Chapter 3

Classes – Introduction to Object Oriented Programming

Java is an object-oriented programming language. Everything in Java is associated with classes and objects, along with its attributes and methods. A class is a template for objects, and an object is an instance of a class. For example, fruit class will have apple, mango, durian as its objects. Car class might have Toyota Avanza, Tesla Model S as its objects.

3.1 Try the following example

```
public class Store {
  // instance fields
  String productType;
  int inventoryCount;
  double inventoryPrice;

  // constructor method
  public Store(String product, int count, double price) {
    productType = product;
    inventoryCount = count;
    inventoryPrice = price;
  }

  // main method
  public static void main(String[] args) {
    Store lemonadeStand = new Store("lemonade", 42, .99);
    Store cookieShop = new Store("cookies", 12, 3.75);

    System.out.println("Our first shop sells " + lemonadeStand.productType
+ " at " + lemonadeStand.inventoryPrice + " per unit.");

    System.out.println("Our second shop has " + cookieShop.inventoryCount +
" units remaining.");
  }
}
```

3.2 Attributes

Class attributes are variables within a class. Attributes are called fields too. In example 3.1, productType, inventoryCount and inventoryPrice are attributes. Access attributes by using dot (.) syntax.

```
public class MyClass {
  int x;

  public static void main(String[] args) {
    MyClass myObj = new MyClass();
    myObj.x = 40; //Modifying attribute value
    System.out.println(myObj.x);
  }
}
```

If you don't want the ability to override existing values, declare the attribute as final.

```
public class MyClass {
  final int x = 10;

  public static void main(String[] args) {
    MyClass myObj = new MyClass();
    myObj.x = 25; // will generate an error
    System.out.println(myObj.x);
  }
}
```

3.3 Methods

A method is a task that an object of a class performs. The domain of this task is marked using curly braces: {, and }. Everything inside the curly braces is part of the task. This domain is called the scope of a method. We can't access variables that are declared inside a method in code that is outside the scope of that method.

```
public class Store {
  // instance fields
  String productType;

  // constructor method
  public Store(String product) {
    productType = product;
  }

  // advertise method
  public void advertise() {
    String message = "Selling " + productType + "!";
          System.out.println(message);
  }

  // main method
  public static void main(String[] args) {
    String cookie = "Cookies";
    Store cookieShop = new Store(cookie);

    cookieShop.advertise();
  }
}
```

Exercise for you to try:

```
public class Store {
  // instance fields
  String productType;

  // constructor method
  public Store(String product) {
    productType = product;
```

```
  }

  // advertise method
  public void advertise() {
    String message = "Selling " + productType + "!";
          System.out.println(message);
  }

  // main method
  public static void main(String[] args) {
    Store lemonadeStand = new Store("Lemonade");
  }
}
```

What you need to do now:

a) Add a method to the Store class called greetCustomer(), this method accepts a String parameter called customer.

b) Inside of the greetCustomer() method, add a print statement to print: "Welcome to the store, " + customer + "!"

c) Inside the main() method, call the greetCustomer() method on the lemonadeStand object. Pass in a String parameter of your choice!

That was easy!

Next, we added a method called increasePrice that takes one parameter, priceToAdd. This method calculates newPrice where newPrice = price + priceToAdd , and then reassign the price attribute to the newPrice value.

```
public class Store {
  // instance fields
  String productType;
  double price;

  // constructor method
  public Store(String product, double initialPrice) {
    productType = product;
    price = initialPrice;
  }

  // increase price method
  public void increasePrice(double priceToAdd){
    double newPrice = price + priceToAdd;
    price = newPrice;
  }

  // main method
  public static void main(String[] args) {
```

```
    Store lemonadeStand = new Store("Lemonade", 3.75);
  }
}
```

What you need to do now is: in the main() method, increase the price at the lemonade stand by 1.5. Then, print the lemonadeStand.price to see how it has changed. Very easy!

Since variables can only exist in the scope that they were declared in, we can use a value outside of the method it was created in only if we return it from the method. We return a value by using the keyword return.

Try the following codes:

```
public class Store {
  // instance fields
  String productType;
  double price;

  // constructor method
  public Store(String product, double initialPrice) {
    productType = product;
    price = initialPrice;
  }

  // increase price method
  public void increasePrice(double priceToAdd){
    double newPrice = price + priceToAdd;
    price = newPrice;
  }

  // get price with tax method
  public double getPriceWithTax(){
    double totalPrice = price + price * 0.08;
    return totalPrice;
  }
  // main method
  public static void main(String[] args) {
    Store lemonadeStand = new Store("Lemonade", 3.75);
  double lemonadePrice = lemonadeStand.getPriceWithTax();

    System.out.println(lemonadePrice);
  }
}
```