**SVKM’s NMIMS**

**Mukesh Patel School of Technology Management & Engineering**

**Computer Engineering Department**

Program: BTech Integrated, Semester IV

**Course: Java Programming**

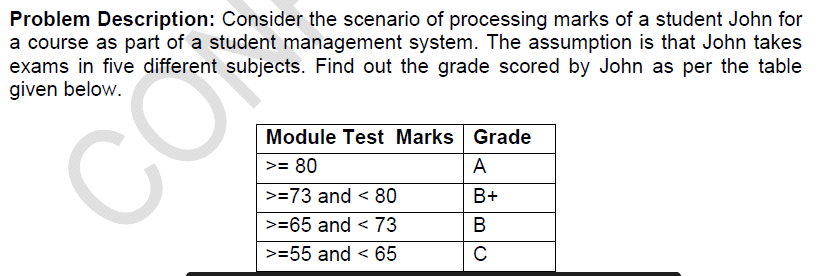
**Experiment No.02**

PART A

(PART A : TO BE REFFERED BY STUDENTS)

**A.1 Aim:** To understand and implement concepts like control statements and arrays in Java.

**TASK 1:**

****

**Task 2: Write a Java program print “Hello” for *n* times.**

**Task 3: Write a Java program that prints multiplication table of a number given by the user.**

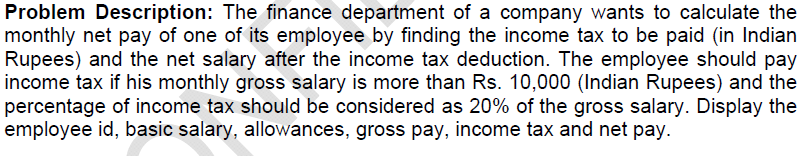
**Task 4: Write a Java program to find factorial of a number entered by the user.**

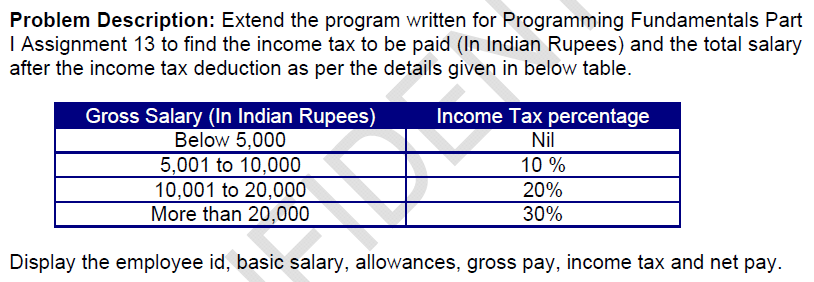
**Task 5: Write a Java program to print the sum of series as below. The user will give value of n.**

**Sum1= 1 + 1/2 + 1/3 + …… + 1/n**

**Sum2 = 1+ 22 + 32 + 42 + ……. nn**

**Task 6:**





**Task 7: Write a Java program to store marks of 10 students and display them.**

**Task 8: Write a Java program to find sum and average of marks stored in an array in task 7.**

**Task 9: Write a Java program to store marks of five students for three subjects. Find maximum marks scored by each student.**

**Task 10: Write a Java program to print “Hello” for n times. The user will provide value of n. However, the program will stop printing “Hello” after printing for 5 times.**

**A.2 Prerequisite:**

1. Fundamental concepts of C\C++.

2. Control statements in Java

**A.3 Outcome:**

**After successful completion of this experiment, students will be able to**

1. Implement concept of different types of control statements in Java

**A.4 Theory:**

**A.4.1.**

Java Control Statements:

Java compiler executes the code from top to bottom. The statements in the code are executed according to the order in which they appear. However, [Java](https://www.javatpoint.com/java-tutorial) provides statements that can be used to control the flow of Java code. Such statements are called control flow statements. It is one of the fundamental features of Java, which provides a smooth flow of program.

Java provides three types of control flow statements.

