Subject: SBL-OOPJ **Class: SE-Data Science**
Semester: III **A.Y. 2022-2023**

Experiment No. 4

❖ **Aim :** Write a Java program to demonstrate method and constructor overloading.

❖ **Theory :**

- **Method Overloading :**

Method Overloading is a feature that allows a class to have more than one method with the same name, if their argument lists are different. This is an important feature in java, there are several cases where we need more than one methods with same name, for example if we are building an application for calculator, we need different variants of add method based on the user inputs such as add(int, int), add(float, float) etc. Writing more than one method within a class with a unique set of arguments is called as method overloading. All methods must share the same name. An overloaded method if not static can only be called after instantiating the object as per the requirement. Overloaded method can be static, and it can be accessed without creating an object. An overloaded method can be final to prevent subclasses to override the method. An overloaded method can be private to prevent access to call that method outside of the class.

Three ways to overload a method

In order to overload a method, the argument lists of the methods must differ in either of these:

1. Number of parameters.

For example: This is a valid case of overloading

add(int, int)

add(int, int, int)

2. Data type of parameters.

For example:

add(int, int)

add(int, float)

3. Sequence of Data type of parameters.

For example:

add(int, float)

add(float, int)

Invalid case of method overloading:

Argument list doesn't mean the return type of the method, for example if two methods have same name, same parameters and have different return type, then this is not a valid method overloading example. This will throw a compilation error.

int add(int, int)

float add(int, int)

Example:

```
class DisplayOverloading
{    public void disp(char c)
   {        System.out.println(c);   }
   public void disp(char c, int num)
   {        System.out.println(c + " "+num);  }
}
class Sample
{   public static void main(String args[])   {
      DisplayOverloading obj = new DisplayOverloading();
      obj.disp('a');
      obj.disp('a',10);
  } }
```

Output:

a

a 10

- **Constructor Overloading:**

In Java, we can overload constructors like methods. The constructor overloading can be defined as the concept of having more than one constructor with different parameters so that every constructor can perform a different task. Writing more than 1 constructor in a class with a unique set of arguments is called as Constructor Overloading. All constructors will have the name of the class. Overloaded constructor will be executed at the time of instantiating an object. An overloaded constructor cannot be static as a constructor relates to the creation of an object. An overloaded constructor cannot be final as constructor is not derived by subclasses it won't make sense

An overloaded constructor can be private to prevent using it for instantiating from outside of the class. Consider the following Java program, in which we have used different constructors in the class.

```
public class Student {
int id;
String name;
Student(){
System.out.println("this a default constructor");
}
Student(int i, String n){
id = i;
name = n;
}
public static void main(String[] args) {
Student s = new Student();
System.out.println("\nDefault Constructor values: \n");
System.out.println("Student Id : "+s.id + "\nStudent Name : "+s.name);
System.out.println("\nParameterized Constructor values: \n");
Student student = new Student(10, "Jerry");
System.out.println("Student Id : "+student.id + "\nStudent Name : "+student.name);
} }
```

Output:

```
this a default constructor


Default Constructor values:


Student Id : 0
Student Name : null


Parameterized Constructor values:


Student Id : 10
Student Name : Jerry
```

- **Conclusion :**
  - Summarize what you understood from this lab.