Ex. No. 1

MusicMania Gallery is a popular music gallery located in New Jersey, USA that sells music and video CDs. The management has recently automated all the transactions such as sales, data entry, purchases, and so on. However, the software is not working as expected and takes long time for completing transactions. The management has decided to optimize the software for better performance.

The CEO of the company and a team of experts have chosen your company to provide a solution for the same. After analysis of the software, the team has come to the conclusion that the reason of imperfect functioning is that all the classes are in the same package and vulnerable to external access. You, as part of the team, have been assigned the following tasks to improve the structure and function of the software

Create an application using different packages and access control specifiers to implement the MusicMania Gallery system. The application should consist of the following:

i. A Java package named cdpkg.

ii. A Java package named orderpkg.

iii. A class named CompactDisc in the package cdpkg to add and display details of the CDs.

iv. A class named Order to add display order details of customers.

v. A Java main class named BuyCD inside the orderpkg package to access the CompactDisc and Order classes and to run the application.

Each file has a specific purpose and functionality. The descriptions of each file are as follows:

CompactDisc.java

ID: A String variable to store the ID of the CD

type: A String variable to store the type of CD: music or video

artist: A String variable to store the name of the artist

Ø price: A public static double variable to store the retail price of the album

Ødiscount: A static final float variable with a fixed value that indicates the discount in percentage allowed on the total amount of purchase

It consists of a constructor and methods to display CD details.

Order.java

The Order class is declared in the package named orderpkg. It stores the following details about the purchases made by a customer:

Ø orderID: A String variable to store the ID of purchase order

custID: A String variable to store the ID of the customer

Ø custName: A String variable to store the name of the customer

Ø quantity: An integer variable to store the number of CDs purchased

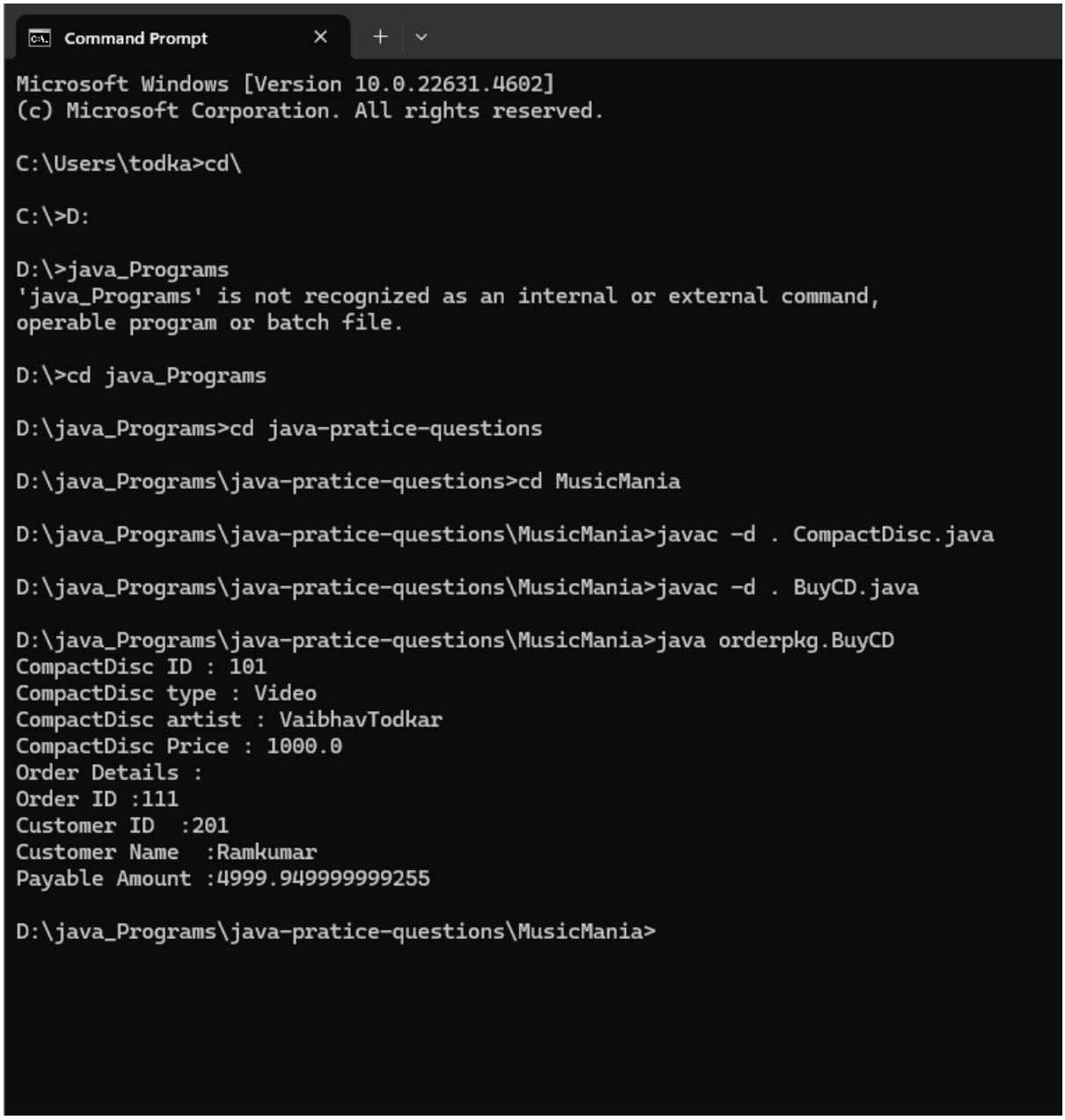
payableAmt: A private double variable to store the amount payable on a purchase