1. Decision Making statements
   * if statements
   * switch statement
2. Loop statements
   * do while loop
   * while loop
   * for loop
   * for-each loop
3. Jump statements
   * break statement
   * continue statement

Java Arrays:

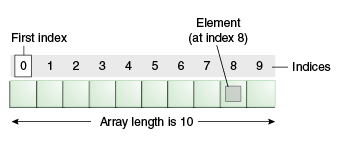
Normally, an array is a collection of similar type of elements which has contiguous memory location.

**Java array** is an object which contains elements of a similar data type. Additionally, The elements of an array are stored in a contiguous memory location. It is a data structure where we store similar elements. We can store only a fixed set of elements in a Java array.

Array in Java is index-based, the first element of the array is stored at the 0th index, 2nd element is stored on 1st index and so on.

Unlike C/C++, we can get the length of the array using the length member. In C/C++, we need to use the sizeof operator.

In Java, array is an object of a dynamically generated class. Java array inherits the Object class, and implements the Serializable as well as Cloneable interfaces. We can store primitive values or objects in an array in Java. Like C/C++, we can also create single dimentional or multidimentional arrays in Java. Moreover, Java provides the feature of anonymous arrays which is not available in C/C++.



PART B

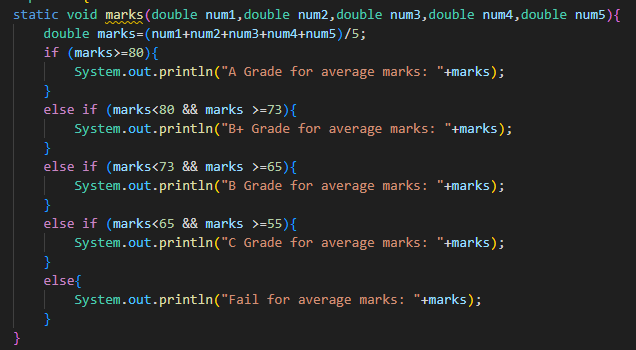
(PART B : TO BE COMPLETED BY STUDENTS)

Students must submit the soft copy as per following segments within two hours of the practical. The soft copy must be uploaded on the portal at the end of the practical. The filename should be B**DS\_batch\_rollno\_experimentno Example: BTI\_BDS\_C1\_C001\_Exp1**

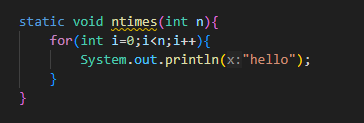
|  |  |
| --- | --- |
| Roll No. | Name: |
| Class : | Batch : |
| Date of Experiment: | Date of Submission |
| Grade : |  |

