Object Oriented Programming Quiz 1

From:

AL AZHAR RIZQI RIFA'I FIRDAUS

Class:

2 I

Absence:

01

Student Number Identity:

2241720263

Department:

Information Technology

Study Program:

Informatics Engineering

Quiz 1

1. Class and Object:

What is meant by "class" in object-oriented programming?

- **Class represents a group of objects that have similar properties and behavior.**

How do you define an object of a class in the Java programming language?

- **First, we need to create class and attribute inside of class. Then we create main class to instantiate the object like below.**

Item.java × Main.java 1

src > main > java > com > azhar > object > Item.java > Item > Item(String)

```java
package com.azhar.object;

public class Item {
    public String name;

    public Item(String name) {
        this.name = name;
    }
}
```

Item.java Main.java 1 ×

src > main > java > com > azhar > object > Main.java > Main > main(String[])

```java
package com.azhar.object;

public class Main {
    Run | Debug | Codeium: Refactor | Explain | Generate Javadoc
    public static void main(String[] args) {
        Item mouse = new Item(name:"mouse");
    }
}
```

Suppose you have a class "Item" in an inventory information system. How would you create a "laptop" object from that class?

- **Same with my answer before, we should to create class and attribute first.**

```
Item.java ×   Main.java 1
src > main > java > com > azhar > object > Item.java > Item > Item(String)
1   package com.azhar.object;
2
3   public class Item {
4       public String name;
5
6       public Item(String name) {
7           this.name = name;
8       }
9   }
10
```

- **Then we create a main class in order to instantiate the laptop object.**

```
Item.java   Main.java 1 ×
src > main > java > com > azhar > object > Main.java > Main > main(String[])
1   package com.azhar.object;
2
3   public class Main {
        Run | Debug | Codeium: Refactor | Explain | Generate Javadoc
4       public static void main(String[] args) {
5           Item laptop = new Item(name:"laptop");
6       }
7   }
8
```

2. Encapsulation:

Explain the concept of encapsulation in object-oriented programming and why it is important in the development of an item inventory information system.

- Encapsulation is a way of hiding the implementation details of a class from outside access and only exposing a public interface that can be used to interact with the class.

In the context of an inventory information system, give an example of an attribute (variable) that should be encapsulated and why.

- **The example of attributes is price, amount, status. It should be encapsulated because to protect value, and to prevent data manipulation.**

3. Class Relationships:

What is meant by the relation between classes in object-oriented programming?

- **Relation means different relations between two or more classes. It can be inheritance, associations, depedency.**

In an item inventory information system, how would you describe the relationship between the class "Item" and the class "Category"?

- **The relation between Item and Category is association because items are often categorized into different categories and manage inventory effectively.**

4. PBL:

Based on the case of the inventory information system, try to create a simple class with attributes and methods that describe an entity. (e.g., class "Item" in the system (for example, class "Item").

Item.java × Main.java 1

src > main > java > com > azhar > object > Item.java > Item

```java
package com.azhar.object;

public class Item {
    public String name;
    public int amount;
    public double price;

    public Item() {

    }

    public Item(String name, int amount, double price) {
        this.name = name;
        this.amount = amount;
        this.price = price;
    }

    Codeium: Refactor | Explain | Generate Javadoc
    public void addAmount(int amount) {
        this.amount += amount;
    }

    Codeium: Refactor | Explain | Generate Javadoc
    public void subAmount(int amount) {
        if (amount < 0) {
            System.out.println("Out of stock!");
        } else {
            this.amount -= amount;
        }
    }

    Codeium: Refactor | Explain | Generate Javadoc
    public void print() {
        System.out.println("Name: " + name);
        System.out.println("Amount: " + amount);
        System.out.println("Price: " + price);
    }
}
```

-

How will you use encapsulation to protect the attributes in the class?

- **Change the modifier attribute from public to private, then add a getter setter method for each attribute.**

Item.java × Main.java 1

src > main > java > com > azhar > object > Item.java > Item > getName()

```java
package com.azhar.object;

public class Item {
    private String name;
    private int amount;
    private double price;

    public Item() {

    }

    public Item(String name, int amount, double price) {
        this.name = name;
        this.amount = amount;
        this.price = price;
    }

    Codeium: Refactor | Explain | Generate Javadoc
    public String getName() {
        return name;
    }

    Codeium: Refactor | Explain | Generate Javadoc
    public int getAmount() {
        return amount;
    }

    Codeium: Refactor | Explain | Generate Javadoc
    public double getPrice() {
        return price;
    }

    Codeium: Refactor | Explain | Generate Javadoc
    public void setName(String name) {
        this.name = name;
    }

    Codeium: Refactor | Explain | Generate Javadoc
    public void setAmount(int amount) {
        this.amount = amount;
    }
```

