1) Accept a byte, int, long, float, double, boolean, char and String type of values through the keyboard and display them.

```
import java.io.*;
class Pg1
{
  public static void main(String args[])throws IOException
  {
    byte n1;
    int n2;
    long n3;
    float n4;
    double n5;
    boolean n6;
    char  n7;
    String n8;

    BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

    System.out.println("Enter byte type value");
    n1=(byte)Integer.parseInt(br.readLine());

    System.out.println("Enter Integer type value");
    n2=Integer.parseInt(br.readLine());

    System.out.println("Enter long type value");
    n3=Long.parseLong(br.readLine());

    System.out.println("Enter float type value");
    n4=Float.parseFloat(br.readLine());

    System.out.println("Enter double type value");
    n5=Double.parseDouble(br.readLine());

    System.out.println("Enter boolean type value");
    n6=Boolean.parseBoolean(br.readLine());

    System.out.println("Enter  char type value");
    n7= br.readLine().charAt(0);

    System.out.println("Enter string type value");
    n8= br.readLine();

    System.out.println("Byte value :" +n1);
    System.out.println("Integer  value :" +n2);
    System.out.println("long value :" +n3);
    System.out.println("Float value :" +n4);
    System.out.println("Double value :" +n5);
    System.out.println("boolean value :" +n6);
    System.out.println("char value :" +n7);
    System.out.println("String value :" +n8);
    }
  }
```

2)Accept two integers and display their sum

```
import java.io.*;
class Pg2
{
  public static void main(String args[])throws IOException
  {
    int n1,n2,sum;
    BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

    System.out.println("Enter First number");
    n1=Integer.parseInt(br.readLine());

    System.out.println("Enter Second number");
    n2=Integer.parseInt(br.readLine());

    sum=n1+n2;

    System.out.println("Sum of 2 Integers :"+sum);
  }
}
```

3) Accept an amount. If the amount is greater than 20000 then calculate the commission as 2% of the amount. Else commission will be 0. Display the amount and commission. (Use simple if).

```
import java.lang.*;
import java.io.*;
class Pg3
{
  public static void main(String args[])throws IOException
  {
    float amount;
    double commission=0;
    BufferedReader br = new BufferedReader(new InputStreamReader(System.in));


    System.out.println("Enter an amount");
    amount=Float.parseFloat(br.readLine());

   if(amount>20000)
   {
    commission=amount*0.2;
   }


      System.out.println("Amount :"+amount);
      System.out.println("Commission :" + commission);
  }
}
```

4) Accept a number and display if it is odd or even. (Use simple if and System.exit(0)).

```
import java.io.*;
class Pg4
{
  public static void main(String args[])throws IOException
  {
    int n;
    BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
    System.out.println("Enter Number");
    n=Integer.parseInt(br.readLine());
    if(n%2==0)
    {
    System.out.println(n+"is even");
    System.exit(0);
    }

    System.out.println(n+"is odd");

  }
}
```

5) Accept a number and display if it is odd or even. (Use if else)

```
import java.io.*;
class Pg5
{
  public static void main(String args[])throws IOException
  {
    int n;
    BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
    System.out.println("Enter Number");
    n=Integer.parseInt(br.readLine());
    if(n%2==0)
    System.out.println(n+"is even");

    else
    System.out.println(n+"is odd");

  }
}
```

6) Accept empno, ename, grade and salary.

If grade='A' or 'a', then commission will be 14% of salary.
If grade='B' or 'b', then commission will be 10% of salary.
If grade='C' or 'c', then commission will be 7% of salary.
Else display "Invalid grade!!!" and terminate the process.
If grade is valid, display the empno, name, grade, salary and commission.
(Use if else if)

```
import java.io.*;
class pg6
{
  public static void main(String args[])throws IOException
```

```
  {
    int empno;
     char grade;
    String ename ;
    float  salary;
    double cms=0;
    BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

    System.out.println("Enter emp Number");
    empno=Integer.parseInt(br.readLine());

    System.out.println("Enter name");
    ename= br.readLine();

    System.out.println("Enter Grade");
    grade= br.readLine().charAt(0);

     System.out.println("Enter Salary");
    salary=Float.parseFloat(br.readLine());

    if(grade=='A'||grade=='a')
    {
      cms=salary*0.14;
    }
   else if(grade=='B'||grade=='b')
    {
      cms=salary*0.10;
    }
    else if(grade=='C'||grade=='c')
    {
      cms=salary*0.07;
    }
   else
      {
       System.out.println("Invalid  Grade!!");
       System.exit(0);
      }

  System.out.println("\nEmpno:"+empno);
System.out.println("Name:"+ename );
System.out.println("Grade:"+grade);
 System.out.println("Salary:"+salary);
 System.out.println("commision:"+cms);


  }
}
```

7) Repeat the question 6 with switch statement.

```
import java.io.*;
class Pg7
{
  public static void main(String args[])throws IOException
  {
    int empno;
     char grade;
```

```
    String ename ;
    float  salary;
    double cms=0;
    BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

    System.out.println("Enter emp Number");
    empno=Integer.parseInt(br.readLine());

    System.out.println("Enter name");
    ename= br.readLine();

    System.out.println("Enter Grade");
    grade= br.readLine().charAt(0);

     System.out.println("Enter Salary");
    salary=Float.parseFloat(br.readLine());

    switch(grade)
    {
      case 'A' : cms=salary*0.14;
                 break;
      case 'B' : cms=salary*0.10;
                 break;
      case 'C' : cms=salary*0.07;
                 break;
      default : System.out.println("Invalid     Grade!!");
     }

      System.out.println("\nEmpno:"+empno);
System.out.println("Name:"+ename );
System.out.println("Grade:"+grade);
 System.out.println("Salary:"+salary);
 System.out.println("commision:"+cms);

 }
}
```

8) Repeat question 6. But if grade is invalid, display the message "Invalid Grade!!!" and accept only grade once again for that employee. (Use while true, continue and break).

```
import java.io.*;
class Pg8
{
  public static void main(String args[])throws IOException
  {
    int empno;
     char grade;
    String ename ;
    float  salary;
    double cms=0;
    BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

    System.out.println("Enter emp Number");
    empno=Integer.parseInt(br.readLine());
```

```
    System.out.println("Enter name");
    ename= br.readLine();

     System.out.println("Enter Salary");
    salary=Float.parseFloat(br.readLine());
 while(true)
     {
     System.out.println("Enter Grade");
    grade= br.readLine().charAt(0);
    if(grade=='A'||grade=='a')
    {
      cms=salary*0.14;
      break;

    }
   else if(grade=='B'||grade=='b')
    {
      cms=salary*0.10;
      break;

    }
    else if(grade=='C'||grade=='c')
    {
      cms=salary*0.07;
      break;

    }
   else
      {
       System.out.println("Invalid  Grade!!");
       continue;

      }
   }

  System.out.println("\nEmpno:"+empno);
System.out.println("Name:"+ename );
System.out.println("Grade:"+grade);
 System.out.println("Salary:"+salary);
 System.out.println("commision:"+cms);


  }
}
```