It consists of methods to calculate the payable amount by multiplying the price of the CD with the quantity and deducting the discount from the final amount. Also, display it along with the other order details.

BuyCD.java

This class creates instances of the CompactDisc and Order classes. It invokes the methods to add the details of CDs as well as method to calculate and display the payable amount with order details. The BuyCD class must import the cdpkg package to use the CompactDisc class. The details will be specified during runtime at command-line. The access specifiers of the variables can be modified according to programmer's discretion and the requirement of the code.

Output:



Code :

CompactDisc.java

```
package cdpkg;

public class CompactDisc{

    String id;

    String type; // music or video

    String artist; // name
```

```
    public double price;
    public final static float discount = 5;

    // constructor

    public CompactDisc(String id, String type, String artist, double price ){
        this.id = id;
        this.type = type;
        this.artist = artist;
        this.price = price;

    }

    public void compactDiscDisplay(){
        System.out.println("CompactDisc ID : " + id);
        System.out.println("CompactDisc type : " + type);
        System.out.println("CompactDisc artist : " + artist);

        // price -= price * (5/100);-
        System.out.println("CompactDisc Price : " + price);
    }
}
```

BuyCD.java

```
package orderpkg;
import cdpkg.CompactDisc;

import java.text.CompactNumberFormat;

class Order  {
    String orderID;
    String custID;
```

```
    String custName;
    int quantity;

    double payableAmt = 0;

    Order(String orderId, String custID, String custName, int quantity){
        this.orderID = orderId;
        this.custID = custID;
        this.custName = custName;
        this.quantity = quantity;

    }

    public void calculatePayableAmount(double price, float discount){
        payableAmt = (quantity * price) - (discount/100) ;

        System.out.println("Order Details :");
        System.out.println("Order ID :" + orderID);
        System.out.println("Customer ID  :" + custID);
        System.out.println("Customer Name  :" + custName);
        System.out.println("Payable Amount :" + payableAmt);
    }
}

public class  BuyCD{
    public static void main(String[] args){
        CompactDisc cd = new CompactDisc("101", "Video", "VaibhavTodkar", 1000);
        Order o = new Order("111", "201", "Ramkumar", 5);
        cd.compactDiscDisplay();
        o.calculatePayableAmount(cd.price, cd.discount)
}
}
```

Ex No 2:

1.

The Australian cricket team is the best team in the world. It has proved its potential over the years since 1999 by its consistent performance in all the matches. The Australian Cricket Board that handles the payment related issues of the players has decided to develop software that would automatically calculates the income of the players based on their grade, the number of matches each player plays, and their performance in the tournament. To accomplish this, the Australian Cricket Board has hired a team of developers. Consider yourself to be a part of the team. You have been assigned the following tasks for designing the software.

