1 . Implement Addition and Subtraction methods to demonstrate overloaded methods and constructors.

2.What restrictions are placed on method overloading?

3.How are this() and super() used with constructors?

4.What is method overloading?

5.Write a java program for subsidy calculation for citizen given by government during floods. The attributes for citizen name, address, adhar no, subsidy amount. There are three types normal citizen, senior citizen, physically challenged citizen. Senior citizen has further attribute age. subsidy is calculated based on Age range 65 to 75 10000 Rs, 75 and above 15000Rs.For , physically challenged citizen further attribute added level of physically challenged values are 1,2 or 3. . subsidy is calculated based on level .For Level 1 25000 Rs Level 2 40000 Rs Level 3 50000 Rs .For normal citizen flat 5000 Rs .Implement runtime polymorphism by using dynamic dispatch method calculateSubsidy() being abstract method in Citizen class.And Use respective Constructors.

ANSWERS:

1.

```
class Adder{
static int add(int a,int b){return a+b;}
static int add(int a,int b,int c){return a+b+c;}
}
class TestOverloading1{
public static void main(String[] args){
System.out.println(Adder.add(11,11));
System.out.println(Adder.add(11,11,11));
}}
import java.lang.*;

class calc
{
int no1, no2;

calc(int a, int b)
{
no1 = a;  // 30
no2 = b;  // 10
}

/*
public void assign(int a, int b)
{
no1 = a;  // 30
            no2 = b;  // 10
}
```

```
*/
public void sum()
{
System.out.println("Sum of "+no1+" and "+no2+" is "+(no1+no2));
}
}

class add
{
public static void main(String args[])
{
/* calc c; // object declaration. It will have null in it.
c = new calc(20,10); // Construction call
*/
calc c = new calc(30, 10);
//c.assign(30,10);
c.sum();
}
}
```

2. When a class has two or more methods by the same name but different parameters, at the time of calling based on the parameters passed respective method is called (or respective method body will be bonded with the calling line dynamically). This mechanism is known as **method overloading**.

While overloading you need to keep the following points in mind −

- Both methods should be in the same class.
- The name of the methods should be same but they should have different number or, type of parameters.

If the names are different they will become different methods and, if they have same name and parameters a compile time error will be thrown saying “method already defined”.

```
class Test{

  public int division(int a, int b){

    int result = a/b;

    return result;

  }

  public double division (float a, float b){

    double result = a/b;

    return result;

  }

}
```

```
public class OverloadingExample{
   public static void main(String args[]){
      Test obj = new Test();
      System.out.println("division of integers: "+obj.division(100, 40));
      System.out.println("division of floating point numbers: "+obj.division(214.225f, 50.60f));
   }
}
```

3.

	super()	this()
Definition	super() - refers immediate parent class instance.	this() - refers current class instance.
Invoke	Can be used to invoke immediate parent class method.	Can be used to invoke current class method.
Constructor	super() acts as immediate parent class constructor and should be first line in child class constructor.	this() acts as current class constructor and can be used in parametrized constructors.
Override	When invoking a superclass version of an overridden method the super keyword is used.	When invoking a current version of an overridden method the this keyword is used.

Ex:

```
class Animal {
   String name;
   Animal(String name) {
      this.name = name;
   }
   public void move() {
      System.out.println("Animals can move");
   }
   public void show() {
      System.out.println(name);
   }
}
class Dog extends Animal {
```

```
  Dog() {
    //Using this to call current class constructor
    this("Test");
  }
  Dog(String name) {
    //Using super to invoke parent constructor
    super(name);
  }
  public void move() {
    // invokes the super class method
    super.move();
    System.out.println("Dogs can walk and run");
  }
}
public class Tester {
  public static void main(String args[]) {
    // Animal reference but Dog object
    Animal b = new Dog("Tiger");
    b.show();
    // runs the method in Dog class
    b.move();
  }
}
```

4. If a class has multiple methods having same name but different in parameters, it is known as **Method Overloading**.

If we have to perform only one operation, having same name of the methods increases the readability of the program.

Suppose you have to perform addition of the given numbers but there can be any number of arguments, if you write the method such as a(int,int) for two parameters, and b(int,int,int) for three parameters then it may be difficult for you as well as other programmers to understand the behavior of the method because its name differs.

Method overloading *increases the readability of the program*.

Different ways to overload the method

There are two ways to overload the method in java

1. By changing number of arguments
2. By changing the data type

```
class Adder{
static int add(int a,int b){return a+b;}
static int add(int a,int b,int c){return a+b+c;}
}
class TestOverloading1{
public static void main(String[] args){
System.out.println(Adder.add(11,11));
System.out.println(Adder.add(11,11,11));
}}
```

5.

```
import
java.util.Scanner;

                    public class Citizens {

                      public static void main(String[] args) {

                        int seniorCitizens, nonSeniorCitizens, age;

                        Scanner input = new Scanner(System.in);

                        System.out.print("Enter current seniorCitizens and
                    nonSeniorCitizens:");
```

```
    seniorCitizens = input.nextInt();
    nonSeniorCitizens = input.nextInt();


    System.out.print("What is new citizen's age?");


    age = input.nextInt();


    if (age >= 65) {
      System.out.println("Senior Citizen");
      //      seniorCitizens = seniorCitizens + 1;
      seniorCitizens += 1;
    } else {
      System.out.println("Non-Senior Citizen");
      nonSeniorCitizens = nonSeniorCitizens + 1;
    }


    System.out.println("New Senior Citizens count " + seniorCitizens);
    System.out.println("New Non-Senior Citizens count " +
nonSeniorCitizens);


  }
}
```