Practices for Lesson 09

Practices Overview

In these practices, you create and use methods with arguments. In a challenge exercise, you create an overloaded method.

Practice 9-1: Writing a Method That Uses Arguments and Return Values

Overview

In this practice, you create a class to order more than one shirt and then display the total order value of the shirts.

Assumptions

This practice assumes that the following files appear in the practice folder for this lesson, `Lesson09`:

- Order.java
- Shirt.java

Tasks

1. Create a new project from existing source called Practice10. Set the **Source Package Folder** to point to `Lesson09`. Remember to also change the Source/Binary Format property. If you need further details, refer to Practice 1-2, Steps 3 and 4.
2. Open the Shirt class and examine the member fields and the method it contains.
3. Open the Order class and examine its member fields and method.
4. Create a new Java Main Class called "OrderTest".
5. Add code to the main method that adds a shirt to a new order and display the total amount of the order. The high-level steps for this are provided in the table below. More detailed assistance is given following the table.

Step	Code Description	Choices or Values
a.	Create and initialize two objects.	Object type: `Shirt` Object type: `Order`
b.	Declare and initialize a local variable of type double.	Variable Name: `totalCost` Value: 0.0
c.	Assign a value to the `price` field of the `Shirt` object.	Value: 14.99
d.	Invoke the `addShirt` method of the `Order` object.	Method argument: the `Shirt` object Method return: assign to `totalCost`
e.	Display the return value with a suitable label.	Example output: "Total amount for the order is $14.00"

Further details:

The documentation for the `addShirt` method is as follows:

Adds a shirt to a list of shirts in an order

Syntax:

```
public double addShirt (Shirt shirt)
```

Parameters:

shirt – An object reference to a Shirt object

Returns:

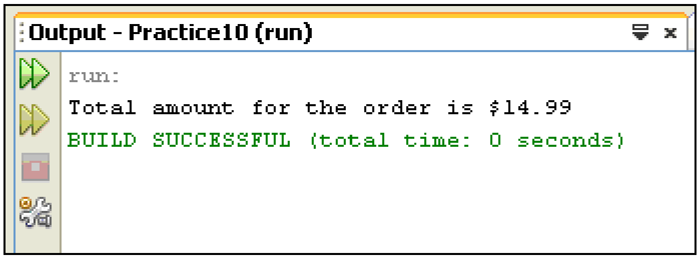
A total current amount for the order

Solution:

```
public static void main(String[] args){
     Order order = new Order();
     Shirt shirt = new Shirt();
     double totalCost = 0.0;

     shirt.price = 14.99;
     totalCost = order.addShirt(shirt);
     System.out.println("Total amount for the order is $" +
           totalCost);
}
```

6. Save and compile your program. Test the order process by running the OrderTest class.



```
run:
Total amount for the order is $14.99
BUILD SUCCESSFUL (total time: 0 seconds)
```

7. In the `main` method of OrderTest, create additional `Shirt` objects, assign values to the `price` field of each new `Shirt` object, and add the `Shirt` objects to your order by invoking the `addShirt` method.

   Note: Remember that the `addShirt` method adds the price of the shirt argument object to the `totalPrice` field of the `Order` object. Therefore the `totalPrice` value grows with each addition of a shirt. You only need to capture the return value of the final `addShirt` method call to get the `totalCost` value.

Solution:

```
public static void main(String[] args) {
        Order order = new Order();
        Shirt shirt = new Shirt(),
              shirt2 = new Shirt(),
              shirt3 = new Shirt();
        double totalCost = 0.0;

        shirt.price = 14.99;
        shirt2.price = 23.55;
        shirt3.price = 49.99;
        order.addShirt(shirt);
        order.addShirt(shirt2);
        totalCost = order.addShirt(shirt3);
        System.out.println("Total amount for the order is $" +
    totalCost);
}
```

8. Save and compile the program and, once again, run the OrderTest class to test it. Make sure that the amount displayed is the total of all of the shirt prices.

