Lab 3

Contents:

Classes, Fields, Methods, Constructors - Default Constructor, Overloaded Constructor, "this" Keyword, Public vs Private Fields

Background:

Classes

A class is a blueprint for creating objects. It defines the properties and methods that an object of that class will have. In Java, a class is defined using the "class" keyword followed by the name of the class. Let's create a practical example of a product class definition in Java. A product class has a few properties like product name, price, and quantity:

```
public class Product {
    private String name;
    private double price;
    private int quantity;

    public Product(String name, double price, int quantity) {
        this.name = name;
        this.price = price;
        this.quantity = quantity;
    }

    public String getName() {
        return name;
    }

    public double getPrice() {
        return price;
    }

    public int getQuantity() {
        return quantity;
    }

    public void setName(String name) {
        this.name = name;
    }

    public void setPrice(double price) {
        this.price = price;
    }

    public void setQuantity(int quantity) {
```

```
        this.quantity = quantity;
    }
}
```

In this example, we have defined a class called "Product". It has three properties: name, price, and quantity. We have also defined a constructor that takes these properties as parameters and sets the values accordingly.

We have also defined getter and setter methods for each property. The getter methods allow us to access the value of the property from outside the class, while the setter methods allow us to modify the value of the property from outside the class.

It is important to note that we have marked the properties as "private". This means that they can only be accessed from within the class itself. This is known as encapsulation, which is a fundamental concept in OOP.

Encapsulation helps to protect the data within an object from external interference. In our example, we have made sure that the properties can only be accessed and modified through the getter and setter methods, which means that we have control over how the data is accessed and modified. We can also mark properties as "public", which means that they can be accessed from outside the class. However, it is generally considered good practice to use encapsulation and keep the properties as "private" in order to maintain control over the data.

Methods

Methods are functions that can be called on an object of a class to perform specific tasks. In Java, methods are defined within a class using the "public" keyword, followed by the return type (if any), and then the name of the method. Methods can be "private" too, to support functionality that need not be expressed outside of it. Let's add a method to our Product class to calculate the total value of the product:

```
public double getTotalValue() {
    return price * quantity;
}
```

In this example, we have defined a method called "getTotalValue" that returns the total value of the product (price multiplied by quantity). We can call this method on an object of the Product class to calculate the total value of that product.

Constructors

Constructors are special methods that are called when an object of a class is created. They are used to initialize the properties of the object with specific values. In Java, constructors are defined within a class using the same name as the class. We have already seen an example of a constructor in our Product class definition.

Let's create an object of the Product class and call the getTotalValue method to calculate the total value of the product:

```
Product laptop = new Product("Dell Inspiron", 800.0, 2);
double totalValue = laptop.getTotalValue();
System.out.println("Total value of " + laptop.getName() + " is " + totalValue);
```

In this example, we have created an object of the Product class called "laptop". We have passed the values "Dell Inspiron", 800.0, and 2 to the constructor to initialize the properties of the object. We have then called the getTotalValue method on the object to calculate the total value of the product and stored it in the variable "totalValue". We have then printed out the name of the product and its total value using the getName method and the totalValue variable.

In addition to the constructor we have seen in our Product class example, there are two other types of constructors in Java: parameterless constructors and overloaded constructors.

A parameterless constructor, also known as a default constructor, is a constructor that takes no parameters. If a class does not have any constructors defined, Java automatically provides a default constructor. Let's add a parameterless constructor to our Product class:

```
public Product() {
    this.name = "";
    this.price = 0.0;
    this.quantity = 0;
}
```

In this example, we have defined a parameterless constructor that initializes the properties of the object with default values.

An overloaded constructor is a constructor that has a different set of parameters than the other constructors in the class. This allows us to create objects of the class with different initial values for the properties. Let's add an overloaded constructor to our Product class:

```
public Product(String name, double price) {
    this.name = name;
    this.price = price;
    this.quantity = 0;
}
```

In this example, we have defined an overloaded constructor that takes only the name and price properties as parameters. The quantity property is initialized with a default value of 0.

Another important concept to understand when working with constructors in Java is the "this" keyword. The "this" keyword refers to the current object being created. It is used to distinguish between properties of the current object and parameters with the same name.

For example, in our Product class constructor, we use "this" to refer to the properties of the current object:

```
public Product(String name, double price, int quantity) {
    this.name = name;
    this.price = price;
    this.quantity = quantity;
}
```

In this example, we use "this.name" to refer to the name property of the current object, and "name" to refer to the parameter passed to the constructor.

Exercises:

1. Create a class called "Book" with properties for title, author, number of chapters and number of pages. Add a constructor that takes these properties as parameters and sets the values accordingly. Add getter and setter methods for each property. Add a method to calculate the average number of pages per chapter.
2. Create an object of the Book class and call the method to calculate the average number of pages per chapter.
3. Add a private field to the Book class called "isbn". Add a getter and setter method for the isbn field. Modify the constructor to additionally take the isbn as a parameter and set the value accordingly.
4. Add a parameterless constructor to the Book class that initializes the properties of the object with default values.
5. Add an overloaded constructor to the Book class that takes only the title and author properties as parameters. The number of pages property should be initialized with a default value of 0.

References:

4. Java: The Complete Reference, Herbert Schildt, McGraw Hill
5. JDK Download Link - https://www.oracle.com/java/technologies/downloads/ 6. Eclipse Download Link - https://www.eclipse.org/downloads/