Things To Know Assignment
-Riya Sharma

Explanation of few questions.

Question 1. Suppose you have a class called "Car" with a static variable called "numOfCars". How would you use this variable to keep track of the total number of cars created by the class?

Explanation: To keep track of the total number of cars created by the class ‘Car’, we will :
1. Declare the "numOfCars" variable inside the "Car" class with the keyword "static" and initialize it to 0.
2. We will add the increment operation "++" to the static variable in the constructor. Therefore, every time a new instance of the "Car" class is created, the constructor will increment the value of "numOfCars" by 1, effectively keeping track of the total number of cars created by the class.

3.Additionally, we can access the value of "numOfCars" outside of the class by using the class name followed by the static variable name, like so:

int totalNumOfCars = Car.numOfCars;

VariableUsageCount.java

```java
package com.knoldus.task1;

3 usages  new *
class Car{
    2 usages
    static int numOfCars = 0;

    1 usage  new *
    public Car() {
        // increment the static variable by 1 for each car object created
        numOfCars++;
    }

    1 usage  new *
    public static int getNumOfCars() {
        return numOfCars;
    }

    }
    no usages  new *
    public class VariableUsageCount{
        no usages  new *
        public static void main(String[] args) {
            Car car = new Car();
            System.out.println(Car.getNumOfCars());
        }
    }
```

Question 2. Imagine you have a static variable called "PI" in a class called "Circle". How would you use this variable to calculate the circumference of a circle?

- In the code, the Circle class has a static final variable named PI which is assigned the value 3.14159. The class also has a private instance variable radius which is set in the constructor.
- The getCircumference method calculates and returns the circumference of the circle using the formula 2 * PI * radius, where PI is accessed using the class name Circle.PI.
- Therefore, to use the static variable "PI", we would simply refer to it using the name of the class ‘Circle’.

```java
package com.knoldus.task2;


import java.util.Scanner;

2 usages  new *
class Circle {
    1 usage
    public static double PI = 3.14159;
    2 usages
    private double radius;

    1 usage  new *
    public Circle(double radius) {
        this.radius = radius;
    }

    1 usage  new *
    public double getCircumference() {
        return 2 * PI * radius;
    }
}
no usages  new *
public class CalculateCircumference {
    no usages  new *
    public static void main(String[] args) {
        Circle circle = new Circle( radius: 7.0);
        double circumference = circle.getCircumference();


    }
}
```

Question 3. Consider a class called "Employee" with a static variable called "nextId". How would you use this variable to assign a unique ID to each new employee object created?

```java
package com.knoldus.task3;

4 usages  new *
public class Employee
{
    2 usages
    static int nextId=1;
    2 usages
    int id;
    2 usages  new *
    Employee()
    {
        id = nextId;
        nextId++;      //value of id getting incremented for the next employee
    }
    2 usages  new *
    public int getId() { return id; }
    no usages  new *
    public static void main(String args[])
    {
        Employee employee1 = new Employee();
        Employee employee2 = new Employee();
        System.out.println("The id of employee 1: "+employee1.getId());
        System.out.println("The id of employee 2: "+employee2.getId());
    }
}
```

- Each time an Employee object is created, its id is set to the current value of nextId, and then nextId is incremented by 1 so that the next Employee object created will have a unique ID.
- Since nextId is a static variable, it is shared by all instances of the Employee class, and its value will persist across all Employee objects created.

Question 4. Suppose you have a class called "BankAccount" with a static variable called "interestRate". How would you use this variable to calculate the interest earned on a bank account balance?

