Object Oriented Programming (Java)
October 2020

Q.NO. 1

a. Assuming that a, b, c and d are declared as below.

Int a=2, b=3, c=4;

Float d=5.0;

What are the data types and values of the following expressions?

I) (b+2)/b+2
II) B*c/d
III) a/(b/c-1)
IV) b%c*(a/d)
V) ++a+b—

Ans:-

I) (b+2)/b+2 = integer, 3

II) B*c/d = float, 2.4

III) a/(b/c-1) = integer, -2

IV) b%c*(a/d) = float, 1.2

V) ++a+b— = integer, 6

b. i. Write a for loop that prints the following output:

2 5 8 11 14

Ans:-

```
for (int i = 2; i <= 14; i += 3) {
System.out.print(i + " ");
}
```

ii. Convert the for loop in b(i) to a while loop.

Ans:-

```
int i = 2;
while (i <= 14) {
  System.out.print(i + " ");
  i += 3;
}
```

c. Write a switch statement to check the Day Number and show the Day Name based on the table below:

Day Number	Day Name
1	Monday
2	Tuesday
3	Wednesday
Other numbers	Invalid Number

Ans:-

```java
public class DayNameSwitch {
    public static void main(String[] args) {
        int dayNumber = 2;

        String dayName;

        switch (dayNumber) {
            case 1:
                dayName = "Monday";
                break;
            case 2:
                dayName = "Tuesday";
                break;
            case 3:
                dayName = "Wednesday";
                break;
            default:
                dayName = "Invalid Number";
                break;
        }

        System.out.println("Day Name: " + dayName);
    }
}
```

d. Study carefully the following program. It finds the common elements in two different integer arrays (array1 and array2) and stores the in another array called array3. At the end of the program, it prints out how many common elements there are. There are several errors in the code. Identify any five errors and fix them.

```
int array1[ ] = {1, 2, 3, 5, 8, 13, 21, 34, 55, 89};
int array2[ ] = {2, 3, 5, 7, 11, 13, 17, 19, 23, 29,};
int array3[ ];
int x, y;
for (x=0; x<10; ++x)
{
        for (y=0; y<10; ++y)
        {       if (array1[x] = array2[y])
                { array3[j] = array2[y];
                        ++n;
                }
        }
}
System.out.println("The total number of common elements" +n);
```

Ans:-

I've identified five errors in the code.

1. I added initialization for `array3` with a size of 10 to store common elements.

2. I added the variable `n` to count common elements and initialized it to 0.

3. I fixed the comparison in the `if` statement by changing `=` to `==` for proper equality comparison.

4. I used the variable `n` as the index for `array3` to store common elements.

5. I corrected the print statement by using double quotes for the string and added a space before the variable `n` for clarity.

Here's the corrected code with explanations for each error:

```java
public class Main {
    public static void main(String[] args) {
        int array1[] = {1, 2, 3, 5, 8, 13, 21, 34, 55, 89};
        int array2[] = {2, 3, 5, 7, 11, 13, 17, 19, 23, 29};
        int array3[] = new int[10]; // Error 1: Initialize array3 with a size.
        int n = 0; // Error 2: Initialize n to count common elements.

        for (int x = 0; x < 10; ++x) {
            for (int y = 0; y < 10; ++y) {
                if (array1[x] == array2[y]) { // Error 3: Use '==' for comparison, not '='.
                    array3[n] = array2[y]; // Error 4: Use 'n' as the index for array3.
                    ++n;
                }
            }
        }

        System.out.println("The total number of common elements: " + n); // Error 5: Corrected the print statement.
    }
}
```

Q.NO. 2

a. Write a class called Cycle with the following description.

Instance variables/data members:

- **model;**
- **year;**
- **price;**

Member methods:

- **default constructor;**
- **parameterized constructor;**
- **void input() – to input and store the title, author and price;**
- **void discount() – to reduce the cycle price by 10%**
- **void display() – to display the cycle model, year and price.**

Ans:-

```java
import java.util.Scanner;

class Cycle {
    private String model;
    private int year;
    private double price;

    public Cycle() {
        model = "";
        year = 0;
        price = 0.0;
    }

    public Cycle(String model, int year, double price) {
        this.model = model;
        this.year = year;
        this.price = price;
    }

    public void input() {
        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter the model: ");
        model = scanner.nextLine();

        System.out.print("Enter the year: ");
        year = scanner.nextInt();

        System.out.print("Enter the price: ");
        price = scanner.nextDouble();
    }
```

```
    public void discount() {
        price -= (0.10 * price);
    }

    public void display() {
        System.out.println("Cycle Model: " + model);
        System.out.println("Year: " + year);
        System.out.println("Price: $" + price);
    }

    public static void main(String[] args) {
        Cycle cycle1 = new Cycle();
        cycle1.input();
        cycle1.display();
        cycle1.discount();
        System.out.println("Price after 10% discount:");
        cycle1.display();

        Cycle cycle2 = new Cycle("Mountain Bike", 2022, 500.0);
        cycle2.discount();
        System.out.println("Price after 10% discount:");
        cycle2.display();
    }
}
```

b. Write a main method to create an object of the class Cycle and call any one the above member methods.