9) Repeat question 8 for any number of employees, that is, accept, calculate, display while ans=”yes”.

```
import java.io.*;
class Pg9
{
  public static void main(String args[])throws IOException
  {
    int empno;
     char grade;
```

```
    String ename ;
    float  salary;
    double cms=0;
    char ans='y';
    BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
    while(ans=='y')
    {

    System.out.println("Enter emp Number");
    empno=Integer.parseInt(br.readLine());

    System.out.println("Enter name");
    ename= br.readLine();



     System.out.println("Enter Salary");
    salary=Float.parseFloat(br.readLine());
 while(true)
     {
     System.out.println("Enter Grade");
    grade= br.readLine().charAt(0);
    if(grade=='A'||grade=='a')
    {
      cms=salary*0.14;
      break;

    }
   else if(grade=='B'||grade=='b')
    {
      cms=salary*0.10;
      break;

    }
    else if(grade=='C'||grade=='c')
    {
      cms=salary*0.07;
      break;

    }
   else
      {
       System.out.println("Invalid  Grade!!");
       continue;

      }
   }

  System.out.println("\nEmpno:"+empno);
System.out.println("Name:"+ename );
System.out.println("Grade:"+grade);
 System.out.println("Salary:"+salary);
 System.out.println("commision:"+cms);

 System.out.println("do you want to continue<y/n/?");
 ans=br.readLine().charAt(0);
  }
```

```
  }
}
```

10) Accept a number and display its factorial. (Use for statement).

```
import java.io.*;
class Pg10
{
  public static void main(String args[])throws IOException
  {
    int num,i;
    long fact=1;
    BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
    System.out.println("Enter  Number");
    num=Integer.parseInt(br.readLine());

    if(num<0)
      System.out.println("error!!");
    else
    {
      for(i=1; i<=num; ++i)
      {
       fact*=i;
      }
      System.out.println("factorial of "+num+  "\tis\t"   +fact);
    }

   }
  }
```

ASSIGNMENT 2

1. **Write a program to create a class Student with data 'name, city and age' along with method printData to display the data. Create the two objects s1 ,s2 to declare and access the values.**

```
import java.io.*;

class student
{
  String name;
  String city;
  int age;
   void getdata() throws IOException
```

```
   {
   BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
   System.out.println("Enter name");
   name=br.readLine();
   System.out.println("Enter city");
   city=br.readLine();
   System.out.println("Enter age");
   age=Integer.parseInt(br.readLine());
   }


  void printdata()
  {
    System.out.println(" student name:"+name+"\n city:"+city+"\nage:"+age);
    }
  }

    class St1
    {
      public static void main(String args[])throws IOException
     {
       student s1=new student();
       s1.getdata();
       s1.printdata();

        student s2=new student();
       s2.getdata();
       s2.printdata();

     }
   }
```

2) Write a program to create a class Student2 along with two method getData(),printData() to get the value through
argument and display the data in printData. Create the two objects s1 ,s2 to declare and access the values from
class STtest

```
import java.io.*;

class student2
{
  String name;
  String city;
  int age;
   void getdata(String sname,String scity ,int sage) throws IOException
   {
    name=sname;
    city=scity;
    age=sage;
   }
```

```
  void printdata()
  {
    System.out.println(" student name:"+name+"\n city:"+city+"\nage:"+age);
    }
  }

    class St2
    {
      public static void main(String args[])throws IOException
     {
       String name,city;
       int age;
       BufferedReader br = new BufferedReader(new
InputStreamReader(System.in));
   System.out.println("Enter name");
   name=br.readLine();
   System.out.println("Enter city");
   city=br.readLine();
   System.out.println("Enter age");
   age=Integer.parseInt(br.readLine());


       student2 s1=new student2();
       s1.getdata(name,city,age);
       s1.printdata();

      System.out.println("Enter name");
   name=br.readLine();
   System.out.println("Enter city");
   city=br.readLine();
   System.out.println("Enter age");
   age=Integer.parseInt(br.readLine());

        student2 s2=new student2();
       s2.getdata(name,city,age);
       s2.printdata();

     }
   }
```

3) Write a Java program to show 0-arguments constructor

```
import java.io.*;

class Alpha
{
  int a,b,c;
  Alpha()
  {
   a=20;
   b=30;
  }

   void calculate()
   {
```

```
     c=a+b;
    }

    void printdata()
     {
       System.out.println(a+ "+" +b+ "=" +c);
      }

    }

   class A2p3
     {
      public static void main(String args[])throws IOException
     {
      Alpha a =new Alpha();
      a.calculate();
      a.printdata();
     }
    }
```

4) Write a Java program to show parameterized constructor

```
. import java.io.*;

class Alpha1
{
  int a,b,c;
  Alpha1(int x,int y)
  {
   a=x;
   b=y;
  }

   void calculate()
   {
    c=a+b;
   }

   void printdata()
    {
      System.out.println(a+ "+" +b+ "=" +c);
     }

   }

  class A2p4
    {
      public static void main(String args[])throws IOException
     {
      Alpha1 a1 =new Alpha1(30,20);
      a1.calculate();
      a1.printdata();
     }
    }
```

5) WAP using parameterized constructor with two parameters id and name. While creating the objects obj1 and obj2 passed two arguments so that this constructor gets invoked after creation of obj1 and obj2.

```
import java.io.*;


class emp
{
  int id;
  String name;

  emp(int x,String y)
  {
   name=y;
   id=x;
   printf();
  }
  void printf()
    {
      System.out.println("Employee Id : "+id);
      System.out.println("Employee name : "+name);
     }

  }

   class A2p5
    {
      public static void main(String args[])throws IOException
     {

      emp e5 =new emp(101,"Dheeraj");

     emp e2 =new emp(102,"Rakshith");



     }
    }
```

6) Write a Java program to show constructor overloading.

```
import java.io.*;

class Delta
{
  int a,b,c;
  Delta(int x,int y)
  {
   a=x;
   b=y;
  }

  Delta()
  {
   a=b=0;
  }
```

```
  Delta(int z)
  {
   a=b=z;
   }


   void calculate()
   {
    c=a+b;
   }

   void printdata()
    {
      System.out.println(a+ "+" +b+ "=" +c);
     }

   }

  class A2p6
    {
      public static void main(String args[])throws IOException
     {
      Delta d1 =new Delta(30,20);
      Delta d2 =new Delta();
      Delta d3 =new Delta(50);

      d1.calculate();
      d1.printdata();

      d2.calculate();
      d2.printdata();

      d3.calculate();
      d3.printdata();
     }
    }
```

7) Write a Java program to show METHOD overloading

```
import java.io.*;

class Beta
{
  int a,b,c;
  void getbeta() throws IOException
  {
    BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
    System.out.println("Enter values for a and b");
    a=Integer.parseInt(br.readLine());
    b=Integer.parseInt(br.readLine());
  }

    void calculate()
   {
    c=a+b;
   }

   void calculate(int p)
```

```
   {
    c=(a*b)/p;
   }
   void printdata()
    {
      System.out.println("Output =" +c);
     }

   }

  class A2p7
    {
     public static void main(String args[])throws IOException
    {
     Beta x = new Beta();
      x.getbeta();
      x.calculate();
      x.calculate(2);
      x.printdata();


    }
   }
```

8) Write a class, Grader, which has an instance variable, score, an appropriate constructor and appropriate methods. A method, letterGrade() that returns the letter grade as O/E/A/B/C/F.
Now write a demo class to test the Grader class by reading a score from the user, using it to create a Grader object after validating that the value is not negative and is not greater then 100. Finally, call the letterGrade() method to get and print the grade.

```
import java.io.*;

class Grader
{
  int score;
  Grader(int score)
  {
    this.score=score;
  }

  char lettergrade()
   {
     if(score<=35)
       return 'F';
     else if(score<=50)
        return 'C';
     else if(score<=65)
        return 'B';
      else if(score<=80)
        return 'A';
      else if(score<=95)
        return 'E';
      else
        return 'O';
```

```
    }

 }

 class A2p8
 {
    public static void main(String args[])throws IOException
     {
       int score;
       BufferedReader br = new BufferedReader(new
InputStreamReader(System.in));

          System.out.println(" Enter a valid score :");
          score=Integer.parseInt(br.readLine());

        Grader g=new Grader(score);
        System.out.println(" Grade :"+g.lettergrade());
    }
 }
```