Create an application using inheritance and polymorphism to implement the software. The application should consist of the following files:

1. Game.java
2. TestMatch.java
3. WorldCup.java
4. Player.java
5. PlayerTest.java

Each file has a specific purpose and functionality. Descriptions of each file are as follows:

Game.java

The Game class is an abstract base class that provides abstract method named double calculateIncome(String numGames) to calculatethe income of the player and double calculateBonus (String performance, String grade) to calculate the bonus based on their performance in a match and the grade. The performance is rated as good, average, and best.

TestMatch.java

The TestMatch class inherits the Game class and overrides the abstract methods to calculate the income and grade based on the test matches played by the player.

WorldCup.java

The WorldCup class inherits the Game class and overrides the abstract methods to calculate the income and grade based on the number of world cup matches played by the player.

Player.java

The Player class contains an instance variables to stores the details of the player such as name, age, gender, and so on and a method displayDetails(String match) to display the details of the player. The displayDetails(String match) method invokes the calculateIncome() and calculateBonus() methods of the TestMatch or WorldCup classes based on the type of match specified by the user, that is, Test Match or World Cup.

PlayerTest.java

The PlayerTest class creates an instance of the Player class and passes appropriate arguments to the constructor. Also, the class displays the details of the player such as personal details, income, and bonus by invoking the display Details() method using the Player class object

Output:

```
PS D:\java_Programs\java-pratice-questions\Australiancricketteam> javac PlayerTest.java
PS D:\java_Programs\java-pratice-questions\Australiancricketteam> java PlayerTest
Player Name : Vaibhav Todkar
Player age : 21
Player gender : Male
Player Income : 30000.0
Player IBonus : 100000.0
Player Name : Vaibhav Todkar
Player age : 21
Player gender : Male
Player Income : 50000.0
Player IBonus : 25000.0
PS D:\java_Programs\java-pratice-questions\Australiancricketteam>
```

PlayerTest.java

```
abstract class Game {
    abstract public double calculateIncome(String numGames);  // calculatethe income of the player
    abstract public double calculateBonus(String performance, String grade);  //calculate the bonus based on their performance
}



class TestMatch extends Game {
   public  double calculateIncome(String numGames){
        int games = Integer.parseInt(numGames);
        double income = games * 10000;
        return income;
    }
```

```
    public double calculateBonus(String performance, String grade) {
        double bonus = 0;
        switch (performance) {
            case "Best":
                bonus = 50000;
                break;
            case "Good":
                bonus = 25000;
                break;
            case "Average":
                bonus = 5000;
                break;
        }

        if (grade == "A") {
            bonus = bonus * 2;
        }else if(grade == "B"){
            bonus = bonus * 1;
        }else{
            System.out.println("Sorry Next Time");
        }

        return bonus;
    }
}



class WorldCup extends Game {
    public  double calculateIncome(String numGames){
        int games = Integer.parseInt(numGames);
```

```
        double income = games * 10000;
        return income;
    }

    public double calculateBonus(String performance, String grade) {
        double bonus = 0;
        switch (performance) {
            case "Best":
                bonus = 50000;
                break;
            case "Good":
                bonus = 25000;
                break;
            case "Average":
                bonus = 5000;
                break;
        }

        if (grade == "A") {
            bonus = bonus * 2;
        }else if(grade == "B"){
            bonus = bonus * 1;
        }else{
            System.out.println("Sorry Next Time");
        }

        return bonus;
    }
}
class Player {
    String name;
    int age;
```

```
    String gender;

    Player(String name, int age, String gender){
        this.name = name;
        this.age = age;
        this.gender = gender;
    }

    public void displayDetails(String match, String numGames, String performance, String grade){
        Game game ;
        if(match == "TestMatch"){
            game = new TestMatch();
        }else{
            game = new WorldCup();
        }

        double income = game.calculateIncome(numGames);
        double bonus = game.calculateBonus(performance, grade);

        System.out.println("Player Name : " + name);
        System.out.println("Player age : " + age);
        System.out.println("Player gender : " + gender);
        System.out.println("Player Income : " + income);
        System.out.println("Player IBonus : " + bonus);

    }


}


public class PlayerTest {
```

```
    public static void main(String[] args) {
        Player  p = new Player("Vaibhav Todkar", 21, "Male");
        p.displayDetails("TestMatch", "3", "Best", "A");

        p.displayDetails("TestMatch", "5", "Good", "B");
    }
}
```

