COMP2396B Object-oriented Programming and Java

Tutorial 4 – State and Behaviour of Objects (2) & ArrayList

Outline:

1. Concept Overview
2. Exercises on Class and Object
3. Exercises on ArrayList

Part 1: Concept Overview

In this tutorial, we will do some practice to revise the concepts in the following lecture slides:

State and Behaviour of Objects

Topic	Readings
Essential Elements in Class Declaration (e.g. parameters, return value, getters, setters, constructors)	**Course Materials:** Lecture 4 – P.8-19
State and Behaviour of Objects	**Course Materials:** Lecture 4 – P.1-7
Basic Concepts in Inheritance	**Course Materials:** Lecture 4 – P.20-23

ArrayList

Topic	Readings
Basics of ArrayList (E.g. Methods, Packages)	**Course Materials:** Lecture 5 – P.5-7, 14-17
Array Vs ArrayList	**Course Materials:** Lecture 5 – P.2-4, 8-13
Wrapper Class, Autoboxing, Number Formatting	**Course Materials:** Lecture 5 – P.18-34

Part 2: Exercises on State and Behaviour of Objects

1. Study **Tester.java** and the sample test case below. Create the following classes and their respective methods:

    - **ElectronicsProduct** class with attributes for product ID, name, and price.
    - **Television** class, which is a subclass of the **ElectronicsProduct** class. It should have an extra attribute that records it's warranty period (in months).

Tester.java

```
import java.io.*;

public class Tester {
    public static void main(String[] args) throws IOException {

        InputStreamReader isr = new InputStreamReader(System.in);
        BufferedReader inData = new BufferedReader(isr);

        String c1_info[] = inData.readLine().split(" ");
        String c2_info[] = inData.readLine().split(" ");

        // Part 1: ElectronicsProduct object
        String productID1 = c1_info[0];
        String name1 = c1_info[1];
        double price1 = Double.valueOf(c1_info[2]);

        ElectronicsProduct product = new ElectronicsProduct(productID1, name1, price1);

        // Apply a discount and display the final price
        product.applyDiscount(10);
        System.out.println("Product ID: " + product.getProductId());
        System.out.println("Name: " + product.getName());
        System.out.println("Price after discount: $" + product.getPrice());
        System.out.println("--------------------");

        // Part 2: WashingMachine object
        String productID2 = c2_info[0];
        String name2 = c2_info[1];
        double price2 = Double.valueOf(c2_info[2]);
        int warrantyPeriod2 = Integer.valueOf(c2_info[3]);

        Television tv = new Television(productID2, name2, price2, warrantyPeriod2);

        // Apply a discount and display the final price
        tv.applyDiscount(15);
        System.out.println("Product ID: " + tv.getProductId());
        System.out.println("Name: " + tv.getName());
        System.out.println("Price after discount: $" + tv.getPrice());
        System.out.println("Warranty period: " + tv.getWarrantyPeriod() + " months");

        // Extend the warranty period and display the new warranty period
        tv.extendWarranty(12);
        System.out.println("Warranty period: " + tv.getWarrantyPeriod() + " months");

    }
}
```

Remarks:

- You cannot use System.out.println() or other print functions in the **ElectronicsProduct** and the **Television** class.

Sample Test Cases:

Test case 1:
Input:
WM123 Cooker 900
W456 Panasonic 800 24

Output:
Product ID: WM123
Name: Cooker
Price after discount: $810.0

Product ID: W456
Name: Panasonic
Price after discount: $680.0
Warranty period: 24 months
Warranty period: 36 months

Test case 2:
Input:
WM456 Cooker 800
W234 LG 650 12

Output:
Product ID: WM456
Name: Cooker
Price after discount: $720.0

Product ID: W234
Name: LG
Price after discount: $552.5
Warranty period: 12 months
Warranty period: 24 months