9) Write a class, Commission, which has an instance variable, sales; an appropriate constructor; and a method, commission() that returns the commission. Now write a demo class to test the Commission class by reading a sale from the user, using it to create a Commission object after validating that the value is not negative. Finally, call the commission() method to get and print the commission. If the sales are negative, your demo should print the message "Invalid Input".

```
import java.io.*;

class Commission
{
  float sales;
  Commission(float sales)
  {
    this.sales=sales;
   }

   float Commission()
    {
     return (float)0.1*sales;
    }

 }


class A2p9
{
  public static void main(String args[])throws IOException
     {
      float sales;
       BufferedReader br = new BufferedReader(new
InputStreamReader(System.in));
        System.out.println(" Enter sales :");
        sales=Float.parseFloat(br.readLine());
```

```
        if(sales<0)
          {
             System.out.println("INVALID SALES!!");
             return;
          }

       Commission c =new Commission(sales);
        System.out.println(" Commisiion (20%) :"+c.Commission());
      }
   }
```

10) Write a Java program to calculate total, average and grade for 10 students. Consider the necessary datamembers and member methods required to achieve the requirements

```
import java.io.*;

class student
{
  int sid;
  String name;
  int m1,m2,m3;
  long total;
  float avg;


   void getdata() throws IOException
   {
   BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
   System.out.println("Enter  student id ");
   sid=Integer.parseInt(br.readLine());
   System.out.println("Enter name");
   name=br.readLine();
   System.out.println("Enter marks in 3subject");
    m1=Integer.parseInt(br.readLine());
    m2=Integer.parseInt(br.readLine());
    m3=Integer.parseInt(br.readLine());
   }


  void printdata()
  {
    System.out.println(" student name:"+name+"\n city:"+city+"\nage:"+age);
    }
  }

    class A2p10
    {
      public static void main(String args[])throws IOException
     {
       student s1=new student();
       s1.getdata();
       s1.printdata();

        student s2=new student();
       s2.getdata();
       s2.printdata();
```

```
    }
  }
```

ASSIGNMENT 3

1)Write a program to demonstrate static variables and methods.

```
//Write a program to demonstrate static variables and methods.


class Demo1
{
	static int age = 25;

	static int getAge()
	{
		return age;
	}

	void display()
	{
		System.out.print("Age (non-static method): "+getAge());
	}
}

class A3P1
{
    public static void main(String args[])
    {
        System.out.println("Age (static method): "+Demo1.getAge());
        Demo1 d = new Demo1();
        d.display();
    }
}
```

2) Write a program for reuse class

```
/*
Write a program for reuse class.
*/

class Demo2
{
	Demo2(int i)
	{
		System.out.println("Class instance "+i);
	}
}

class A3P2
{
    public static void main(String args[])
    {
        new Demo2(1);
        new Demo2(2);
```

```
    }
}
```

3)Write a program to give the example for method overriding concepts

```
class X
{
      void display()
      {
            System.out.println("Class X method (parent class method)");
      }
}

class Y extends X
{
      void display()
      {
            System.out.println("Class Y method (overloaded child class
method)");
            super.display();
      }
}

class A3P3
{
    public static void main(String args[])
    {
        Y obj = new Y();
        obj.display();
    }
}

 /*
4)Write a program to give the example for 'super' keyword.
*/

class C
{
      void display()
      {
            System.out.println("Class C method (parent/super class method)");
      }
}

class D extends C
{
      void display()
      {
            System.out.println("Class D method (child class method)");
            super.display();
      }
}

class A3P4
{
    public static void main(String args[])
    {
```

```
        D obj = new D();
        obj.display();
    }
```

5)
```
/*
Write a program to create a class named shape. In this class we have three
sub classes circle, triangle and square each class has two member function
named draw () and erase (). Create these using polymorphism concepts.
*/

class Shape
{
      void draw()
      {
            System.out.println("Draw shape");
      }

      void erase()
      {
            System.out.println("Erase shape");
      }
}

class Circle extends Shape
{
      void draw()
      {
            System.out.println("Draw circle");
      }

      void erase()
      {
            System.out.println("Erase circle");
      }
}

class Triangle extends Shape
{
      void draw()
      {
            System.out.println("Draw triangle");
      }

      void erase()
      {
            System.out.println("Erase triangle");
      }
}

class Square extends Shape
{
      void draw()
      {
            System.out.println("Draw square");
      }
```

```
      void erase()
      {
            System.out.println("Erase square");
      }
}

class A3P5
{
    public static void main(String args[])
    {
        Shape obj;

        obj = new Shape();
        obj.draw();
        obj.erase();
        System.out.println();
        obj = new Circle();
        obj.draw();
        obj.erase();
        System.out.println();
        obj = new Triangle();
        obj.draw();
        obj.erase();
        System.out.println();
        obj = new Square();
        obj.draw();
        obj.erase();
    }
}
```

6)
```
/*
Write a program to create interface A in this interface we have two method
meth1 and meth2. Implements this interface in another class named MyClass.
*/

interface A
{
      void meth1();
      void meth2();
}

class MyClass implements A
{
      public void meth1()
      {
            System.out.println("Don't do meth");
      }

      public void meth2()
      {
            System.out.println("Let's do meth");
      }
}

class A3P6
```

```
{
    public static void main(String args[])
    {
        MyClass obj = new MyClass();
        obj.meth1();
        obj.meth2();
    }
}
```

7)
```
/*
Write a program to give example for multiple inheritance in Java.
*/

interface P
{
      void meth1();
}

interface Q
{
      void meth2();
}

class Demo7 implements P, Q
{
      public void meth1()
      {
            System.out.println("Don't do meth");
      }

      public void meth2()
      {
            System.out.println("Let's do meth");
      }
}

class A3P7
{
    public static void main(String args[])
    { Demo7
        Demo7 obj = new ();
        obj.meth1();
        obj.meth2();
    }
}
```

8)
```
/*
Write a program to create interface named test. In this interface the member
function is square. Implement this interface in arithmetic class. Create one
new class called ToTestInt in this class use the object of arithmetic class.
*/

interface Test
{
```

```
      float square(float x);
}

class Arithmetic implements Test
{
      public float square(float x)
      {
            return (float)(x*x);
      }
}

class A3P8
{
    public static void main(String args[])
    {
      float x = 5;
        Arithmetic obj = new Arithmetic();
        System.out.println(x+"^2 = "+obj.square(x));
    }
}
```

9)

```
/*
Create an outer class with a function display, again create another class
inside the outer class named inner with a function called display and call
the two functions in the main class.
*/

class OuterClass
{
      void display()
      {
            System.out.println("Outer class display method");
      }

      class InnerClass
      {
            void display()
            {
                  System.out.println("Inner class display method");
            }
      }
}

class A3P9
{
    public static void main(String args[])
    {
      OuterClass obj = new OuterClass();
      obj.display();
      OuterClass.InnerClass objk = obj.new InnerClass();
      objk.display();
    }
}
```

10) /* Write a program to give the example for 'this' operator And also use the 'this' keyword as return statement.*/

```
class Student
{
String name;

Student(String name)
{
this.name = name;
}

String display()
{
return this.name;
}
}

public class program10
{
public static void main(String args[])
{
Student s1 = new Student("ramesh");
Student s2 = new Student("suresh");

System.out.println(s1.display());
System.out.println(s2.display());
}
}
```

