IT5512- WEB TECHNOLOGY LAB-SESSION-3

DATE: 20/09/2021 FN

NAME: A.S. PRUTHIEV

REG NO.2019506067

JAVA INHERITANCE AND POLYMORPHISM PROGRAMS

1) AIM:

To write a single level inheritance java program.

ALGORITHM:

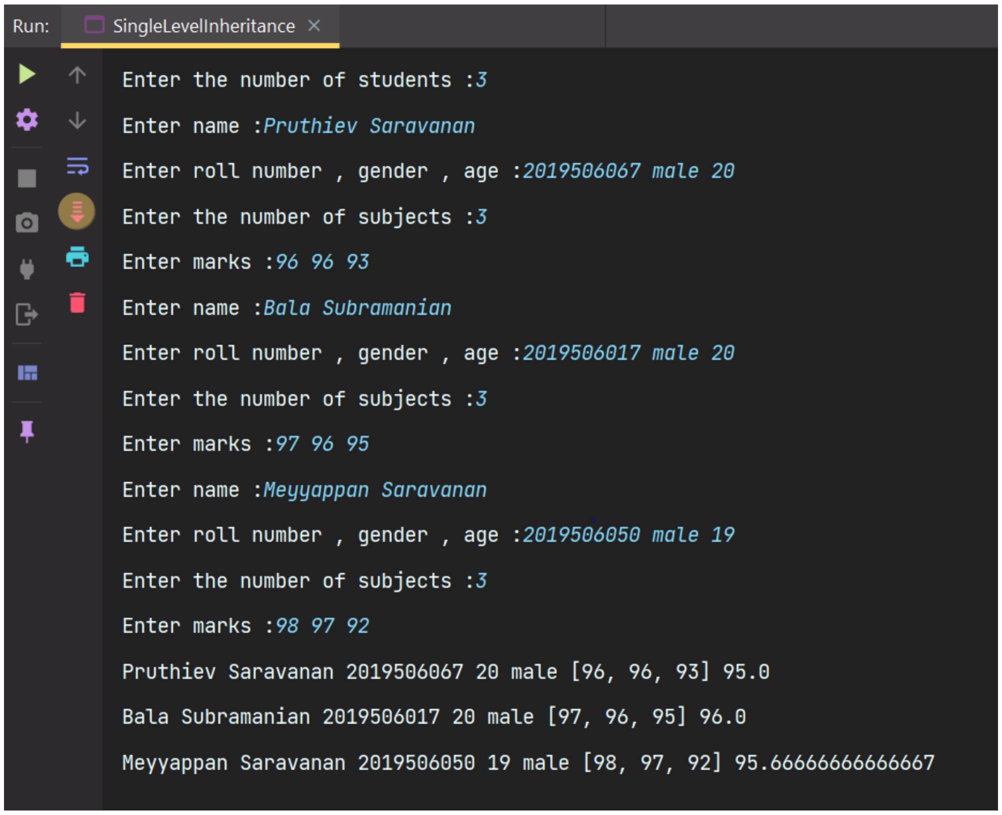
- ✓ Create a class Student with private varaibles name,roll number,gender
- ✓ Create a constructor intializing the values
- ✓ Create a getter methods for each variables
- ✓ Create another classs Resut which extends Student whch has double cgpa ,and an array of marks
- ✓ Create a constructor which calls the super constructor and intializes the marks array
- ✓ Overload the toString method to return the result accordingly
- ✓ Create a method calculate cgpa which does the operation and stores the result in the cgpa variable.