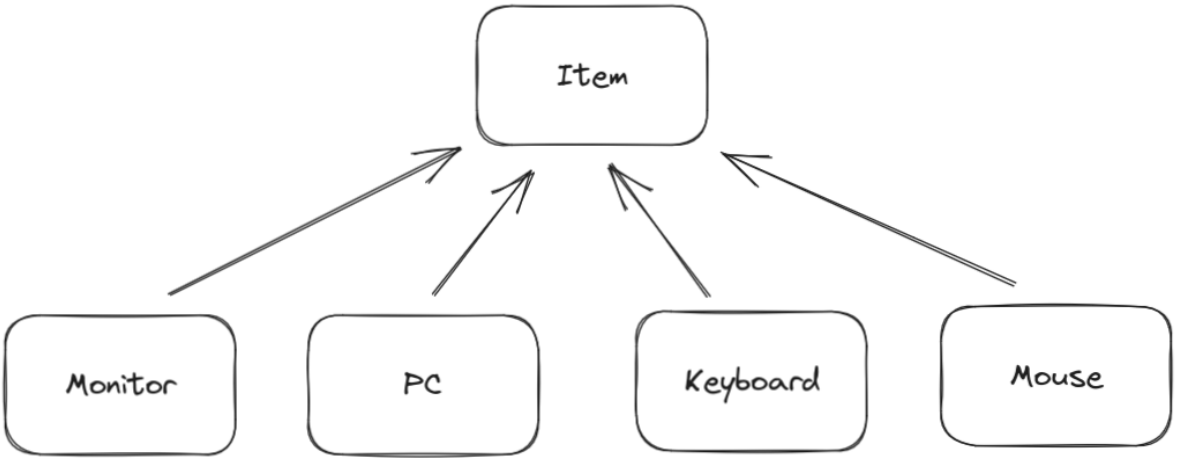
-

```java

    Codeium: Refactor | Explain | Generate Javadoc
    public void setPrice(double price) {
        this.price = price;
    }

    Codeium: Refactor | Explain | Generate Javadoc
    public void addAmount(int amount) {
        this.amount += amount;
    }

    Codeium: Refactor | Explain | Generate Javadoc
    public void subAmount(int amount) {
        if (amount < 0) {
            System.out.println("Out of stock!");
        } else {
            this.amount -= amount;
        }
    }

    Codeium: Refactor | Explain | Generate Javadoc
    public void print() {
        System.out.println("Name: " + name);
        System.out.println("Amount: " + amount);
        System.out.println("Price: " + price);
    }
}
```

Describe the class hierarchy or the relationship between classes that may exist in the system information system in the Information Technology department. Give an example of relationships between classes (for example, inheritance or association) in that context.

- **Class hierarchy**

-



- **Code**

Item.java × | PC.java | Monitor.java | Keyboard.java | Mouse.java

src > main > java > com > azhar > object > Item.java > Item > print()

```java
package com.azhar.object;

public class Item {
    private String brand;
    private int amount;
    private double price;
    private String status;

    public Item() {

    }

    public Item(String brand, int amount, double price, String status) {
        this.brand = brand;
        this.amount = amount;
        this.price = price;
        this.status = status;
    }

    Codeium: Refactor | Explain | Generate Javadoc
    public String getBrand() {
        return brand;
    }

    Codeium: Refactor | Explain | Generate Javadoc
    public String getStatus() {
        return status;
    }

    Codeium: Refactor | Explain | Generate Javadoc
    public int getAmount() {
        return amount;
    }

    Codeium: Refactor | Explain | Generate Javadoc
    public double getPrice() {
        return price;
    }

    Codeium: Refactor | Explain | Generate Javadoc
    public void setBrand(String brand) {
        this.brand = brand;
```

-

```java
    }

    Codeium: Refactor | Explain | Generate Javadoc
    public void setStatus(String status) {
        this.status = status;
    }

    Codeium: Refactor | Explain | Generate Javadoc
    public void setAmount(int amount) {
        this.amount = amount;
    }

    Codeium: Refactor | Explain | Generate Javadoc
    public void setPrice(double price) {
        this.price = price;
    }

    Codeium: Refactor | Explain | Generate Javadoc
    public void addAmount(int amount) {
        this.amount += amount;
    }

    Codeium: Refactor | Explain | Generate Javadoc
    public void subAmount(int amount) {
        if (amount < 0) {
            System.out.println("Out of stock!");
        } else {
            this.amount -= amount;
        }
    }

    Codeium: Refactor | Explain | Generate Javadoc
    public void print() {
        System.out.println("Brand: " + brand);
        System.out.println("Amount: " + amount);
        System.out.println("Price: " + price);
        System.out.println("Status: " + status);
    }
}
```

Item.java | PC.java × | Monitor.java | Keyboard.java

src > main > java > com > azhar > object > PC.java > PC > printPC()

```java
package com.azhar.object;

public class PC extends Item {
    private String cpu;
    private int ram;
    private int storage;

    public PC() {

    }

    public PC(String cpu, int ram, int storage) {
        this.cpu = cpu;
        this.ram = ram;
        this.storage = storage;
    }

    Codeium: Refactor | Explain | Generate Javadoc
    public String getCpu() {
        return cpu;
    }

    Codeium: Refactor | Explain | Generate Javadoc
    public int getRam() {
        return ram;
    }

    Codeium: Refactor | Explain | Generate Javadoc
    public int getStorage() {
        return storage;
    }

    Codeium: Refactor | Explain | Generate Javadoc
    public void setCpu(String cpu) {
        this.cpu = cpu;
    }

    Codeium: Refactor | Explain | Generate Javadoc
    public void setRam(int ram) {
        this.ram = ram;
    }
```

-

```java

    Codeium: Refactor | Explain | Generate Javadoc
    public void setStorage(int storage) {
        this.storage = storage;
    }

    Codeium: Refactor | Explain | Generate Javadoc
    public void printPC() {
        print();
        System.out.println("CPU: " + cpu);
        System.out.println("RAM: " + ram);
        System.out.println("Storage: " + storage);
    }
}
```