11) Create a base class Building that stores the number of floors of a building, number of rooms and it's total footage. Create a derived class House that inherits Building and also stores the number of bedrooms and bathrooms. Demonstrate the working of the classes

```
// Building class
class Building {
    private int floors;
    private int rooms;
    private double footage;

    public Building(int floors, int rooms, double footage) {
        this.floors = floors;
        this.rooms = rooms;
        this.footage = footage;
    }

    public int getFloors() {
        return floors;
    }

    public int getRooms() {
        return rooms;
    }
```

```
    public double getFootage() {
        return footage;
    }
}

// House class (derived from Building)
class House extends Building {
    private int bedrooms;
    private int bathrooms;

    public House(int floors, int rooms, double footage, int bedrooms, int
bathrooms) {
        super(floors, rooms, footage);
        this.bedrooms = bedrooms;
        this.bathrooms = bathrooms;
    }

    public int getBedrooms() {
        return bedrooms;
    }

    public int getBathrooms() {
        return bathrooms;
    }
}

// Demonstration of the classes
public class program11 {
    public static void main(String[] args) {

        // Create a building
        Building building = new Building(5, 20, 5000.0);
        System.out.println("Building - Floors: " + building.getFloors() +
                ", Rooms: " + building.getRooms() +
                ", Footage: " + building.getFootage());

        // Create a house
        House house = new House(2, 6, 2000.0, 3, 2);
        System.out.println("House - Floors: " + house.getFloors() +
                ", Rooms: " + house.getRooms() +
                ", Footage: " + house.getFootage() +
                ", Bedrooms: " + house.getBedrooms() +
                ", Bathrooms: " + house.getBathrooms());
    }
}
```

12)In the earlier program, create a second derived class Office that inherits Building and stores the number of telephones and tables. Now demonstrate the working of all three classes.

```
class Building {
    private int floors;
    private int rooms;
    private double footage;
```

```
    public Building(int floors, int rooms, double footage) {
        this.floors = floors;
        this.rooms = rooms;
        this.footage = footage;
    }

    public int getFloors() {
        return floors;
    }

    public int getRooms() {
        return rooms;
    }

    public double getFootage() {
        return footage;
    }
}

// House class (derived from Building)
class House extends Building {
    private int bedrooms;
    private int bathrooms;

    public House(int floors, int rooms, double footage, int bedrooms, int
bathrooms) {
        super(floors, rooms, footage);
        this.bedrooms = bedrooms;
        this.bathrooms = bathrooms;
    }

    public int getBedrooms() {
        return bedrooms;
    }

    public int getBathrooms() {
        return bathrooms;
    }
}

// Office class (derived from Building)
class Office extends Building {
    private int telephones;
    private int tables;

    public Office(int floors, int rooms, double footage, int telephones, int
tables) {
        super(floors, rooms, footage);
        this.telephones = telephones;
        this.tables = tables;
    }

    public int getTelephones() {
        return telephones;
    }
```

```
    public int getTables() {
        return tables;
    }
}

// Demonstration of the classes
public class program12 {
    public static void main(String[] args) {

        // Create a building
        Building building = new Building(5, 20, 5000.0);
        System.out.println("Building - Floors: " + building.getFloors() +
                ", Rooms: " + building.getRooms() +
                ", Footage: " + building.getFootage());

        // Create a house
        House house = new House(2, 6, 2000.0, 3, 2);
        System.out.println("House - Floors: " + house.getFloors() +
                ", Rooms: " + house.getRooms() +
                ", Footage: " + house.getFootage() +
                ", Bedrooms: " + house.getBedrooms() +
                ", Bathrooms: " + house.getBathrooms());

        // Create an office
        Office office = new Office(10, 50, 10000.0, 50, 100);
        System.out.println("Office - Floors: " + office.getFloors() +
                ", Rooms: " + office.getRooms() +
                ", Footage: " + office.getFootage() +
                ", Telephones: " + office.getTelephones() +
                ", Tables: " + office.getTables());
    }
}
```

13) Write a Java program which creates a base class Num and contains an integer number along with a method shownum() which displays the number. Now create a derived class HexNum which inherits Num and overrides shownum() which displays the hexadecimal value of the number. Demonstrate the working of the classes.

```
// Num base class
class Num {
    int number;

    public Num(int number) {
        this.number = number;
    }

    public void showNum() {
        System.out.println("Number: " + number);
    }
}

// HexNum derived class (inherits Num)
class HexNum extends Num {
    public HexNum(int number) {
        super(number);
    }
```

```
    @Override
    public void showNum() {
        System.out.println("Hexadecimal Value: " +
Integer.toHexString(super.number));
    }
}

// Demonstration of the classes
public class program13 {
    public static void main(String[] args) {
        // Create an instance of Num class
        Num num = new Num(42);
        num.showNum();

        // Create an instance of HexNum class
        HexNum hexNum = new HexNum(42);
        hexNum.showNum();
    }
}
```

14)Create a base class called “vehicle” that stores number of wheels and speed. Create the following derived classes – “car” that inherits “vehicle” and also stores number of passengers. “truck” that inherits “vehicle” and also stores the load limit. Write a main function to create objects of these two derived classes and display all the information about “car” and “truck”. Also compare the speed of these two vehicles - car and truck and display which one is faster.

```
// Vehicle base class
class Vehicle
{
int wheels;
double speed;

public Vehicle(int wheels, double speed)
{
this.wheels = wheels;
this.speed = speed;
}


public void displayInfo()
{
System.out.println("Wheels: " + wheels + ", Speed: " + speed + "km/h");
}
}

// Car derived class (inherits Vehicle)
class Car extends Vehicle
{
int passengers;
```

```
public Car(int wheels, double speed, int passengers)
{
super(wheels, speed);
this.passengers = passengers;
}

public void displayInfo()
{
super.displayInfo();
System.out.println("Passengers: " + passengers);
}
}

// Truck derived class (inherits Vehicle)
class Truck extends Vehicle
{
double loadLimit;

public Truck(int wheels, double speed, double loadLimit)
{
super(wheels, speed);
this.loadLimit = loadLimit;
}

public void displayInfo()
{
super.displayInfo();
System.out.println("Load Limit: " + loadLimit + " tons");
}
}

// Demonstration of the classes
public class program14
{
public static void main(String[] args)
{

// Create a car object
Car car = new Car(4, 120.0, 4);
System.out.println("Car Information:");car.displayInfo();

// Create a truck object
Truck truck = new Truck(6, 80.0, 10.0);
System.out.println("Truck Information:");
truck.displayInfo();

// Compare speeds
if (car.speed > truck.speed)
{
System.out.println("The car is faster than the truck.");
}

else if (car.speed < truck.speed)
{
System.out.println("The truck is faster than the car.");
}
```

```
else
{
System.out.println("The car and truck have the same speed.");
}
}
}
```

ASSIGNMENT 4

1. Write a program to create a class named shape. In this class we have three sub classes circle, triangle and square each class has two member function named draw () and erase (). Create these using polymorphism concepts in java

```
// Shape class (parent class)
class Shape {
    // Common methods for all subclasses
    public void draw() {
        System.out.println("Drawing a shape");
    }

    public void erase() {
        System.out.println("Erasing a shape");
    }
}

// Circle class (subclass of Shape)
class Circle extends Shape {
```

```
    @Override
    public void draw() {
        System.out.println("Drawing a circle");
    }

    @Override
    public void erase() {
        System.out.println("Erasing a circle");
    }
}

// Triangle class (subclass of Shape)
class Triangle extends Shape {
    @Override
    public void draw() {
        System.out.println("Drawing a triangle");
    }

    @Override
    public void erase() {
        System.out.println("Erasing a triangle");
    }
}

// Square class (subclass of Shape)
class Square extends Shape {
    @Override
    public void draw() {
```

```
        System.out.println("Drawing a square");
    }

    @Override
    public void erase() {
        System.out.println("Erasing a square");
    }
}

public class Main {
    public static void main(String[] args) {
        // Using polymorphism to call draw and erase methods
        Shape shape1 = new Circle();
        Shape shape2 = new Triangle();
        Shape shape3 = new Square();

        shape1.draw();
        shape1.erase();

        shape2.draw();
        shape2.erase();

        shape3.draw();
        shape3.erase();
    }
}
```