Part 3: Exercises on ArrayList

2. Convert the following array program into ArrayList version:

```
public class Card {
    public static void main(String[] args) {
        int[] deck = new int[52];
        String[] suits = {"Spade", "Heart", "Diamond", "Club"};

        // Initialize cards
        for (int i = 0; i < deck.length; i++) {
            deck[i] = i;
        }

        // Shuffle the cards.
        for (int i = 0; i < deck.length; i++) {
            int index = (int) (Math.random() * deck.length);
            int temp = deck[i];
            deck[i] = deck[index];
            deck[index] = temp;
        }

        // Display the card(s) that is heart from the first five cards.
        for (int i = 0; i < 5; i++) {
            String suit = suits[deck[i] / 13];
            if (suit.equals("Heart")) {
                System.out.println("Card number " + deck[i] + " is " + suit);
            }
        }
    }
}
```

3. Heung Shing Exhibition Center uses an event logging system for keeping tracks of guest's activities in events. Events are something too big and should be held in different venues. **Note that a guest can join multiple events and can join the same event in another venue.**

   Due to an epidemic, the system can check whether pairs of guests have joined the same venue of the same event to determine whether they have close contact.

   You are given a program **Main.java** below. Creates all classes to complete the program.

```java
public class Main {
    public static void main(String[] args) {
        Guest wing = new Guest("Wing");
        Guest joy = new Guest("Joy");
        Guest marco = new Guest("Marco");

        Event bookFair = new Event("Book Fair");
        bookFair.addVenue("Hall A");
        bookFair.addVenue("Room 1");

        wing.joinEvent(bookFair, "Hall A");
        joy.joinEvent(bookFair, "Room 1");
        marco.joinEvent(bookFair, "Room 1");

        Event gunplaExpo = new Event("Gunpla Expo");
        gunplaExpo.addVenue("Room 1");

        wing.joinEvent(gunplaExpo, "Room 1");

        System.out.println(wing.hasCloseContactWith(joy)); // output: false
        System.out.println(wing.hasCloseContactWith(marco)); // output: false
        System.out.println(joy.hasCloseContactWith(marco)); // output: true
    }
}
```

Remarks:

- hasCloseContactWith() should return a boolean type value
- You must not use System.out.println() in your classes.

Sample Test Cases:

Test case 1:
Input:

Output:
false
false
true

4. Hang Ten Restaurant uses a reservation system to handle guests. The restaurant is on first-come-first-served basis. There are 4 tables in restaurant, T1, T2 with 2 seats and T3, T4 with 4 seats. Note that the table is allowed to share with other guests.

   You are given a program **Main.java** below. Create all classes to complete the program.

```
import java.io.*;

public class Main {
    public static void main(String[] args) throws Exception {
        Restaurant res = new Restaurant("Hang Ten Restaurant");
        res.add_table("T1", 2);
        res.add_table("T2", 2);
        res.add_table("T3", 4);
        res.add_table("T4", 4);

        InputStreamReader isr = new InputStreamReader(System.in);
        BufferedReader inData = new BufferedReader(isr);

        String input = inData.readLine();
        while (!input.equals("Exit")) {
            int n = Integer.parseInt(input);
            System.out.println(res.make_reservation(n));
            input = inData.readLine();
        }

        System.out.println("bye");
    }
}
```

Remarks:

- make_reservation() should return a string for program output.
- You must not use System.out.println() in your classes.

Sample Test Cases:

Test case 1:
Input:
2
4
2
1
3
3
Exit

Output:
Table T1 for 2 guest(s).
Table T3 for 4 guest(s).
Table T2 for 2 guest(s).
Table T4 for 1 guest(s).
Table T4 for 3 guest(s).
No table available for 3 guest(s).
bye

Test case 2:
Input:
4
1
1
1
1
Exit

Output:
Table T3 for 4 guest(s).
Table T1 for 1 guest(s).
Table T2 for 1 guest(s).
Table T4 for 1 guest(s).
Table T1 for 1 guest(s).
bye

Sample Solution:

1. ElectronicsProduct.java

```java
public class ElectronicsProduct {

    private String productId;
    private String name;
    private double price;

    // Constructor to initialize the ElectronicsProduct object
    public ElectronicsProduct(String productId, String name, double price) {
        this.productId = productId;
        this.name = name;
        this.price = price;
    }

    public void applyDiscount(double discountPercentage) {
        double discountAmount = price * discountPercentage / 100;
        price -= discountAmount;
    }

    public double getPrice() {
        return price;
    }

    public String getProductId() {
        return productId;
    }

    public String getName() {
        return name;
    }

}
```