Item.java | PC.java | Monitor.java | Keyboard.java | M

src > main > java > com > azhar > object > Monitor.java > Monitor > printMonit

```java
package com.azhar.object;

public class Monitor extends Item {
    private String resolution;
    private int screenSize;

    public Monitor() {

    }

    public Monitor(String resolution, int screenSize) {
        this.resolution = resolution;
        this.screenSize = screenSize;
    }

    Codeium: Refactor | Explain | Generate Javadoc
    public String getResolution() {
        return resolution;
    }

    Codeium: Refactor | Explain | Generate Javadoc
    public int getScreenSize() {
        return screenSize;
    }

    Codeium: Refactor | Explain | Generate Javadoc
    public void setResolution(String resolution) {
        this.resolution = resolution;
    }

    Codeium: Refactor | Explain | Generate Javadoc
    public void setScreenSize(int screenSize) {
        this.screenSize = screenSize;
    }

    Codeium: Refactor | Explain | Generate Javadoc
    public void printMonitor() {
        print();
        System.out.println("Resolution: " + resolution);
        System.out.println("Screen Size: " + screenSize);
    }
}
```

Item.java | PC.java | Monitor.java | Keyboard.java

src > main > java > com > azhar > object > Keyboard.java > Keyboard > pri

```java
package com.azhar.object;

public class Keyboard extends Item {
    private int layout;

    public Keyboard() {

    }

    public Keyboard(int layout) {
        this.layout = layout;
    }

    Codeium: Refactor | Explain | Generate Javadoc
    public int getLayout() {
        return layout;
    }

    Codeium: Refactor | Explain | Generate Javadoc
    public void setLayout(int layout) {
        this.layout = layout;
    }

    Codeium: Refactor | Explain | Generate Javadoc
    public void printKeyboard() {
        print();
        System.out.println("Layout: " + layout + "%");
    }
}
```

-

Main.java

src > main > java > com > azhar > object > Main.java > Main > main(String[])

```java
package com.azhar.object;

public class Main {
    Run | Debug | Codeium: Refactor | Explain | Generate Javadoc
    public static void main(String[] args) {
        PC pc1 = new PC();
        pc1.setBrand(brand:"Lenovo");
        pc1.setAmount(amount:10);
        pc1.setPrice(price:1000);
        pc1.setStatus(status:"Available");
        pc1.setCpu(cpu:"i7");
        pc1.setRam(ram:8);
        pc1.setStorage(storage:500);
        pc1.printPC();

        System.out.println();

        Monitor monitor1 = new Monitor();
        monitor1.setBrand(brand:"Samsung");
        monitor1.setAmount(amount:10);
        monitor1.setPrice(price:1000);
        monitor1.setStatus(status:"Available");
        monitor1.setResolution(resolution:"4K");
        monitor1.setScreenSize(screenSize:27);
        monitor1.printMonitor();

        System.out.println();

        Keyboard keyboard1 = new Keyboard();
        keyboard1.setBrand(brand:"Apple");
        keyboard1.setAmount(amount:10);
        keyboard1.setPrice(price:1000);
        keyboard1.setStatus(status:"Available");
        keyboard1.setLayout(layout:60);
        keyboard1.printKeyboard();

        System.out.println();

        Mouse mouse1 = new Mouse();
        mouse1.setBrand(brand:"Logitech");
        mouse1.setAmount(amount:10);
```

-
```java
        mouse1.setPrice(price:1000);
        mouse1.setStatus(status:"Available");
        mouse1.setType(type:"Wireless");
        mouse1.printMouse();
    }
}
```

- **Result**

```
→ zharsuke@box  ~/Documents/College/Semester_3/oop/quiz1/coding git:(master) ✗
xceptionMessages -cp /home/zharsuke/Documents/College/Semester_3/oop/quiz1/codi
Picked up _JAVA_OPTIONS: -Dawt.useSystemAAFontSettings=on -Dswing.aatext=true
Brand: Lenovo
Amount: 10
Price: 1000.0
Status: Available
CPU: i7
RAM: 8
Storage: 500

Brand: Samsung
Amount: 10
Price: 1000.0
Status: Available
Resolution: 4K
Screen Size: 27

Brand: Apple
Amount: 10
Price: 1000.0
Status: Available
Layout: 60%

Brand: Logitech
Amount: 10
Price: 1000.0
Status: Available
Type: Wireless
→ zharsuke@box  ~/Documents/College/Semester_3/oop/quiz1/coding git:(master) ✗
```