2) Write a program to give a simple example for abstract class.

```
abstract class Shape {
    protected String color;
```

```
    public Shape(String color) {
        this.color = color;
    }

    // Abstract method to calculate the area of the shape
    public abstract double calculateArea();

    // Concrete method
    public void display() {
        System.out.println("This is a " + color + " shape.");
    }
}

class Circle extends Shape {
    private double radius;

    public Circle(String color, double radius) {
        super(color);
        this.radius = radius;
    }

    @Override
    public double calculateArea() {
        return Math.PI * radius * radius;
    }
}

class Rectangle extends Shape {
    private double length;
    private double width;

    public Rectangle(String color, double length, double width) {
        super(color);
        this.length = length;
        this.width = width;
    }

    @Override
    public double calculateArea() {
        return length * width;
    }
}

public class a4prog2 {
    public static void main(String[] args) {
        Circle circle = new Circle("Red", 5.0);
        circle.display();
        System.out.println("Area: " + circle.calculateArea());

        Rectangle rectangle = new Rectangle("Blue", 4.0, 6.0);
        rectangle.display();
        System.out.println("Area: " + rectangle.calculateArea());
    }
}
```

3) Write a program to create interface A in this interface we have two method meth1 and meth2. Implements this interface in another class named MyClass

```
// Interface A
interface A {
    void meth1();

    void meth2();
}

// MyClass implementing interface A
class MyClass implements A {
    @Override
    public void meth1() {
        System.out.println("Implementation of meth1()");
    }

    @Override
    public void meth2() {
        System.out.println("Implementation of meth2()");
    }
}

public class Main {
    public static void main(String[] args) {
        // Creating an object of MyClass
        MyClass myClass = new MyClass();

        // Calling the methods of the implemented interface
        myClass.meth1();
        myClass.meth2();
    }
}
```

4) Write a program to give example for multiple inheritance in Java

```
// Interface A
interface A {
    void methodA();
}

// Interface B
interface B {
    void methodB();
}

// Class implementing both interfaces A and B
class MyClass implements A, B {
    @Override
    public void methodA() {
        System.out.println("Implementation of methodA()");
    }
```

```
    @Override
    public void methodB() {
        System.out.println("Implementation of methodB()");
    }
}

public class Main {
    public static void main(String[] args) {
        // Creating an object of MyClass
        MyClass myClass = new MyClass();

        // Calling methods from Interface A
        myClass.methodA();

        // Calling methods from Interface B
        myClass.methodB();
    }
}
```

5) Write a program to create interface named test. In this interface the member function is square. Implement this interface in arithmetic class. Create one new class called ToTestInt in this class use the object of arithmetic class

```
// Interface Test

interface Test {

    int square(int num);

}


// Arithmetic class implementing interface Test

class Arithmetic implements Test {

    @Override

    public int square(int num) {

        return num * num;

    }

}


// ToTestInt class

class ToTestInt {
```

```
    public static void main(String[] args) {
        // Creating an object of Arithmetic class
        Arithmetic arithmetic = new Arithmetic();

        // Using the object to call the square() method
        int result = arithmetic.square(5);
        System.out.println("Square of 5 is: " + result);
    }
}
```

6) Create an outer class with a function display, again create another class inside the outer class named inner with a function called display and call the two functions in the main class.

```
// Outer class
class Outer {
    void display() {
        System.out.println("Outer class display()");
    }

    // Inner class
    class Inner {
        void display() {
            System.out.println("Inner class display()");
        }
    }
}

public class Main {
    public static void main(String[] args) {
        // Creating an object of the Outer class
        Outer outer = new Outer();

        // Calling the display() method of the Outer class
        outer.display();

        // Creating an object of the Inner class (inside Outer class)
        Outer.Inner inner = outer.new Inner();

        // Calling the display() method of the Inner class
        inner.display();
    }
```

}

7) Write a program to create a package named mypack and import it in circle class.

8) Write a program to create a package named package in ex1 class

9) Create class box and box3d. box3d is extended class of box. The above two classes going to pull fill following requirement

i.™ Include constructor.

ii. set value of length, breadth, height

iii. Find out area and volume.

Note: Base class and sub classes have respective methods and instance variables.

```
class Box {
    protected double length;
    protected double breadth;
    protected double height;

    public Box(double length, double breadth, double height) {
        this.length = length;
        this.breadth = breadth;
        this.height = height;
    }

    public double calculateArea() {
        return length * breadth;
    }
    public double calculateVolume() {
        return length * breadth * height;
    }
}

class Box3D extends Box {
```

```
    public Box3D(double length, double breadth, double height) {
        super(length, breadth, height);
    }

    public double calculateArea1() {
        return length * breadth;
}
    public double calculateVolume1() {
        return length * breadth * height;
    }
}

public class a4prog9 {
    public static void main(String[] args) {
        Box box = new Box(5.0, 3.0, 2.0);
        Box3D box3d = new Box3D(5.0, 3.0, 2.0);

      double area = box.calculateArea();
        double volume = box.calculateVolume();

        double area1 = box3d.calculateArea1();
        double volume1 = box3d.calculateVolume1();

        System.out.println("For Box");
        System.out.println("Area: " + area);
        System.out.println("Volume: " + volume);

        System.out.println("For Box3D");
        System.out.println("Area: " + area1);
        System.out.println("Volume: " + volume1);
    }

}
```

10) Write a program for example of try and catch block. In this check whether the given array size is negative or not.

```
import java.util.Scanner;

public class a4prog10 {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        try {
            System.out.print("Enter the size of the array: ");
            int size = scanner.nextInt();

            if (size < 0) {
                throw new NegativeArraySizeException();
            }

            int[] array = new int[size];
            System.out.println("Array created successfully!");
        }

        catch (NegativeArraySizeException e) {
```

```
            System.out.println("Array size cannot be negative.");
        }

        catch (Exception e) {
            System.out.println("An error occurred: " + e.getMessage());
        }

        scanner.close();
    }
}
```

11) Write a program for example of multiple catch statements occurring in a program

```
public class a4prog11 {
    public static void main(String[] args) {
        try {
            int[] numbers = { 1, 2, 3 };
            System.out.println(numbers[4]); // Accessing index out of bounds
            int result = 10 / 0; // ArithmeticException and
ArrayIndexOutOfBoundsException

            System.out.println("Result: " + result);
        }

        catch (ArithmeticException e) {
            System.out.println("ArithmeticException caught: " +
e.getMessage());
        }

        catch (ArrayIndexOutOfBoundsException e) {
            System.out.println("ArrayIndexOutOfBoundsException caught: " +
e.getMessage());
        }

        catch (Exception e) {
            System.out.println("Exception caught: " + e.getMessage());
        }
    }
}
```

12) Write a program to illustrate usage of try/catch with finally clause.

```
public class a4prog12 {
    public static void main(String[] args) {
        try {
            int result = divide(10, 0); // Division by zero
            System.out.println("Result: " + result);
        }
        catch (ArithmeticException e) {
            System.out.println("Exception caught: " + e.getMessage());
        }
        finally {
            System.out.println("Finally block executed.");
        }
```

```
    }

    public static int divide(int numerator, int denominator) {
        try {
            return numerator / denominator;
        }
        catch (ArithmeticException e) {
            throw e;
        }
    }
}
```