**B.1 Software Code written by student:**

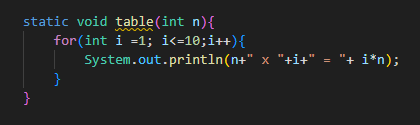
**Task 1:**

****

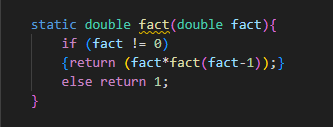
**Task 2:**

****

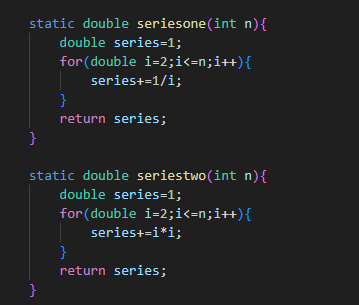
**Task 3:**

****

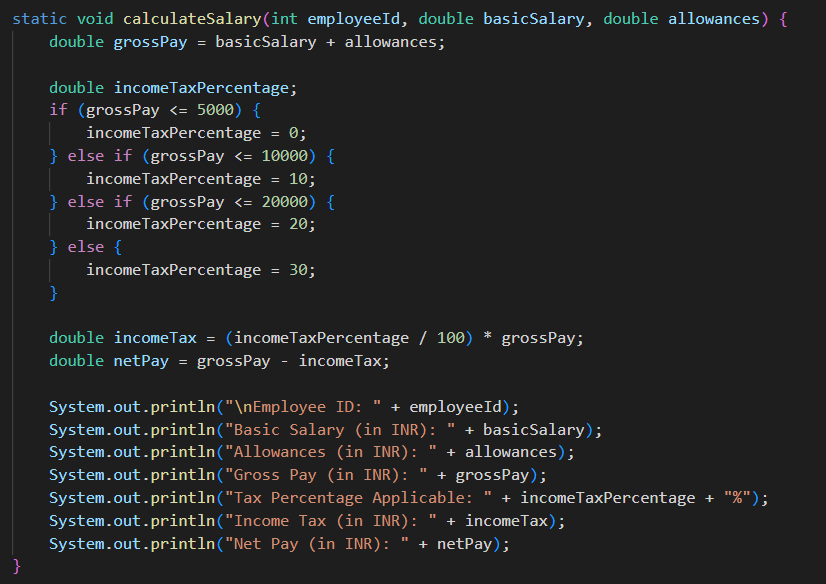
**Task 4:**

****

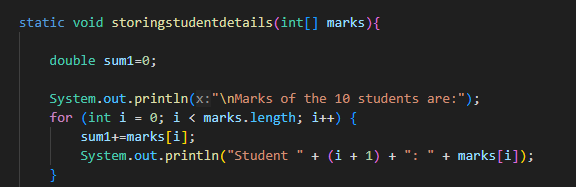
**Task 5:**

****

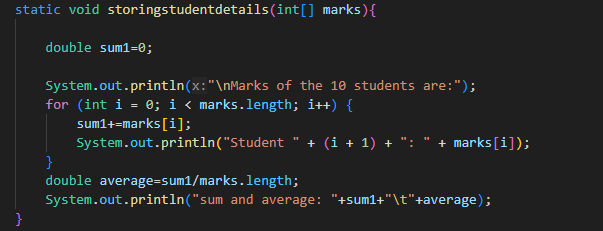
**Task 6:**

****

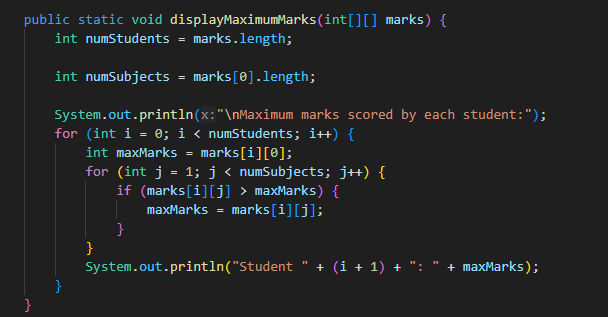
**Task 7:**

****

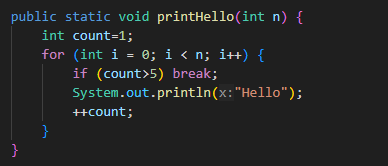
**Task 8:**

****

**Task 9:**

****

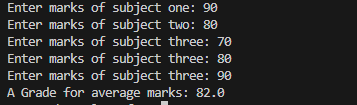
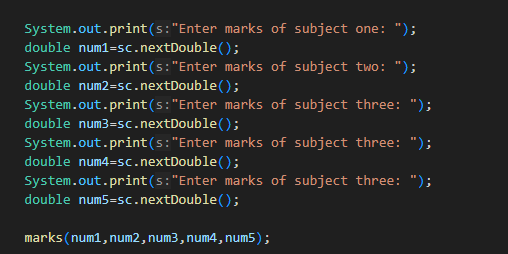
**Task 10:**

****

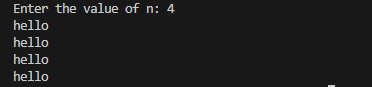
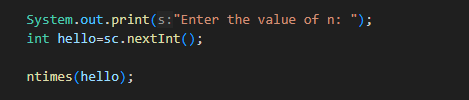
**B.2 Input and Output:**

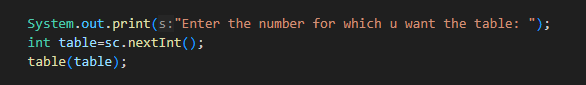
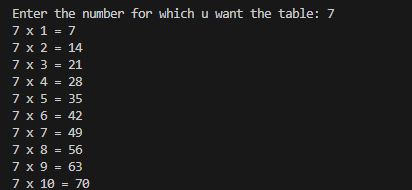
***(Paste your program input and output in following format, If there is error then paste the specific error in the output part. In case of error with due permission of the faculty extension can be given to submit the error free code with output in due course of time. Students will be graded accordingly.)***

