**Lab Tasks**

Marks : 10, All Questions carry equal marks

**Exercise 1 CalculateBMI.java**

Write a Java application with the following prototypes that returns the user's body mass index (BMI)

**public static double calcluateBMI(double weight, double height)**

To calculate BMI based on weight in pounds (lb) and height in inches (in), use this formula:

 and

**public static String findStatus(double bmi)**

Categorizes it as underweight, normal, overweight, or obese, based on the table from the United States Centers for Disease Control:

|  |  |
| --- | --- |
| **BMI** | **Weight Status** |
| Below 18.5 | Underweight |
| 18.5 – 24.9 | Normal |
| 25.0-29.9 | Overweight |
| 30.0 and above | Obese |

Prompt the user to enter weight in pounds and height in inches.

=============================Task 01===============================

import java.util.Scanner;

class CalculateBMI{

public static double calculateBMI(double height, double weight)

{

double bmi = 703\*(weight/(height\*height));

if (bmi < 18.5)

{

System.out.println("Under Weight BMI is " + bmi);

}

else if (bmi >=18.5 && bmi <= 24.9 )

{

System.out.println("Normal BMI is " + bmi);

}

else if (bmi >=25.0 && bmi <= 29.9 )

{

System.out.println("Over Weight BMI is " + bmi);

}

else if (bmi >=30.0 )

{

System.out.println("Obese BMI is " + bmi);

}

return bmi;

}

public static void main(String arg[]){

Scanner user\_input =new Scanner(System.in);

System.out.print("Enter Height : ");

double height = user\_input.nextDouble();

System.out.print("Enter Weight : ");

double weight = user\_input.nextDouble();

calculateBMI(height, weight);

}

}



===================================Task 02==========================

**Exercise 2 (Method Overloading) PrintTest.java**

Create a class to print an integer and a character with two methods having the same name but different sequence of the integer and the character parameters.  
For example, if the parameters of the first method are of the form (int n, char c), then that of the second method will be of the form (char c, int n).

import java.util.Scanner;

public class PrintTest

{

static void meth(int num, char alp)

{

System.out.println("Number"+ num + " char ="+ alp);

}

static void meth( char alp, int num)

{

System.out.println("Number"+ num + " char ="+ alp);

}

public static void main(String [] arg )

{

Scanner input = new Scanner(System.in);

int num = 78;

char alp = 'a';

meth(num, alp);

meth( alp, num);

}

}



**Exercise 3 (Static Method) FindLastDigit.java**

Write a static method named lastDigit that returns the last digit of an integer. For example, lastDigit(3852) should return 2

import java.util.Scanner;

public class FindLastDigit

{

static int retnum(int num[])

{

System.out.println(num[3]);

return num[3];

}

public static void main(String arg[])

{

Scanner input = new Scanner(System.in);

int return\_num[] = new int[4];

int num[] = new int[4];

for (int i = 0; i<4 ; i++ ){

num[i] = input.nextInt();

return\_num[i] = num[i];

}

retnum(return\_num);

}

}



**Exercise 4 (Recursion) GCDCalculation.java**

Find Greatest Common Divisor (GCD) of 2 numbers using recursion. Numbers should be taken as an input from user.

import java.util.Scanner;

public class GCDCalculation

{

public static void main(String args[])

{

Scanner scanner = new Scanner(System.in);

System.out.print("Please enter first number to find GCD : ");

int number1 = scanner.nextInt();

System.out.print("Please enter second number to find GCD : ");

int number2 = scanner.nextInt();

System.out.println("GCD of two numbers " + number1 +" and " + number2 +" is :" + findGCD(number1,number2));

}

private static int findGCD(int number1, int number2)

{

if(number2 == 0)

{

return number1;

}

return findGCD(number2, number1%number2);

}

}



**Exercise 5 (Recursion) StrReverse.java**

Write a recursive function that, given a string s= “OOP is Fun”, print the characters of s in reverse order

public class StrReverse {

public static void main(String args[])

{

String s = "OOPs is fun";

byte[] s\_reverse = s.getBytes();

byte[] res = new byte[s\_reverse.length];

for (int i = 0; i < s\_reverse.length; i++)

{

res[i] = s\_reverse[s\_reverse.length - i - 1];

}

System.out.println(new String(res));

}

}



|  |
| --- |
| **Post lab questions to ponder** |

* Can constructors be static in java? Try it out and justify.
* **Answer:** We know static keyword belongs to a class rather than the object of a class. A constructor is called when an object of a class is created, so no use of the static constructor. We can’t use the static constructor.
* Why use iterations when we have recursion and vice versa?
* **Answer:** A program is called recursive when an entity calls itself. A program is call iterative when there is a loop (or repetition). Finding the Time complexity of Recursion is more difficult than that of Iteration. Usage of either of these techniques is a trade-off between time complexity and size of code. If time complexity is the point of focus, and number of recursive calls would be large, it is better to use iteration. However, if time complexity is not an issue and shortness of code is, recursion would be the way to go. Infinite Repetition in recursion can lead to CPU crash but in iteration, it will stop when memory is exhausted.
* Can we overload by return type?
* **Answer:** No, we can’t overload by return type because we know that method overloading can only be done if both methods have the same return type and name but different parameters (i.e., the number of the parameters, the order of the parameters, and data types of the parameters) within the same class. Return type doesn’t affect method overloading.
* Can you have static classes in java? if yes then justify?
* **Answer:** The answer is Yes; some classes can be made static in Java. Java supports [Static Instance Variables](https://www.geeksforgeeks.org/static-keyword-in-java/), [Static Methods](https://www.geeksforgeeks.org/static-keyword-in-java/), [Static Block](https://www.geeksforgeeks.org/g-fact-79/) and Static Classes Java allows a class to be defined within another class. These are called **Nested Classes**. The class in which the nested class is defined is known as the **Outer Class**.

**END**