PROGRAM CODE:

```
package Java.Lab.lab3;

import java.util.ArrayList;
import java.util.Scanner;

class Student{
   private String name,rollNumber,gender;
   int age;
   public Student(String name, String rollNumber, String gender, int age) {
```

```
            this.name = name;
            this.rollNumber = rollNumber;
            this.gender = gender;
            this.age = age;
        }
        public String getName() {
            return name;
        }
        public String getRollNumber() {
            return rollNumber;
        }
        public String getGender() {
            return gender;
        }
        public int getAge() {
            return age;
        }
    }
    class Result extends Student{
        private double cgpa;
        ArrayList<Integer>marks;
        Result(String name,String rollNumber,String gender,int
age,ArrayList<Integer>marks){
            super(name,rollNumber,gender,age);
            this.marks = marks;
        }
        void calculateCGPA(){
            cgpa = marks.stream().mapToInt(x -> x).average().orElse(-1);
        }
        @Override
        public String toString() {
            return getName() + " " + getRollNumber() + " " + getAge() + " " + getGender() + "
" + marks.toString() + " " + cgpa;
        }
    }
    public class SingleLevelInheritance {
        private static final Scanner input = new Scanner(System.in);
        public static void main(String[] args) {
            int n;
            String name,rollNumber,gender;
            int age;
            System.out.print("Enter the number of students :");
            n = input.nextInt();
            ArrayList<Result>students = new ArrayList<>();
            for(int i = 0 ; i < n ; i++){
                input.nextLine();
                System.out.print("Enter name :");
                name = input.nextLine();
                System.out.print("Enter roll number , gender , age :");
                rollNumber = input.next();
                gender = input.next();
                age = input.nextInt();
                int numMarks;
                System.out.print("Enter the number of subjects :");
                numMarks = input.nextInt();
                System.out.print("Enter marks :");
                ArrayList<Integer>marks = new ArrayList<>();
                for(int j = 0 ; j < numMarks ; j++){
                    int m = input.nextInt();
                    marks.add(m);
                }
                students.add(new Result(name,rollNumber,gender,age,marks));
                students.get(i).calculateCGPA();
            }
            for(int i = 0 ; i < n ; i++){
                System.out.println(students.get(i));
            }
        }
}
```

OUTPUT:



```
Enter the number of students :3
Enter name :Pruthiev Saravanan
Enter roll number , gender , age :2019506067 male 20
Enter the number of subjects :3
Enter marks :96 96 93
Enter name :Bala Subramanian
Enter roll number , gender , age :2019506017 male 20
Enter the number of subjects :3
Enter marks :97 96 95
Enter name :Meyyappan Saravanan
Enter roll number , gender , age :2019506050 male 19
Enter the number of subjects :3
Enter marks :98 97 92
Pruthiev Saravanan 2019506067 20 male [96, 96, 93] 95.0
Bala Subramanian 2019506017 20 male [97, 96, 95] 96.0
Meyyappan Saravanan 2019506050 19 male [98, 97, 92] 95.66666666666667
```

RESULT:

Thus, the program has been executed successfully.

2)

AIM:

To Write a Multilevel inheritance java program

ALGORITHM:

- ✓ Create a player class with name,state,country,age
- ✓ Create a constructor which intializes the variables and getter methods
- ✓ Create a Gamelevel class which extends player with fields like level,numberof matches,yearsofexperience
- ✓ Create a constructor which calls the super and intializes the variables and getter methods
- ✓ Create a Chess player class which inturn extends from Gamelevel with fields like totalRating,BlitzRating,rapidRating,classicalRating
- ✓ Create a constructor which calls the super and intializes the variables and getter methods
- ✓ Calculate the totalRating and return the level according to the condition