- The BankAccount class has a private static variable named interestRate, which ican be set according to the requirement or user input.
- The class also has a private instance variable balance which is set in the constructor.
- The calculateInterest method calculates and returns the interest earned on the bank account balance using the formula balance * interestRate, **where interestRate is accessed using the class name BankAccount.interestRate.**

```java
package com.knoldus.task4;
1 usages  new *
class BankAccount {
    2 usages
    private static double interestRate = 0.01; // interest rate is set to 1%

    2 usages
    private double balance;

    1 usage  new *
    public BankAccount(double balance) {
        this.balance = balance;
    }

    2 usages  new *
    public double calculateInterest() {
        return balance * interestRate;
    }

    1 usage  new *
    public static void setInterestRate(double rate) {
        interestRate = rate;
    }
}
no usages  new *
public class CalculateInterest {
    no usages  new *
    public static void main(String[] args) {
        BankAccount account = new BankAccount( balance: 1000.0);
        double interestEarned = account.calculateInterest();
        System.out.println("Interest earned: " + interestEarned);

        BankAccount.setInterestRate(0.02); // interest rate is changed to 2%

        interestEarned = account.calculateInterest();
        System.out.println("Interest earned: " + interestEarned);
    }
}
```

Question 5. Imagine you have a class called "Student" with a static variable called "numOfStudents". How would you use this variable to display the total number of students enrolled in a course?

```java
package com.knoldus.task5;

7 usages  new *
public class Student {
    // Static variable to keep track for the number of students enrolled
    2 usages
    private static int numOfStudents = 0;

    3 usages  new *
    public Student() { numOfStudents++; }

    1 usage  new *
    public static int getNumOfStudents() { return numOfStudents; }
    no usages  new *
    public static void main(String[]args) {

        Student s1 = new Student();
        Student s2 = new Student();
        Student s3 = new Student();
        System.out.println("Total number of students enrolled: " + Student.getNumOfStudents());
    }
}
```

- We can use the static variable numOfStudents to keep track of the total number of students enrolled in a course.
- Every time a new Student object is created , the constructor is called, the numOfStudents variable is incremented by 1. Finally, to display the total number of students enrolled, you can simply access the static variable numOfStudents using the class name Student.

Question 7. Suppose you have a class called "Database" with a static variable called "numOfConnections". How would you use this variable to keep track of the number of active database connections?

- We can use the static variable numOfConnections in the Database class to keep track of the number of active database connections.
- Every time a new connection is created, the numOfConnections variable is incremented by 1 and the getNumOfConnections() method is used to retrieve the current number of active database connections.

```java
package com.knoldus.task7;

5 usages  new *
public class Database {
    // Creating Static variable for the number of active database connections
    2 usages
    private static int numOfConnections = 0;

    2 usages  new *
    public Database() {
        // Code to establish a database connection
        numOfConnections++;
    }

    1 usage  new *
    public static int getNumOfConnections() { return numOfConnections; }
    no usages  new *
    public static void main(String[] args){
        Database databaseUser1 = new Database();
        Database databaseUser2 = new Database();
        int activeConnections = Database.getNumOfConnections();
        System.out.println("Number of active database connections: " + activeConnections);
    }
}
```

Question 11. Create a Java package called "game" that contains a class called "Player". Define instance variables for the player's name, score, and level. Write methods to set and get the values of the variables. Import the package into a Java program and create some Player objects.

Main.java

```java
package com.knoldus.task11;
import game.Player;

//Importing the game package into a Java program and create some Player objects
no usages  new *
public class main {
    no usages  new *
    public static void main(String[] args) {

        Player player1 = new Player( name: "Abhinay",  score: 1000,  level: 15);
        Player player2 = new Player( name: "Virat",  score: 1500,  level: 22);

        System.out.println("Player 1: " + player1.getName() + "," + " Score: " + player1.getScore() + "," + " Level: " + player1.getLevel());
        System.out.println("Player 2: " + player2.getName() + "," + " Score: " + player2.getScore() + "," + " Level: " + player2.getLevel());

        player1.setName("Ray");
        player2.setScore(300);
        player2.setLevel(25);

        System.out.println("Player 1: " + player1.getName() + "," + " Score: " + player1.getScore() + "," + " Level: " + player1.getLevel());
        System.out.println("Player 2: " + player2.getName() + "," + " Score: " + player2.getScore() + "," + " Level: " + player2.getLevel());

    }
}
```

package game;

```java
package game;

5 usages  new *
public class Player {
        3 usages
        private String name;
        3 usages
        private int score;
        3 usages
        private int level;

        2 usages  new *
        public Player(String name, int score, int level) {
            this.name = name;
            this.score = score;
            this.level = level;
        }
    1 usage  new *
    public void setName(String name) {
        this.name = name;
    }

    4 usages  new *
    public String getName() {
        return name;
    }
    1 usage  new *
    public void setScore(int score) {
        this.score = score;
    }

    4 usages  new *
    public int getScore() {
        return score;
    }

    1 usage  new *
    public void setLevel(int level) {
        this.level = level;
    }
```

In the program we create two Player objects, sets and gets their properties, and prints out the results.