**Task 1:**

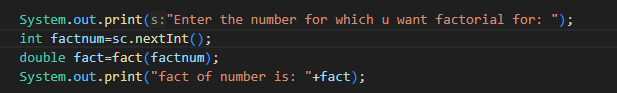
****

**Task 2:**

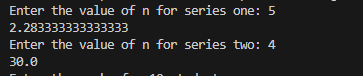
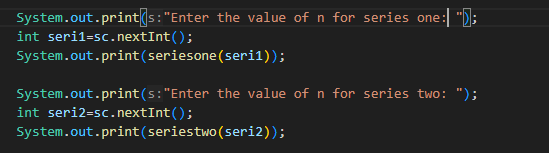
****

**Task 3:** ****

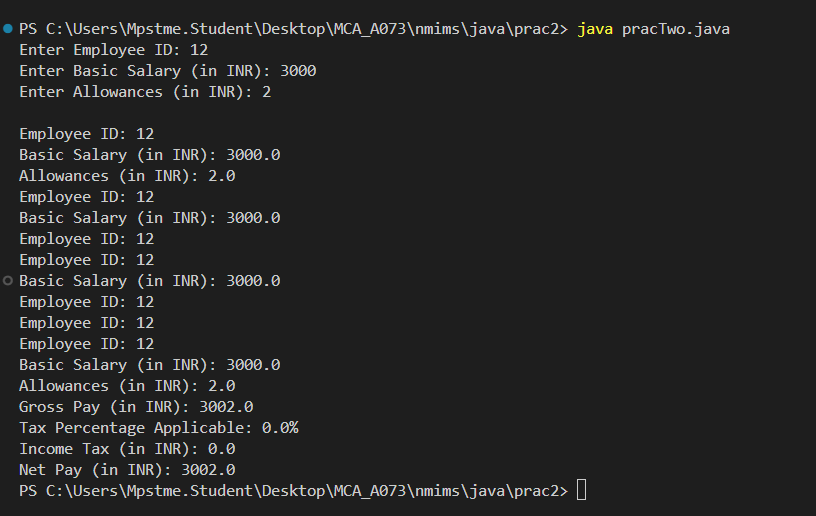
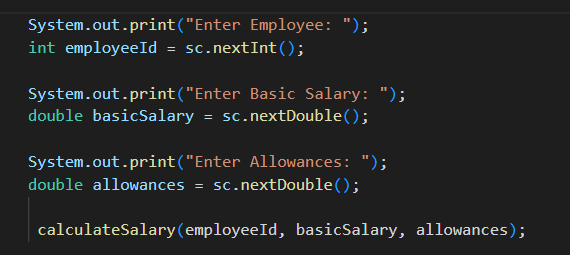
**Task 4:**