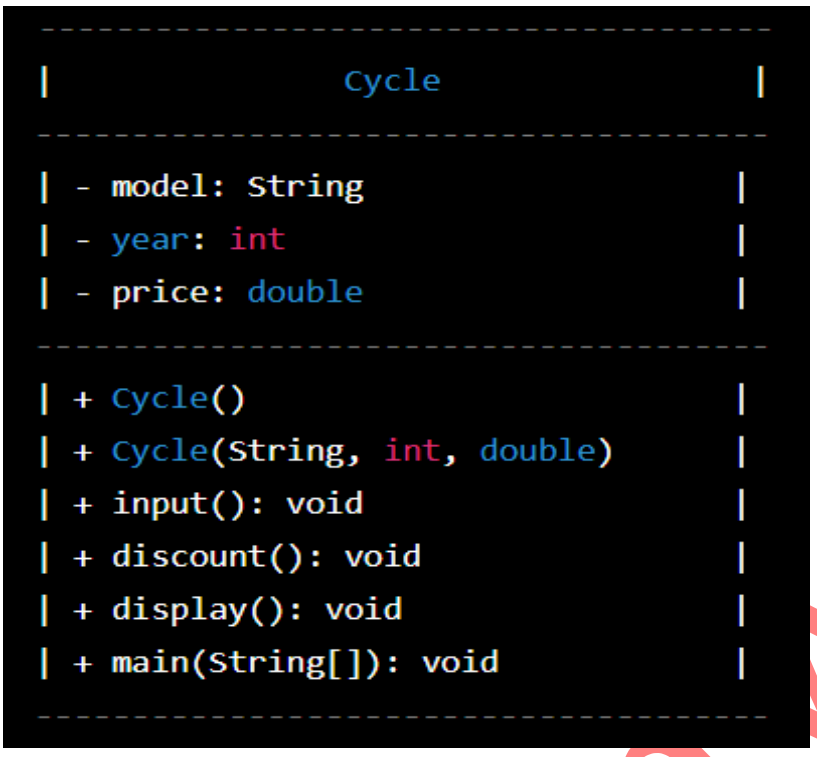
Ans:-

```
public static void main(String[] args) {
    Cycle cycle1 = new Cycle();
    cycle1.input();
    cycle1.display();
    cycle1.discount();
    System.out.println("Price after 10% discount:");
    cycle1.display();

    Cycle cycle2 = new Cycle("Mountain Bike", 2022, 500.0);
    cycle2.discount();
    System.out.println("Price after 10% discount:");
    cycle2.display();
}
```

c. Draw the UML class diagram for the above Cycle class.

Ans:-



Q.NO. 3

a. What will be the output of the following code?

```
int i = 0;
int s = 1;
int [] a = {10, 20, 30, 40, 50, 60};
 for (i=0;i<a.length;i++){
        if (a[i]%3==0){
        s=s+(a[i]/s);
       a[i]++;
    }
System.out.println(s);
}
```

Ans:-

1

1

31

31

61

61

b. Write java code that declares a 2-dimensional array of integers with four rows and five columns. Use a nested for loop to assign values to the array which is equal to row index multiply by column index.

Ans:-

```java
public class TwoDimensionalArrayExample {
    public static void main(String[] args) {
        int[][] array = new int[4][5];

        for (int i = 0; i < 4; i++) {
            for (int j = 0; j < 5; j++) {
                array[i][j] = i * j;
            }
        }

        for (int i = 0; i < 4; i++) {
            for (int j = 0; j < 5; j++) {
                System.out.print(array[i][j] + " ");
            }
            System.out.println();
        }
    }
}
```

c. Write a method in Java to compute the wages of an hourly employee. The method should take the number of hours worked, the pay rate and the overtime rate of this employee. The method should return the amount of the pay. The computation of wages should include the fact that employees who work more than 40 hours get more than the normal rate for overtime pay. You only need to write the method not a full class or calling method.

Ans:-

```java
public static double calculateWages(double hoursWorked, double payRate, double overtimeRate) {
    double regularWages;
    double overtimeWages;

    if (hoursWorked <= 40) {
        regularWages = hoursWorked * payRate;
        overtimeWages = 0;
    } else {
        regularWages = 40 * payRate;
        double overtimeHours = hoursWorked - 40;
        overtimeWages = overtimeHours * overtimeRate;
    }

    double totalWages = regularWages + overtimeWages;
    return totalWages;
}

```

d. Explain the difference between a get method and a set method. Give an example declaration of each.