Question 12. Create a Java class with a private instance variable. Define getter and setter methods for the variable. Import the class into another Java program and use the getter and setter methods to access the variable.

Explanation : Thus the program shows how we can use getter and setter methods to access private instance variables of a class from another Java program.

```java
package com.knoldus.task12;

no usages  new *
public class PatientDetails {
    no usages  new *
    public static void main(String[] args) {
        //the getter and setter methods used to access the variable from other class
        PatientInformation patient = new PatientInformation();
        patient.setPatient_name("Rose");
        System.out.println("getting my number "+ patient.getPatient_name());
    }
}
```

```java
package com.knoldus.task12;

2 usages  new *
public class PatientInformation {
    2 usages
    private String Patient_name;

    1 usage  new *
    public String getPatient_name() {
        return Patient_name;
    }

    1 usage  new *
    public void setPatient_name(String name) {
        Patient_name = name;
    }
}
```

Question 13. Create a Java class with a protected method. Define a subclass of the class and try to call the protected method from the subclass. What happens?

Explanation : Thus, in above program, even though myProtectedMethod() is not public and cannot be accessed from outside the package, it can be accessed from a subclass within the same package. This is because protected methods are visible to subclasses and classes within the same package.

```java
package com.knoldus.task13;


2 usages  new *
public class MySubclass extends MyClass {
    1 usage  new *
    public void myPublicMethod() {
        myProtectedMethod();
    }

    no usages  new *
    public static void main(String[] args) {
        MySubclass subclassObject= new MySubclass();
        subclassObject.myPublicMethod();
    }

}
```

```java
package com.knoldus.task13;

1 usage  1 inheritor  new *
public class MyClass {
    1 usage  new *
    protected void myProtectedMethod() {
        System.out.println("This is a protected method.");
    }
}
```

Question 14. Create a Java class with a public method. Define a method in the same package that tries to access the public method. What happens?

```java
package com.knoldus.task14;


no usages  new *
public class GetName
{
    no usages  new *
    public void printName()
    {
        //can access public method of PublicMethod class in different method of same package
        PublicMethod public1 = new PublicMethod();
        System.out.println(public1.returnName());
    }
}
```

```java
ain.java ×  Employee.java ×  Student.java ×  Database.java ×  MathUtils.java ×  Calculator.java ×  GetName.java
package com.knoldus.task14;

2 usages  new *
public class PublicMethod
{
    1 usage  new *
    public String returnName() { return "john"; }
}
```

- Since, the PublicMethod class is in the same package as the GetName class, so it can access the public method printName() of the GetClass class without any issues.

- However, if the printName() was declared as private, it could not be accessed from outside the GetName class even if they were in the same package. In that case, you would need to declare the method as protected or create a public method that calls the private method to allow access to it from outside the class.

Question 15. Create a Java class with a default (package-private) instance variable. Define a method in a different package that tries to access the variable. What happens?

```
package com.knoldus.task15.Package1;

3 usages  riyaknol11 *
public class DefaultInstance
{
    no usages
    int number = 10;

}
```

```
package com.knoldus.task15.Package2;
import com.knoldus.task15.Package1.DefaultInstance;

no usages  riyaknol11 *
public class DefaultInstanceAccess
{
    riyaknol11 *
    public static void main(String args[])
    {
        DefaultInstance defaultInstance = new DefaultInstance();
        //cant access the default instance variable from a different package
        System.out.println(defaultInstance.number);
    }
}
```

Explanation :

- A compile-time error occurs. Creating a Java class with a default (package-private) instance variable and define a method in a different package that tries to access the variable, a compile-time error will occur.
- Therefore, it is not allowed to access the default instance variable from a different package.