SP21-BSE-090

**Comsats University Islamabad**
**Department of Computer Science**
**Final Examination, Spring 2022**

| | | |
|---|---|---|
| Class/Section: - | BSSE – 3A,3B | Marks: - 50 |
| Subject: - | Object Oriented Programming | Time: - 3 hours |
| Instructor: - | Ms. Fazila Shams | Dated: - July 01, 2022 |

**Question 1.** Give short answers for the following: **(10 marks) (20 min) [CLO 1]**

a. "Java is not 100% Object Oriented", Please justify the statement either you agree or disagree?

b. Name the types of inheritances? Which type of inheritance is not supported by Java and how to cure it?

c. What is the advantage of Generic types?

d. Assume that we have five classes: Person, Employee, Teacher, Student, and PhDStudent. Employee and Student are both the subclasses of Person. PhDStudent is a subclass of Student. Teacher is a subclass of Employee. Which of the following assignments are valid?

a. Person p1 = new Student();

b. Person p2 = new PhDStudent();

c. Person p3=new Employee();

d. PhDStudent phd1 = new Student();

e. PhDStudent phd2=new Employee();

f. PhDStudent phd3=new Person();

g. Teacher t1 = new Person();

h. Student s1 = new PhDStudent();

i. Employee e1=new Teacher();

j. Teacher t2=new Student();

**Question 2.** Consider the following classes

```
public class Weapon {
    private String name;
    private double cost;

    public Weapon(String name,
double cost) {
        this.name = name;
        this.cost = cost;
    }
    public double getCost() {
        return cost;
    }
```

```
public class Soldier {
    private String name;
    private int age;
    private double salary;

    public Soldier(String name, int
age, double salary) {
        this.name = name;
        this.age = age;
        this.salary = salary;
    }
     public double getSalary() {
```

```
}
```

```
    return salary;
  }
}
```

```
public class Gun extends Weapon {
    private String bulletType;

    public Gun(String name, double
cost, String bulletType) {
      super(name, cost);
      this.bulletType = bulletType;
    }
}
```

```
public class Missile extends
Weapon{
    private int rangeInKm;

    public Missile(String name,
double cost, int rangeInKm) {
      super(name, cost);
      this.rangeInKm = rangeInKm;
    }
}
```

```
public interface showState {
    String to_String();
}
```

Write definition of methods and constructor in following class.

**(10 marks) (30 min) [CLO 3]**

```
public class MilitaryBase implements showState{
   private Weapon w_array [] = new Weapon[10];
   private Soldier s_array[] = new Soldier[10];
    public MilitaryBase( ) {
        //Argumented constructor
    }
   public int countMissiles()
   {
      //return the number of missiles in the military base
   }
   public int countGuns()
   {
       //return the number of Guns in the military base
   }
   public double calculateCostOfMilitaryBase()
   {
       //Cost of military base is the cost of Guns and Missiles and
payment of soldiers
    }
}
Note: Remember that this class implements an interface
```

**Question 3.** Suppose you are designing a "shopManagement" application in Java using OOP. Your shop management application contains the following classes.

a. ShopItem : Represents the general definition of shop item

b. Mobilephone: Represents a specific shop item.

c. LED : Represents a specific shop item.

d. Shop: Contains a collection of shop Items in form of array list. All items are added to array list and complete array list is written to File for permanent storage. For deletion of items from shop, items are deleted from array list and updated list is written to file. Please note that the constructor of the Shop class calls the readFromFile function to read all elements from file at the creation of shop object.

