethod(){
    // Can't use "this.demoVariable" or
"this.demoNonStaticMethod()"
  }
}
```

Print Share