13) Write a program to describe usage of throws clause for a user defined method

```
public class a4prog13 {
    public static void main(String[] args) {
        try {
            validateAge(19); // Calling the method with an invalid age
            System.out.println("Age is valid.");
        } catch (Exception e) {
            System.out.println("Exception caught: " + e.getMessage());
        }
    }

    public static void validateAge(int age) throws Exception {
        if (age < 18) {
            throw new Exception("Age must be 18 or above.");
        }
    }
}

/*class InvalidAgeException extends Exception {
    public InvalidAgeException(String message) {
        super(message);
    }
}*/
```

14) Write a program for creation of user defined exception.

```
// CustomException.java - User-defined exception class
class CustomException extends Exception {
  public CustomException(String message) {
    super(message);
  }
}

// Example of using the custom exception
public class Main {
  public static void main(String[] args) {
    try {
      int dividend = 10;
      int divisor = 0;
```

```
        if (divisor == 0) {
            throw new CustomException("Division by zero is not allowed.");
        } else {
            int result = dividend / divisor;
            System.out.println("Result of division: " + result);
        }
    } catch (CustomException e) {
        System.out.println("Exception: " + e.getMessage());
    }
  }
}
```

ASSIGNMENT 5

1) Write a program to demonstrate creating and handling main thread

```
public class MainThreadExample {
    public static void main(String[] args) {
        // Get the reference to the main thread
        Thread mainThread = Thread.currentThread();

        // Display information about the main thread
        System.out.println("Main thread ID: " + mainThread.getId());
        System.out.println("Main thread name: " + mainThread.getName());
        System.out.println("Main thread priority: " + mainThread.getPriority());
        System.out.println("Main thread is daemon: " + mainThread.isDaemon());
        System.out.println("Main thread is alive: " + mainThread.isAlive());

        // Simulate some work in the main thread
        for (int i = 1; i <= 5; i++) {
            System.out.println("Main thread counting: " + i);
            try {
                Thread.sleep(1000); // Pause for 1 second
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }

        System.out.println("Main thread execution finished!");
    }
}
```

2) Write a program to demonstrate thread life cycle

```
public class ThreadLifecycleDemo {
   public static void main(String[] args) {
      // Create a new thread
      Thread thread = new Thread(new MyRunnable());

      // Get the initial state of the thread
      Thread.State initialState = thread.getState();
      System.out.println("Initial state: " + initialState);

      // Start the thread
      thread.start();

      // Get the state of the thread after starting
      Thread.State afterStartState = thread.getState();
      System.out.println("After start state: " + afterStartState);

      try {
         // Sleep for a while to let the thread do some work
         Thread.sleep(2000);
      } catch (InterruptedException e) {
         e.printStackTrace();
      }

      // Get the state of the thread after some work
      Thread.State afterWorkState = thread.getState();
      System.out.println("After work state: " + afterWorkState);

      // Wait for the thread to finish its execution
      try {
         thread.join();
      } catch (InterruptedException e) {
         e.printStackTrace();
      }

      // Get the final state of the thread
      Thread.State finalState = thread.getState();
      System.out.println("Final state: " + finalState);
   }
}

class MyRunnable implements Runnable {
   public void run() {
      try {
         // Simulate some work by sleeping for 1 second
         Thread.sleep(1000);
      } catch (InterruptedException e) {
         e.printStackTrace();
      }
```

```
        System.out.println("Thread is executing.");
    }
}
```

3) Write a program to demonstrate handling of multiple threads

```
public class MultipleThreadsDemo {
    public static void main(String[] args) {
        // Creating and starting three threads
        Thread thread1 = new MyThread("Thread 1");
        Thread thread2 = new MyThread("Thread 2");
        Thread thread3 = new MyThread("Thread 3");

        thread1.start();
        thread2.start();
        thread3.start();
    }
}

class MyThread extends Thread {
    private String name;

    public MyThread(String name) {
        this.name = name;
    }

    @Override
    public void run() {
        for (int i = 1; i <= 5; i++) {
            System.out.println(name + " - Count: " + i);

            try {
                Thread.sleep(1000); // Pause for 1 second
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
}
```

4) Write a program to demonstrate thread priorities

```
public class ThreadPriorityDemo {
    public static void main(String[] args) {
        Thread lowPriorityThread = new MyThread("Low Priority Thread");
        Thread normalPriorityThread = new MyThread("Normal Priority Thread");
        Thread highPriorityThread = new MyThread("High Priority Thread");
```

```
        // Set thread priorities
        lowPriorityThread.setPriority(Thread.MIN_PRIORITY);
        normalPriorityThread.setPriority(Thread.NORM_PRIORITY);
        highPriorityThread.setPriority(Thread.MAX_PRIORITY);

        // Start all threads
        lowPriorityThread.start();
        normalPriorityThread.start();
        highPriorityThread.start();
    }
}

class MyThread extends Thread {
    private String name;

    public MyThread(String name) {
        this.name = name;
    }

    @Override
    public void run() {
        for (int i = 1; i <= 5; i++) {
            System.out.println(name + " - Count: " + i);

            try {
                Thread.sleep(1000); // Pause for 1 second
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
}
```

5) Write a program to demonstrate synchronization

```
public class SynchronizationDemo {
    public static void main(String[] args) {
        SharedCounter sharedCounter = new SharedCounter();

        // Create three threads and start them
        Thread thread1 = new CounterThread(sharedCounter);
        Thread thread2 = new CounterThread(sharedCounter);
        Thread thread3 = new CounterThread(sharedCounter);

        thread1.start();
        thread2.start();
        thread3.start();

        // Wait for all threads to complete their execution
```

```
        try {
            thread1.join();
            thread2.join();
            thread3.join();
        } catch (InterruptedException e) {
            e.printStackTrace();
        }

        // Print the final value of the counter
        System.out.println("Final counter value: " + sharedCounter.getCounter());
    }
}

// SharedCounter class to hold the shared counter
class SharedCounter {
    private int counter;

    public SharedCounter() {
        this.counter = 0;
    }

    // Method to increment the counter (without synchronization)
    public void increment() {
        int temp = counter;
        temp++;
        counter = temp;
    }

    // Method to get the counter value
    public int getCounter() {
        return counter;
    }
}

// CounterThread class to represent a thread that increments the shared counter
class CounterThread extends Thread {
    private SharedCounter sharedCounter;

    public CounterThread(SharedCounter sharedCounter) {
        this.sharedCounter = sharedCounter;
    }

    @Override
    public void run() {
        for (int i = 0; i < 100000; i++) {
            sharedCounter.increment();
        }
    }
```

```
}
```

6) Write a program to demonstrate Interthread communication

```
public class InterthreadCommunicationDemo {
   public static void main(String[] args) {
      SharedData sharedData = new SharedData();

      Thread thread1 = new NumberPrinterThread(sharedData, true); // Print even numbers
      Thread thread2 = new NumberPrinterThread(sharedData, false); // Print odd numbers

      thread1.start();
      thread2.start();
   }
}

class SharedData {
   private volatile boolean isEven = true;

   public synchronized void printEven(int number) {
      while (!isEven) {
         try {
            wait();
         } catch (InterruptedException e) {
            e.printStackTrace();
         }
      }
      System.out.println("Even: " + number);
      isEven = false;
      notify();
   }

   public synchronized void printOdd(int number) {
      while (isEven) {
         try {
            wait();
         } catch (InterruptedException e) {
            e.printStackTrace();
         }
      }
      System.out.println("Odd: " + number);
      isEven = true;
      notify();
   }
}

class NumberPrinterThread extends Thread {
```

```
    private SharedData sharedData;
    private boolean printEven;

    public NumberPrinterThread(SharedData sharedData, boolean printEven) {
        this.sharedData = sharedData;
        this.printEven = printEven;
    }

    @Override
    public void run() {
        int start = printEven ? 2 : 1;
        for (int i = start; i <= 10; i += 2) {
            if (printEven) {
                sharedData.printEven(i);
            } else {
                sharedData.printOdd(i);
            }
        }
    }
}
```