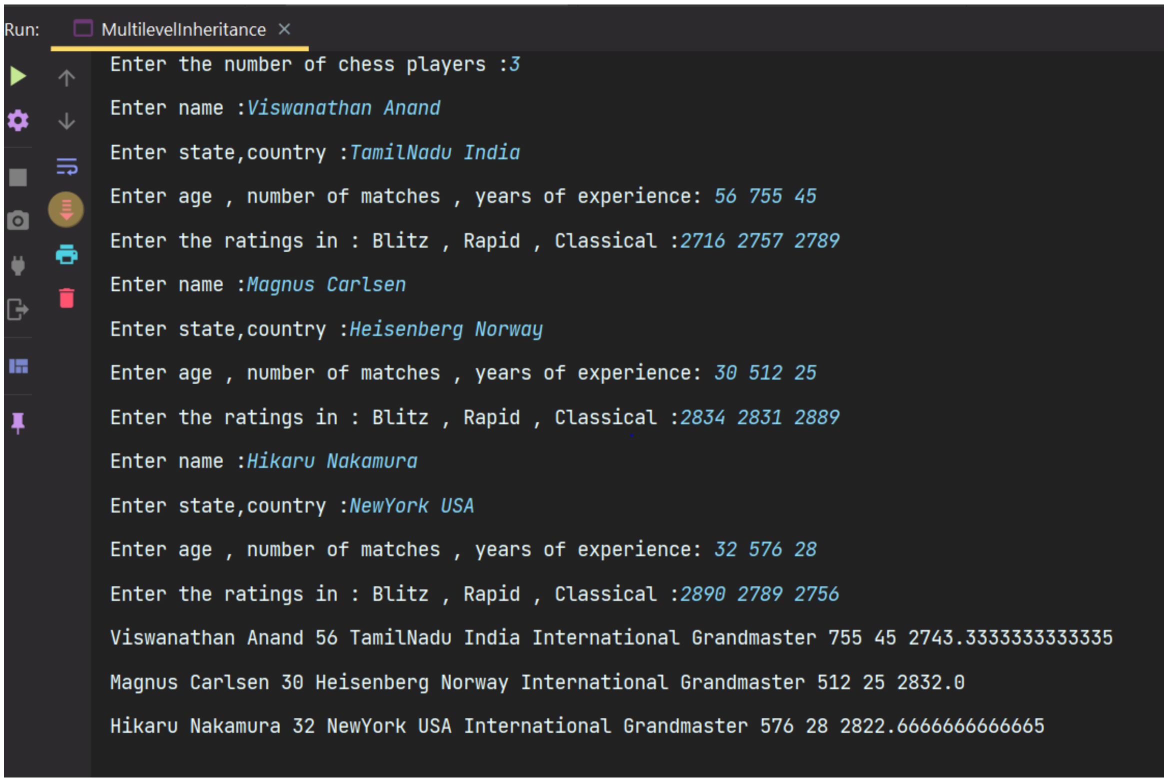
PROGRAM CODE

```
package Java.Lab.lab3;

import java.util.ArrayList;
import java.util.Scanner;

class Player{
   private String name,state,country;
   private int age;
   public Player(String name, String state, String country,int age) {
      this.name = name;
```

```
        this.state = state;
        this.country = country;
        this.age = age;
    }
    public String getName() {
        return name;
    }
    public String getState() {
        return state;
    }
    public String getCountry() {
        return country;
    }
    public int getAge() {
        return age;
    }
}
class GameLevel extends Player{
    protected String level;
    int numberOfMatches;
    int yearsOfExperience;
    public GameLevel(String name, String state, String country,int age, int
numberOfMatches, int yearsOfExperience) {
        super(name, state, country,age);
        this.numberOfMatches = numberOfMatches;
        this.yearsOfExperience = yearsOfExperience;
    }
    public String getLevel() {
        return level;
    }
    public int getNumberOfMatches() {
        return numberOfMatches;
    }
    public int getYearsOfExperience() {
        return yearsOfExperience;
    }
}
class ChessPlayer extends GameLevel{
    double blitzRating , rapidRating , classicalRating,totalRating;
    public ChessPlayer(String name, String state, String country,int age,int numberOfMatches,int yearsOfExperience,double blitzRating, double rapidRating, double classicalRating) {
        super(name,state,country,age,numberOfMatches,yearsOfExperience);
        this.blitzRating = blitzRating;
        this.rapidRating = rapidRating;
        this.classicalRating = classicalRating;
        this.totalRating = (this.blitzRating + this.rapidRating + this.rapidRating)/3;
    }
    public double getBlitzRating() {
        return blitzRating;
    }
    public double getRapidRating() {
        return rapidRating;
    }
    public double getClassicalRating() {
        return classicalRating;
    }
    public void determineLevel(){
        if(totalRating >=0 && totalRating <=1500)level = "Beginner";
        else if(totalRating > 1500 && totalRating <= 1800)level = "Expert";
        else if(totalRating > 1800 && totalRating <= 2000)level = "Specialist";
        else if(totalRating > 2000 && totalRating <= 2300)level = "Candidate Master";
        else if(totalRating > 2300 && totalRating <= 2500)level = "Master";
        else if(totalRating > 2500 && totalRating <= 2700)level = "Grandmaster";
        else if(totalRating > 2700 && totalRating <= 2900)level = "International Grandmaster";
        else level = "Legendary Grandmaster";
    }

    @Override
    public String toString() {
        return getName() + " " + getAge() + " " + getState() + " " + getCountry() + " " +
              getLevel() + " " + getNumberOfMatches() + " " + getYearsOfExperience() + " " +
              totalRating;
    }
}
public class MultilevelInheritance {
    private static final Scanner input = new Scanner(System.in);
    public static void main(String[] args) {
        int n;
        String name,state,country;
        int age ,numberOfMatches,yearsOfExperience;
        double blitzRating , rapidRating , classicalRating;
        System.out.print("Enter the number of chess players :");
        n = input.nextInt();
        ArrayList<ChessPlayer>chessPlayers = new ArrayList<>();
        for(int i = 0 ; i < n ; i ++){
            input.nextLine();
            System.out.print("Enter name :");
            name = input.nextLine();
            System.out.print("Enter state,country :");
            state = input.next();
            country = input.next();
            System.out.print("Enter age , number of matches , years of experience: ");
            age = input.nextInt();
            numberOfMatches = input.nextInt();
            yearsOfExperience = input.nextInt();
            System.out.print("Enter the ratings in : Blitz , Rapid , Classical :");
            blitzRating = input.nextInt();
            rapidRating = input.nextInt();
            classicalRating = input.nextInt();
            chessPlayers.add(new
ChessPlayer(name,state,country,age,numberOfMatches,yearsOfExperience,blitzRating,rapi
dRating,classicalRating));
            chessPlayers.get(i).determineLevel();
        }
```

```
            for(int i = 0; i < n ; i++){
              System.out.println(chessPlayers.get(i));
            }
          }
}
```