Challenge Practice 9-2: Writing a Class That Contains an Overloaded Method

This practice is optional. Check with your instructor for recommendations about which optional practices to do. Perform this practice only if you are certain that you have enough time to perform all of the non-optional practices.

Overview

In this practice, you write a Customer class with an overloaded method called setCustomerInfo.

Assumptions

This practice assumes that the CustomerTest.java file appears in the practice folder for this lesson, `Lesson09`, and consequently also in the Practice10 project.

Tasks

1. Create a new Java Class called "Customer". Declare the following fields and initialize them as shown in the table below:

Field	Type	Initial Value
`customerID`	int	`0`
`name`	`String`	`"-name required-"`
`address`	`String`	`"-address required-"`
`phoneNumber`	`String`	`"-phone required-"`
`eMail`	`String`	`"-email optional-"`

2. **High-level instructions:** Within the Customer class, add two overloaded methods called `setCustomerInfo`. Depending upon how the `setCustomerInfo` method is called, it does one of the following:
   - Sets the ID, name, address, and phone number for a Customer object. (This is the minimum information needed for a new customer.)
   - Sets the ID, name, address, phone number, and email address for a Customer object.

   Detailed Instructions:
   - The two method signatures should be as follows:

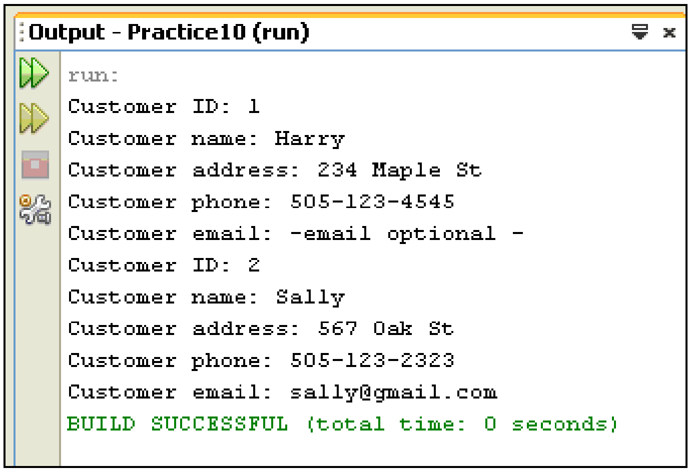
   ```
   public void setCustomerInfo(int Id, String nm, String addr,
       String phNum)
   public void setCustomerInfo(int Id, String nm, String addr,
       String phNum, String email)
   ```
   - Within each method, assign each argument in the method to its corresponding field.
3. Create a `display` method to display the values of all the member fields of the Customer class.
4. Save and compile the program.

5. Open the CustomerTest class. Modify the `main` method as follows so that it can be used to test the overloaded methods of the Customer class.
    - Create two object references to different Customer objects.
    - Use each variation of the `setCustomerInfo` method to provide information for each Customer object.
    - Display the contents of each Customer object.

    Solution:

```
public static void main(String[] args){
    Customer c1 = new Customer(),
              c2 = new Customer();
    c1.setCustomerInfo(1,”Harry”,”234 Maple St”,
               ”505-123-4545”);
    c2.setCustomerInfo(2,”Sally”,”567 Oak St”,
               ”505-123-2323”,”sally@gmail.com”);
    c1.display();
    c2.display();
}
```

6. Once again, save and compile the program. Run the CustomerTest file to test the program. Make sure that the output is different for each of the display methods.



```
run:
Customer ID: 1
Customer name: Harry
Customer address: 234 Maple St
Customer phone: 505-123-4545
Customer email: -email optional -
Customer ID: 2
Customer name: Sally
Customer address: 567 Oak St
Customer phone: 505-123-2323
Customer email: sally@gmail.com
BUILD SUCCESSFUL (total time: 0 seconds)
```