7) Write a program to demonstrate RandomAccessFile

```
import java.io.IOException;
import java.io.RandomAccessFile;

public class RandomAccessFileDemo {
    public static void main(String[] args) {
        // File path
        String filePath = "data.txt";

        // Write data to the file
        writeDataToFile(filePath);

        // Read data from the file
        readDataFromFile(filePath);
    }

    public static void writeDataToFile(String filePath) {
        try (RandomAccessFile file = new RandomAccessFile(filePath, "rw")) {
            // Write data to the file at position 0
            file.writeInt(42);

            // Move the file pointer to the end of the file
            file.seek(file.length());

            // Write more data to the end of the file
```

```
            file.writeUTF("Hello, RandomAccessFile!");

            System.out.println("Data has been written to the file.");
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    public static void readDataFromFile(String filePath) {
        try (RandomAccessFile file = new RandomAccessFile(filePath, "r")) {
            // Read data from the file at position 0
            int intValue = file.readInt();
            System.out.println("Integer value read from the file: " + intValue);

            // Move the file pointer to the end of the file
            file.seek(file.length());

            // Read data from the end of the file
            String stringValue = file.readUTF();
            System.out.println("String value read from the file: " + stringValue);
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

8) Write a program to demonstrate Object Serialization and Deserialization

```
import java.io.*;

class Person implements Serializable {
    private static final long serialVersionUID = 1L;
    private String name;
    private int age;

    public Person(String name, int age) {
        this.name = name;
        this.age = age;
    }

    public String getName() {
        return name;
    }

    public int getAge() {
        return age;
```

```
    }

    @Override
    public String toString() {
        return "Person [name=" + name + ", age=" + age + "]";
    }
}

public class SerializationDemo {
    public static void main(String[] args) {
        // Create a Person object
        Person person = new Person("John Doe", 30);

        // Serialize the object to a file
        serializeObject(person, "person.ser");

        // Deserialize the object from the file
        Person deserializedPerson = deserializeObject("person.ser");

        // Display the deserialized object
        System.out.println("Deserialized Object: " + deserializedPerson);
    }

    private static void serializeObject(Person person, String filename) {
        try (FileOutputStream fileOut = new FileOutputStream(filename);
             ObjectOutputStream objectOut = new ObjectOutputStream(fileOut)) {
            objectOut.writeObject(person);
            System.out.println("Object has been serialized.");
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    private static Person deserializeObject(String filename) {
        Person person = null;
        try (FileInputStream fileIn = new FileInputStream(filename);
             ObjectInputStream objectIn = new ObjectInputStream(fileIn)) {
            person = (Person) objectIn.readObject();
            System.out.println("Object has been deserialized.");
        } catch (IOException | ClassNotFoundException e) {
            e.printStackTrace();
        }
        return person;
    }
}
```

1. **Write a program to demonstrate difference between abstract class and interface**

```
abstract class adder{
     void add(int a, int b){
         System.out.println("Sum is: "+ (a+b));
}
    abstract void mul(int a, int b);
}
interface subtraction{
      void sub(int a, int b);
}
class addmul extends adder{
        void mul(int a, int b){
              System.out.println("Product is: "+ (a*b));
}
}
class subt implements subtraction{
        public void sub(int a, int b){
             System.out.println("Difference is: "+ (a-b));
}
}
class abint{
      public static void main(String args[]){
               adder a=new  addmul();
               subtraction b=new subt();
               a.add(6,4);
               b.sub(6,4);
               a.mul(6,4);
}
}
```

2. **Write a program to demonstrate upcasting and compile time polymorphism**

```
class over {
   void meth1(int rollno, String name, int marks) {
      System.out.println("Rollno: " + rollno + " Name: " + name + " Marks: " + marks);
   }

   void meth2(int empno, String city) {
      System.out.println("Empno: " + empno + " City: " + city);
   }

   void meth2(int empno, String name, String city) {
      System.out.println("Empno: " + empno + " Name: " + name + " City: " + city);
```

```
    }
}

class upcast extends over {
    void meth1(int rollno, String name, int marks) {
        System.out.println("Rollno: " + rollno + " Name: " + name + " Marks: " + marks);
    }

}

class upcastover {
    public static void main(String args[]) {
        over s1 = new over();
        over s2 = new upcast();
        s1.meth1(101, "ABC", 98);
        s1.meth2(501, "Mangalore");
        s1.meth2(502, "DEF", "Udupi");
        s2.meth1(102, "XYZ", 54);
    }
}
```

3. Display multiplication table of positive integer in applet

```
import java.applet.Applet;
import java.awt.Button;
import java.awt.Label;
import java.awt.TextField;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

public class MultiplicationTableApplet extends Applet implements ActionListener {
    private TextField numberInput;
    private Label resultLabel;
    private Button displayButton;

    @Override
    public void init() {
        numberInput = new TextField(10);
        resultLabel = new Label();
        displayButton = new Button("Display Table");
        displayButton.addActionListener(this);

        add(new Label("Enter a positive integer:"));
        add(numberInput);
        add(displayButton);
        add(resultLabel);
    }
```

```
    @Override
    public void actionPerformed(ActionEvent e) {
        if (e.getSource() == displayButton) {
            String inputText = numberInput.getText();
            try {
                int number = Integer.parseInt(inputText);
                if (number > 0) {
                    generateMultiplicationTable(number);
                } else {
                    resultLabel.setText("Please enter a positive integer.");
                }
            } catch (NumberFormatException ex) {
                resultLabel.setText("Invalid input. Please enter a valid positive integer.");
            }
        }
    }

    private void generateMultiplicationTable(int number) {
        StringBuilder table = new StringBuilder();
        for (int i = 1; i <= 10; i++) {
            table.append(number).append(" x ").append(i).append(" = ").append(number *
i).append("\n");
        }
        resultLabel.setText(table.toString());
    }
}
```

4. **Celsius to Fahrenheit converter using frames**

```
import java.awt.*;
import java.awt.event.*;
import java.applet.*;
class tempConv extends Frame implements ActionListener
{
 Label lbDeg,lbFar;
 TextField tfDeg,tfFar;
 Button btComp,btClear;
 tempConv()
 {
 setLayout(new GridLayout(4,2));
 setSize(400,200);
 setVisible(true);
 lbDeg = new Label("Enter temperature in degree
Celcius:");
 add(lbDeg);
 tfDeg = new TextField();
```

```
add(tfDeg);
lbFar = new Label("Temperature in Farhenheit is:");
add(lbFar);
tfFar = new TextField();
add(tfFar);
btComp = new Button("Convert Celcius to Farhenheit");
btComp.addActionListener(this);
add(btComp);
addWindowListener(new WindowAdapter()
{
public void windowClosing(WindowEvent we)
{
System.exit(0);
}
});
}
public void actionPerformed(ActionEvent ae)
{
if(ae.getSource() == btComp)
{
int celcius = Integer.parseInt(tfDeg.getText());
int farhenheit = (celcius *9/5)+32;
tfFar.setText(String.valueOf(farhenheit));
}

if(ae.getSource() == btClear)
{
tfDeg.setText("");
tfFar.setText("");
tfDeg.requestFocus();
}
}
class apoorva
{
public static void main(String[] args)
{
new tempConv();
}
}
}
```

5. **Write a java program to demonstrate use of pakages and implement interface inheritance**

   Step1: Create a folder as Student
   Step2: Create a file Student.java in created folder.

Student.java

