|  |  |
| --- | --- |
| *A close up of a logo  Description automatically generated* | *DEPARTMENT OF COMPUTER ENGINEERING* |

|  |  |
| --- | --- |
| Semester | S.E. Semester III – Computer Engineering |
| Subject | Object Oriented Programming Using Java (Skill Based Lab) |
| Subject Professor In-charge | Prof. Indu Anoop |
| Laboratory | Online Lab |

|  |  |  |
| --- | --- | --- |
| Student Name | Trisha Shah | |
| Roll Number | 20102A0004 | |
| Grade and Subject Teacher’s Signature |  |  |

|  |  |  |
| --- | --- | --- |
| Experiment | 4 | |
| Problem Statement | Write a java program demonstrate the Method overloading and Constructor Over loading | |
| Resources / Apparatus Required | Hardware: Computer System | Software: jdk 1.8, Eclipse / Notepad++/IntelliJ IDEA |
| Details | **Method Overloading**: Overloading allows different methods to have the same name, but different signatures where the signature can differ by the number of input parameters or type of input parameters or both. Overloading is related to compile-time (or static) polymorphism. Method Overloading is a feature that allows a class to have more than one method having the same name, if their argument lists are different. It is similar to constructor overloading in Java that allows a class to have more than one constructor having different argument lists.  **Constructor Overloading:** Constructor overloading is a technique in which a class can have any number of constructors that differ in parameter list. The compiler differentiates between constructors by taking into account the number of parameters in the list and their type. | |
| Code | public class Sum {      //METHOD OVERLOADING          int add(int x,int y){              System.out.println("add method with two int parameters/arguments called");              return(x+y);          }          //Invalid: Method overloading in this as return type of method must be same if same no ,type and sequence of parameters          //float add(int x,int y){          //  System.out.println("add method with two int parameters/arguments called");          //  return(x+y);          //}          int add(int x,int y,int z){              System.out.println("overloaded add method with three int parameters/arguments called");              return(x+y+z);          }          double add(double x,double y){              System.out.println("overloaded add method with same number of parameters/arguments called, but different datatype");              return(x+y);          }          double add(int x,double y){              System.out.println("overloaded add method with change in only sequence of the parameters:int,double");              return(x+y);          }              double add(double x,int y){              System.out.println("overloaded add method with change in only sequence of the parameters:double,int");           return (x+y);          }        //CONSTRUCTOR OVERLOADING          Sum(){            }  Sum(int a,int b){              System.out.println("constructor with parameters called a="+a+" b="+b);          }          Sum(int a,int b,int c){              System.out.println("overloaded constructor with diff number of parameters called a="+a+" b="+b+" c="+c);          }          public static void main(String[] args) {                Sum object=new Sum();              //overloaded method is called              System.out.println(object.add(10, 20)); System.out.println(object.add(10, 20,30));              System.out.println(object.add(10.01, 20.02));              System.out.println(object.add(10.01, 20)); System.out.println(object.add(20,10.01));              System.out.println(object.add(10.01f, 20));//type promotion                //overloaded constructor is called              Sum object1=new Sum(20,20);              Sum object2=new Sum(20,20,30);          }  } | |
| Output |  | |
| Conclusion | Through this experiment we were able to successfully demonstate method overloading and constructor overloading. | |