Ex No 3:

Smart Toys is a famous toy manufacturing company located in New York, USA. The company manufactures and sells different types of automated toys. With the increasing number and types of automated toys, the company is finding it difficult to keep track of the demand and supply of the toys. For this purpose, the management has decided to automate the following tasks in the company:

Adding and retrieving toy information

Testing the product

Order management

You, as a developer, have been assigned the task to provide the solution for the same. Create an application to accomplish the task. The application should consist of the following classes and interfaces:

Interfaces

Testing.java: This interface should declare methods to test the product such as moveObject(), stopObject(), startObject(), turnObject(), and so on. The Toy class should implement this interface and define the methods for the toys.

Classes

Toy.java: This class should store the basic information about a toy such as Id, name, price, color, and type. It should consist of a method to display toy details.

Order.java: This class should store the order information such as order Id, quantity, and payable amount. It should consist of a method to display order details.

Stock.java: This class should be nested within the Order class. It should have a method int getStock(String toyId) that accepts the toy id as a parameter and returns the available stock value of the toy.

TestToy.java: This class should contain the main() method for execution of the program. It should create an instance of the Toy and Order classes with appropriate arguments. Next, the class should invoke various methods to display the toy details, order details, and methods to test the toy.

Output:

```
PS D:\java_Programs\java-pratice-questions\SmartToys> javac TestToy.java
PS D:\java_Programs\java-pratice-questions\SmartToys> java TestToy
Toy id :101
Toy name :Robot
Toy price :50.0
Toy color :Red
Toy type :Machine
Toy Started
Toy Moved
Toy Turned
Toy Stoped
Order Id : 001
Order quantity : 2
Order Pay Able Amount : 100.0
Available Stock for Toy ID 101: 100
PS D:\java_Programs\java-pratice-questions\SmartToys>
```

TestToy.java

```
interface Testing {

    void moveObject();
    void stopObject();
    void startObject();
    void turnObject();
}

class Toy implements Testing {
    String id;
    String name;
    double price;
    String color;
    String type;

    Toy(String id, String name, double price, String color, String type){
```

```
        this.id = id;
        this.name = name;
        this.price = price;
        this.color = color;
        this.type = type;
    }

    public void toyDetails(){
        System.out.println("Toy id :"+  id);
        System.out.println("Toy name :"+  name);
        System.out.println("Toy price :" + price);
        System.out.println("Toy color :"+  color);
        System.out.println("Toy type :" + type);
    }

    public void moveObject() {
        System.out.println("Toy Moved");
    }

    public void stopObject() {
       System.out.println("Toy Stoped");
    }

    public void startObject() {
        System.out.println("Toy Started");
    }

    public void turnObject() {
       System.out.println("Toy Turned");
    }
}
```

```
class Order {
    String orderId;
    int quantity;
    double payableAmount;
    public Order(String orderId, int quantity, double payableAmount) {
        this.orderId = orderId;
        this.quantity = quantity;
        this.payableAmount = payableAmount;
    }

    void orderDetails(){
        System.out.println("Order Id : " + orderId);
        System.out.println("Order quantity : " + quantity);
        System.out.println("Order Pay Able Amount : " + payableAmount * quantity);
    }

    static class Stock {
        public int getStock(String topId) {
            return 100;
        }
    }
}

public class TestToy {
    public static void main(String[] args) {
        Toy toy = new Toy("101", "Robot", 50, "Red", "Machine");

        toy.toyDetails();

        toy.startObject();
```

```
        toy.moveObject();
        toy.turnObject();
        toy.stopObject();

        Order order = new Order("001", 2, 50);

        order.orderDetails();

        Order.Stock stock = new Order.Stock();
        int availableStock = stock.getStock("101");
        System.out.println("Available Stock for Toy ID 101: " + availableStock);

    }
}
```