```
package student;
public interface student
{
public void Rollno();
public void Course();
public void marks();
}
```

Step3:Create a file mymain.java outside the student folder

Mymain.java

```
import student.*;
class test implements student
{
public void Rollno()
{
System.out.println("18C535");
}
public void Course()
{
System.out.println("oop");
}
public void marks()
{
System.out.println("85");
}
}
class mymain
{
public static void main(String args[])
{
test t=new test();
t.Rollno();
t.Course();
t.marks();
}
}
```

Now run Mymain.java

6. **US dollar to INR convert in applet**

```
import java.awt.*;
import java.awt.event.*;
import java.applet.*;
```

```
/*
<applet code="MyProgram" width=600 height=300>
</applet>
*/

class MyProgram extends Applet implements ActionListener
{
Label l1;
Label l2;
TextField txtUS;
TextField txtINR;
Button b1;

public void init(){
l1=new Label("US currency");
txtUS=new TextField(20);
l2=new Label("INR currency");
txtINR=new TextField(20);
b1=new Button("Convert");

add(l1);
add(txtUS);
add(l2);
add(txtINR);
add(b1);

b1.addActionListener(this);

}
public void actionPerformed(ActionEvent e){
if(e.getSource()==b1){
float f1,f2;
f1=Float.parseFloat(txtUS.getText());
f2=f1*82.30f;
txtINR.setText(Float.toString(f2));
}
}
}
```

7. Count the characters in the file

```
import java.io.*;
class fileinput
{
public static void main (String[] args) throws IOException
{
 FileInputStream fis= new FileInputStream("file1.txt");
int ch,count=0;
try{
```

```
while((ch=fis.read())!= -1){
    //if(ch!=' ') //if you don't want to consider spaces
         count++;
}
System.out.println((count));
}
catch(Exception e){}


}
      }

Create another text file file.txt
```

8. Gui for largest number finder of 3 numbers using frame

```
import java.awt.event.*;
import java.awt.*;
class Largest_no_Finder extends Frame implements ActionListener {
   TextField t1,t2,t3;
   Label l1,l2,l3,l4;
   Button cal;
   Largest_no_Finder(){
      l1=new Label("Enter the first number:");
      l1.setBounds(100,80,150,30);
      add(l1);
      l2=new Label("Enter the second number:");
      l2.setBounds(100,115,150,30);
      add(l2);
      l3=new Label("Enter the third number:");
      l3.setBounds(100,150,150,30);
      add(l3);
      t1=new TextField();
      t1.setBounds(255,80,150,25);
      add(t1);
      t2=new TextField();
      t2.setBounds(255,115,150,25);
      add(t2);
      t3=new TextField();
      t3.setBounds(255,150,150,25);
      add(t3);
      cal=new Button("calculate");
      cal.setBounds(200,185,120,30);
      add(cal);
      l4=new Label("Oops! Enter the numbers");
      l4.setBounds(180,220,150,30);
      cal.addActionListener(this);
      setLayout(null);
      setSize(600,600);
```

```
        setVisible(true);
        this.addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent e) {
                System.exit(0);
            }
        });
    }
    public void actionPerformed(ActionEvent e){
        if(t1.getText().equals("") || t2.getText().equals("") || t3.getText().equals("")) {
            l4.setForeground(Color.RED);
            add(l4);
        }
        else{
            int n1,n2,n3,max;
            n1=Integer.parseInt(t1.getText());
            n2=Integer.parseInt(t2.getText());
            n3=Integer.parseInt(t3.getText());
            if(n1>n2 && n1>n3)
                max=n1;
            else if(n2>n3)
                max=n2;
            else
                max=n3;

            l4.setText("Largest no. is "+max);
            l4.setForeground(Color.GREEN);
            add(l4);
        }

    }
}

public class p1 {
    public static void main(String a[]){
        new Largest_no_Finder();
    }
}
```

9. applet to find Fibonacci

```
import java.awt.*;
import java.awt.event.*;
import java.applet.*;

public class FibonacciApplet extends Applet {
```

```
private TextField inputField;
private TextArea resultArea;

public void init() {
    Label inputLabel = new Label("Enter the number of terms:");
    inputField = new TextField(10);
    Button calculateButton = new Button("Calculate Fibonacci");
    resultArea = new TextArea(10, 30);
    resultArea.setEditable(false);

    calculateButton.addActionListener(e -> calculateFibonacci());

    add(inputLabel);
    add(inputField);
    add(calculateButton);
    add(resultArea);
}

private void calculateFibonacci() {
    try {
        int n = Integer.parseInt(inputField.getText());
        if (n < 1) {
            resultArea.setText("Please enter a positive number.");
            return;
        }

        int[] fibonacciSeries = new int[n];
        fibonacciSeries[0] = 0;
        if (n > 1) {
            fibonacciSeries[1] = 1;
        }

        for (int i = 2; i < n; i++) {
            fibonacciSeries[i] = fibonacciSeries[i - 1] + fibonacciSeries[i - 2];
        }

        StringBuilder result = new StringBuilder();
        for (int num : fibonacciSeries) {
            result.append(num).append(", ");
        }

        resultArea.setText("Fibonacci series up to " + n + " terms:\n" + result.toString());
    } catch (NumberFormatException ex) {
        resultArea.setText("Invalid input. Please enter a valid integer.");
    }
}
```

```
}
```

10. using swing find the factorial

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

public class FactorialCalculator extends JFrame {
    private JTextField numberField;
    private JButton calculateButton;
    private JLabel resultLabel;

    public FactorialCalculator() {
        setTitle("Factorial Calculator");
        setSize(300, 150);
        setLayout(new GridLayout(3, 1));

        numberField = new JTextField();
        numberField.setHorizontalAlignment(JTextField.CENTER);

        calculateButton = new JButton("Calculate Factorial");
        calculateButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                calculateFactorial();
            }
        });

        resultLabel = new JLabel("", JLabel.CENTER);

        add(new JLabel("Enter a number:"));
        add(numberField);
        add(calculateButton);
        add(resultLabel);

        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }

    private void calculateFactorial() {
        try {
            int num = Integer.parseInt(numberField.getText());
            if (num < 0) {
                resultLabel.setText("Please enter a non-negative integer.");
                return;
```

```
                }

                long factorial = 1;
                for (int i = 1; i <= num; i++) {
                    factorial *= i;
                }

                resultLabel.setText("Factorial of " + num + " is: " + factorial);
            } catch (NumberFormatException ex) {
                resultLabel.setText("Invalid input. Please enter a valid integer.");
            }
        }

        public static void main(String[] args) {
            SwingUtilities.invokeLater(new Runnable() {
                public void run() {
                    FactorialCalculator frame = new FactorialCalculator();
                    frame.setVisible(true);
                }
            });
        }
    }
```

11.File copy example

```
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;

public class FileCopyExample {
    public static void main(String[] args) {
        FileReader fileReader = null;
        FileWriter fileWriter = null;
        try {
            fileReader = new FileReader("FileCopyExample.java");
            fileWriter = new FileWriter("destination.txt");

            int character;
            while ((character = fileReader.read()) != -1) {
                fileWriter.write(character);
            }

            System.out.println("File copied successfully.");
        } catch (IOException e) {
            e.printStackTrace();
        } finally {
```

```
            try {
                if (fileReader != null) {
                    fileReader.close();
                }
                if (fileWriter != null) {
                    fileWriter.close();
                }
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }
}
```