Ans:-

Aspect	Get Method (Getter)	Set Method (Setter)
Purpose	Retrieves the value of an attribute	Sets the value of an attribute
Access Modifier	Typically public	Typically public
Name	Usually prefixed with "get"	Usually prefixed with "set"
Parameters	None	Accepts a parameter (new value)
Return Type	Returns the value of the attribute	Usually void (no return value)

Q.NO. 4

a. Write a java program named "TotalHours.java" to calculate the total hours worked by all employees in a department. The program should read the hours worked by each employee from an input file called "hours.txt". It should then calculate and write the total hours worked by all employee into an output file called "hoursout.txt".

Ans:-

```java
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;

public class TotalHours {
    public static void main(String[] args) {
        String inputFileName = "hours.txt";
        String outputFileName = "hoursout.txt";

        try {
            BufferedReader reader = new BufferedReader(new FileReader(inputFileName));

            double totalHoursWorked = 0.0;

            String line;
            while ((line = reader.readLine()) != null) {
                double hoursWorked = Double.parseDouble(line);
                totalHoursWorked += hoursWorked;
            }

            reader.close();

            BufferedWriter writer = new BufferedWriter(new FileWriter(outputFileName));
            writer.write("Total Hours Worked by All Employees: " + totalHoursWorked);

            writer.close();

            System.out.println("Total hours worked by all employees has been written to " + outputFileName);
        } catch (IOException e) {
            System.err.println("An error occurred: " + e.getMessage());
        }
    }
}
```

b. Give any two reasons for java.io.FileNotFoundException to be thrown at run-time.

Ans:- In Java, the `java.io.FileNotFoundException` is an exception that is thrown at runtime when there are issues related to file operations. Here are two common reasons for this exception to be thrown:

1. File Not Found: This is the most common reason for `FileNotFoundException`. It occurs when you try to access a file that does not exist at the specified path. For example:

```
FileInputStream fileInputStream = new FileInputStream("nonexistentfile.txt");
```

If the file "nonexistentfile.txt" does not exist in the specified location, Java will throw a `FileNotFoundException`.

2. Incorrect File Path: Another reason for this exception is providing an incorrect file path. If the path you specify does not point to a valid file location, Java will be unable to find the file, and a `FileNotFoundException` will be thrown. For instance:

```
FileInputStream fileInputStream = new FileInputStream("C:\\InvalidPath\\file.txt");
```

If "C:\\InvalidPath\\file.txt" does not exist or is not accessible, a `FileNotFoundException` will be raised.

c. Describe what modular programming is. Explain how Java support modular programming by using suitable examples.

Ans:- Modular programming is a software development approach where a complex system is divided into smaller, independent modules or components, each responsible for a specific function or task. These modules can be developed and tested separately, making it easier to manage and maintain large codebases.

Java supports modular programming through various mechanisms, with one of the most prominent being the Java Platform Module System (JPMS), introduced in Java 9. Here's a brief explanation with an example:

1. Java Platform Module System (JPMS): JPMS allows you to encapsulate classes and resources into modules, which are self-contained units of code. Modules can declare their dependencies and expose only what is necessary for other modules to use.

Example: Suppose you're building a Java application for a library management system. You can create separate modules for different functionalities:

- **`library.core` module:** Contains core classes and interfaces for managing books, users, and transactions.

- **`library.ui` module:** Handles the user interface and interacts with `library.core`.

- **`library.database` module:** Manages database interactions, encapsulating database-specific code.

2. Packages and Classes: Even without JPMS, Java supports modularity through packages and classes. You can organize related classes into packages, which act as modules at a smaller scale. By defining clear interfaces and access modifiers (e.g., public, private, protected), you can control the visibility and accessibility of classes and their members.

Example: In a banking application, you can create a `com.bank.accounts` package containing classes for various account types (e.g., `SavingsAccount`, `CheckingAccount`). You can then control access to these classes to ensure that only authorized parts of the code can interact with them.

d. In your opinion give a reason why is it necessary to decompose a large module into multiple sub modules.

Ans:- Decomposing a large module into multiple sub-modules is necessary for several reasons:

1. Modularity: Breaking down a large module into smaller sub-modules promotes modularity in software design. Each sub-module can focus on a specific functionality or feature, making the codebase more organized and easier to maintain.

2. Readability: Smaller sub-modules are typically easier to read and understand than a large, complex module. This improves code comprehensibility, making it simpler for developers to work with and debug.

3. Reusability: Sub-modules can often be reused in different parts of an application or in other projects. This reduces duplication of code and promotes a more efficient development process.

4. Scalability: When new features or changes are needed, it's easier to scale and adapt the software by working on smaller, self-contained sub-modules. This agility helps in accommodating evolving requirements.

5. Collaboration: Decomposing a module allows multiple developers to work on different sub-modules simultaneously. This parallel development can accelerate project completion and encourage collaboration within a development team.