Lab 4

Contents:

Method Overloading and Call by Reference.

Background:

Product Class:

Let's create a Product class that contains the following properties: ● Name (String)
● Price (double)
● Quantity (int)
We will then create a constructor and getter methods for each property.

```
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;
}
}
```

Method Overloading:

Method overloading allows us to define multiple methods with the same name but different parameters. Java decides which method to call based on the number and type of arguments passed. Let's create an example of method overloading in our Product class. We will create two methods that calculate the total price of a product, but with different types of parameters.

```
public class Product {
  // Product class code here
   public double calculateTotalPrice(int quantity) {
     return price * quantity;
   }

   public double calculateTotalPrice(double discount) {
     return price * quantity * (1 - discount);
   }

   public double calculateTotalPrice(double taxRate, double discount)
   { double totalPriceBeforeTax = price * quantity * (1 - discount);
   return totalPriceBeforeTax + totalPriceBeforeTax * taxRate; }
}
```

In the example above, we have created two methods with the same name "calculateTotalPrice", but one takes an integer parameter "quantity" and the other takes a double parameter "discount". we have added another method "calculateTotalPrice" that takes two double parameters, "taxRate" and "discount". This method calculates the total price of the product after applying a discount and adding tax.

Call By Reference:

Let’s add the following method to the Product class.

```
public void updateProduct(Product product) {
    // check if the product being passed is not null
    if (product != null) {
        // update the properties of this object with the properties of
the passed object
        this.name = product.getName();
        this.price = product.getPrice();
        this.quantity = product.getQuantity();
    }
}
```

In the example above, we have added a method "updateProduct" that takes a Product object as a

parameter. This method updates the properties of the current object with the properties of the passed object.
To call this method and pass a Product object as a reference, we would do something like this:

```
Product p1 = new Product("Product A", 10.0, 5);
Product p2 = new Product("Product B", 20.0, 3);
p1.updateProduct(p2);
```

The above code creates two Product objects "p1" and "p2". It then calls the "updateProduct" method of "p1", passing "p2" as the parameter. This will update the properties of "p1" to have the same values as "p2". The new values of "p1" will be "Product B", price 20.0, and quantity 3.
This shows how call by reference can be used to update the properties of an object passed as a parameter, allowing us to reuse the same object in different parts of our code without creating new instances.