e. Shopping Cart: Contains a collection of shop Items in form of array list. The array list in this class will contain the items that are to be added to cart. For permanent storage of your cart, write the complete array list of cart items to a file. On checkout items will be deleted from shop. You may also empty the shopping cart.

```
public class ShopItem implements
   Serializable {
     protected String itemID;
     protected double price;

     public ShopItem(String itemID,
   double price) {
          this.itemID = itemID;
          this.price = price;
     }
     public String getItemID() {
          return itemID;
     }
     public double getPrice() {
          return price;
     }
   @Override
    public String toString() {
         return "ShopItem{" +
  "itemID=" + itemID + ", price=" +
  price + '}';
     }
}
```

```
public class MobilePhone extends
ShopItem implements Serializable {
     private String name;
     private String color;
     private boolean hasCard;

     public MobilePhone(String name,
String color, boolean hasCard,
String itemID, double price) {
          super(itemID, price);
          this.name = name;
          this.color = color;
          this.hasCard = hasCard;
     }
     public String getName() {
          return name;
     }
    @Override
     public String toString() {
          return "MobilePhone{" +
"itemId=" + itemID  + ", name=" +
name + ", color=" + color + ",
hasCard=" + hasCard + '}';
     }
}
```

```
public class LED extends ShopItem implements Serializable {
     private String model;
     private String company;
     private double area;

     public LED(String model, String company, double area, String itemID,
double price) {
          super(itemID, price);
          this.model = model;
          this.company = company;
          this.area = area;
     }
     public String getModel() {
          return model;
     }
      @Override
     public String toString() {
          return "LED {" + "itemID=" + itemID  + ", model=" + model + ",
company=" + company + ", area=" + area + '}';
     }
}
```

```
public class Shop {
     private String filename;
     private ArrayList<ShopItem> shopList = new ArrayList<ShopItem>();

     public Shop(String filename) {
          this.filename = filename;
          readFromFile();
     }
     public ArrayList getList() {
          return shopList;
     }
     public void addItemToShop(ShopItem item) {
          shopList.add(item);
          writeToFile();
```

```
public void writeToFile() {
    try {
        File f = new File(filename);

        ObjectOutputStream oos;
        oos = new ObjectOutputStream(new FileOutputStream(f));
        oos.writeObject(shopList);
        oos.close();
    }
     catch (IOException e) {        }
}
@Override
public String toString(){
    String r="";
        for (int i = 0; i<shopList.size();i++){
        r += (shopList.get(i).getItemID()+", ") ;
}
    }
     return r;
}

    public  void readFromFile() {
        try {
            ObjectInputStream input = new ObjectInputStream(new
FileInputStream(filename));
            shopList = ( ArrayList <ShopItem> ) input.readObject();

        } catch (ClassNotFoundException c) {

        } catch (ClassCastException c) {

        } catch (EOFException e) {

        } catch (FileNotFoundException ex) {

        } catch (IOException ex) {        }
    }
    public  void deleteFromShop(String id)
    {
        for (int i=0; i<shopList.size();i++)
       {
            if (shopList.get(i).getItemID().equals(id))
            {
                shopList.remove(i);
                        break;
            }
        }
        writeToFile();
    }
}
```

Write definitions of method and constructors in following class.

**(15 marks) (60 min) [CLO 3]**

```
public class ShoppingCart  {

    private ArrayList<ShopItem> cartList=new ArrayList();
    private Shop s;
    private String cartID;
    private double totalCost;
```

```
ShoppingCart(Shop s, String id)
{
    //arugumented constructor
}
public void buyMobile(String name)
{
        //Buy mobile by NAME. Update cartList and total price.
  }
public double getTotalCost() {
 //get total cost
}
public void buyLED(String model)
{
        //Buy LED by MODEL. Update cartList and total price
}
public void cancelOrder(String itemID)
{
       //Cancel an item by itemID. Update cartList and total price.
}
public void writeCartToFile( )
{
        // Write the Cart to the File.
}
void checkOut() {
   // Remove all the bought (items in the cartList) from the shop.
 } }
```

**Question 4.** Airport management system manages the ticket counter, *flights, and notice* board for flights. Airport management have two types of employees, i.e., *airport and airlines* employees. Airport employees are associated with ticket counter, *flights, and notice board,* whereas airline employees work only with flights and ticket counter.

You need to create UML for following classes.

1. Airport management
2. Ticket Counter
3. Flight
4. Notice Board
5. Employees (airline and airport)

Each class should have a toString() and at least 3 data fields. **(10 marks) (30 min) [CLO 2]**

**Question 5.** Convert the following class to generic class. The class should accept java.lang.Number as type. Test it for any two Number types. **(5 marks) (20 min) [CLO 3]**

```
public class Test {
    private double salary;
    private double expenses;
    public Test(double salary, double expenses) {
        this.salary = salary;
        this.expenses = expenses;
    }
    public double calculatesaving() {
        return salary - expenses;
    }
    public static void main(String args[]) {
        Test t = new Test(25000, 18000);
        System.out.println(t.calculatesaving());
    } }
```