****

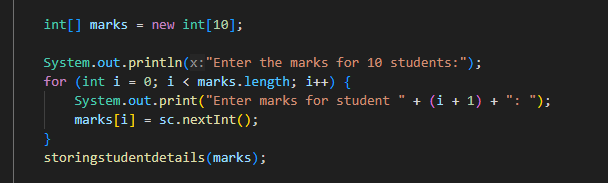
**Task 5:**

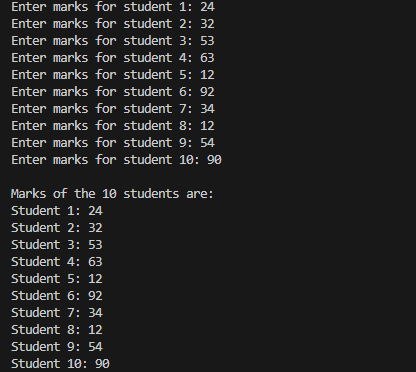


**Task 6:**

****

**Task 7:**

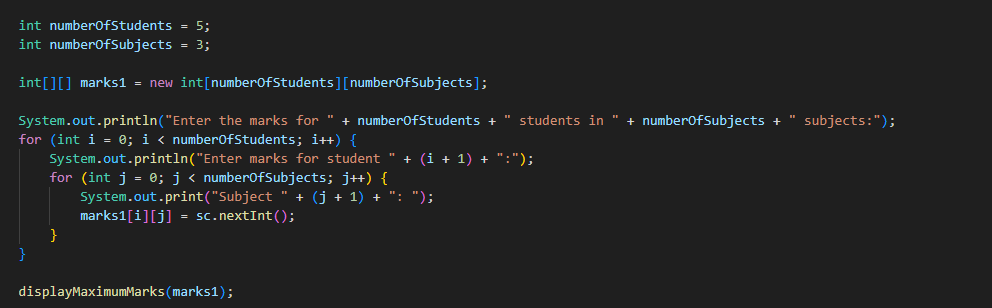
****

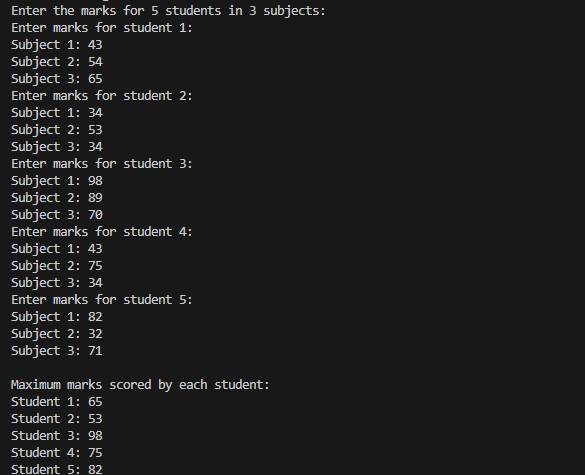
****

**Task 8:**

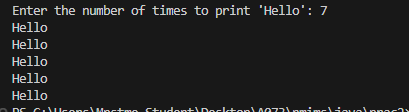
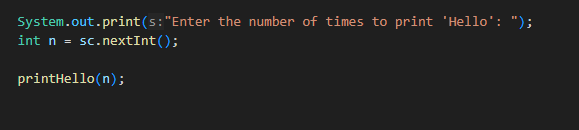
****

**Task 9:**

****

****

**Task 10:**

****

**B.3 Question of Curiosity:**

1. **Explain difference between while loop and do-while loop.**

A while loop is a control flow statement that repeatedly executes a block of statements as long as a condition is true. The condition is evaluated before entering the loop body. If the condition is initially false, the loop body will never execute.

int i = 1;

while (i <= 5) {

System.out.println(i);

i++;

}

A do-while loop also repeatedly executes a block of statements, but it first executes the block and then checks the condition. This means the loop body executes at least once, even if the condition is false initially.

int j = 1;

do {

System.out.println(j);

j++;

} while (j <= 5);

1. **Explain if-else ladder with an example.**

An if-else ladder is a series of if and else if statements in which each condition is checked sequentially until a true condition is found, and the corresponding block of statements is executed

int score = 85;

if (score >= 90) {

System.out.println("Excellent! You scored an A grade.");

} else if (score >= 80) {

System.out.println("Very good! You scored a B grade.");

} else if (score >= 70) {

System.out.println("Good! You scored a C grade.");

} else if (score >= 60) {

System.out.println("Fair! You scored a D grade.");

} else {

System.out.println("Sorry, you failed. Better luck next time.");

}

**B.3 Conclusion:**

* **Control Flow:** The use of if-else ladders allows for clear decision-making based on conditions, such as categorizing average marks into grades (A, B+, B, C, Fail).
* **Loops:** For loops are utilized in methods like ntimes to print a message repeatedly and in table to generate multiplication tables. The code also shows recursion with the fact method, which calculates the factorial of a number by calling itself.
* **Series Calculations:** The code computes mathematical series, including the harmonic series and the sum of squares showcasing iterative methods for numerical analysis.
* **Array and Matrix Handling:** Operations on single-dimensional arrays, like calculating and displaying student marks, and two-dimensional arrays, such as finding maximum marks per student, are demonstrated. This illustrates how to manage and process collections of data.
* **Arithmetic Operations:** The calculateSalary method performs arithmetic calculations and applies conditional logic to compute employee salaries, integrating tax deductions based on gross pay.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*