Television.java

```java
public class Television extends ElectronicsProduct {

    private int warrantyPeriod;

    // Constructor to initialize the Television object
    public Television(String productId, String name, double price, int warrantyPeriod) {
        super(productId, name, price);
        this.warrantyPeriod = warrantyPeriod;
    }

    public void extendWarranty(int additionalMonths) {
        warrantyPeriod += additionalMonths;
    }

    public int getWarrantyPeriod() {
        return warrantyPeriod;
    }
}
```

2. Solution.java

```java
import java.util.ArrayList;

public class Solution {
    public static void main(String[] args) {
        ArrayList<Integer> deck = new ArrayList<Integer>();
        ArrayList<String> suits = new ArrayList<String>();
        suits.add("Spade");
        suits.add("Heart");
        suits.add("Diamond");
        suits.add("Club");

        // Initialize cards
        for(int i = 0; i < 52; i++) {
            deck.add(i); // auto-boxing
        }

        // Shuffle the cards.
        for(int i = 0; i < deck.size(); i++) {
            int index = (int) (Math.random() * deck.size());
            int temp = deck.get(i);
            deck.set(i, deck.get(index));
            deck.set(index, temp);
        }

        // Display the card(s) that is heart from the first five cards.
        for(int i = 0; i < 5; i++) {
            String suit = suits.get(deck.get(i) / 13);
            if (suit.equals("Heart")) {
                System.out.println("Card number " + deck.get(i) + " is " + suit);
            }
        }
    }
}
```

3. EventVenue.java

```
public class EventVenue {

    String venueName;

    public EventVenue(String name) {
        this.venueName = name;
    }

    public String getName() {
        return this.venueName;
    }

}
```

Event.java

```
import java.util.ArrayList;

public class Event {

    String name;
    ArrayList<EventVenue> eventVenues;

    public Event(String name) {
        this.name = name;
        this.eventVenues = new ArrayList<EventVenue>();
    }

    public void addVenue(String name) {
        this.eventVenues.add(new EventVenue(name));
    }

    public EventVenue findEventVenue(String name) {
        for (EventVenue v : eventVenues) {
            if (v.getName().equals(name)) {
                return v;
            }
        }
        return null;
    }

}
```

Guest.java

```java
import java.util.ArrayList;

public class Guest {

    String name;
    ArrayList<EventVenue> joinedEventVenues;

    public Guest(String name) {
        this.name = name;
        this.joinedEventVenues = new ArrayList<EventVenue>();
    }

    public void joinEvent(Event event, String venueName) {
        EventVenue venue = event.findEventVenue(venueName);
        if (venue != null) {
            joinedEventVenues.add(venue);
        }
    }

    public boolean hasCloseContactWith(Guest guest) {
        for (EventVenue v : joinedEventVenues) {
            if (guest.joinedEventVenues.contains(v)) {
                return true;
            }
        }
        return false;
    }

}
```

4. Restaurant.java

```
import java.util.*;

public class Restaurant {

    private String name;
    private ArrayList<Table> tables = new ArrayList<Table>();

    public Restaurant(String resName) {
        name = resName;
    }

    public void add_table(String tableName, int headCount) {
        Table table = new Table(tableName, headCount);
        tables.add(table);
    }

    public String make_reservation(int headCount) {
        for (Table t : tables) {
            if ((t.getUnused() == true) && t.getChairLeft() >= headCount) {
                t.reduceChair(headCount);
                return "Table " + t.getName() + " for " + headCount + " guest(s).";
            }
        }

        for (Table t : tables) {
            if (t.getChairLeft() >= headCount) {
                t.reduceChair(headCount);
                return "Table " + t.getName() + " for " + headCount + " guest(s).";
            }
        }

        return "No table available for " + headCount + " guest(s).";
    }

}
```

Table.java

```
public class Table {

     private String name;
     private int chairLeft;
     private boolean unused;

     public Table(String tableName, int chair) {
          name = tableName;
          chairLeft = chair;
          unused = true;
     }

     public void reduceChair(int num) {
          chairLeft -= num;
          unused = false;
     }

     public String getName() {
          return name;
     }

     public int getChairLeft() {
          return chairLeft;
     }

     public boolean getUnused() {
          return unused;
     }

}
```