OUTPUT:



```
Enter the number of chess players :3
Enter name :Viswanathan Anand
Enter state,country :TamilNadu India
Enter age , number of matches , years of experience: 56 755 45
Enter the ratings in : Blitz , Rapid , Classical :2716 2757 2789
Enter name :Magnus Carlsen
Enter state,country :Heisenberg Norway
Enter age , number of matches , years of experience: 30 512 25
Enter the ratings in : Blitz , Rapid , Classical :2834 2831 2889
Enter name :Hikaru Nakamura
Enter state,country :NewYork USA
Enter age , number of matches , years of experience: 32 576 28
Enter the ratings in : Blitz , Rapid , Classical :2890 2789 2756
Viswanathan Anand 56 TamilNadu India International Grandmaster 755 45 2743.3333333333335
Magnus Carlsen 30 Heisenberg Norway International Grandmaster 512 25 2832.0
Hikaru Nakamura 32 NewYork USA International Grandmaster 576 28 2822.6666666666665
```

RESULT:

Thus, the program has been executed successfully.

3)AIM :

To Write a Hierarchical Inheritance java progam

ALGORITHM:

- ✓ Create pair class with data members like name,points .create a constructor and getter methods
- ✓ Create a class Cricket with fields like format , numOvers,numberofdays
- ✓ Creata a constructor Cricket which intializes the variables and getter methods
- ✓ Create a class TestCricket which inherits from Cricket with fields like Pair teams[] ,name, points
- ✓ Create a contructor which intializes the variables and enter the three team names and their points
- ✓ Create a sortData method which sort the names of the teams according to the points
- ✓ Similarly create a class for ODICrciket and T20Cricket which inherites from Cricket and repeat the same operations as we did it in the TestCricket class
- ✓ Create an object of the TestCricket , ODICricket,T20Cricket and call the sortdata methods an finally print it.

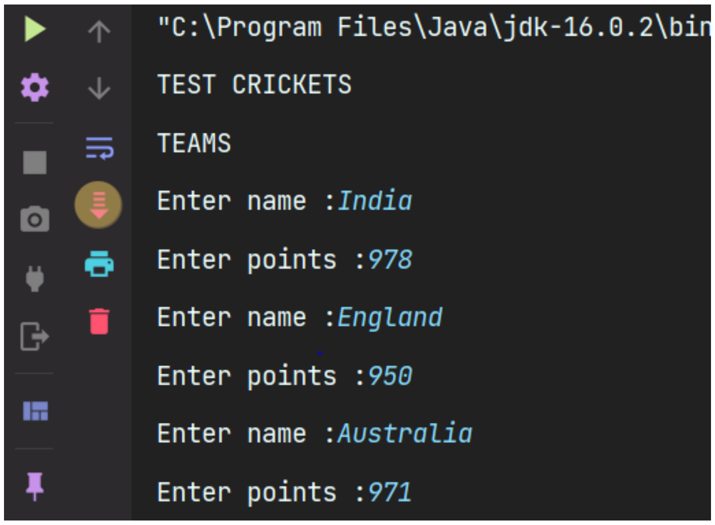
PROGRAM CODE :

```
package Java.Lab.lab3;
import java.util.Arrays;
import java.util.Comparator;
import java.util.Scanner;
class Pair{
   private String name;
   double points;
   public Pair(String name, double points) {
      this.name = name;
      this.points = points;
   }
   public String getName() {
      return name;
   }
```

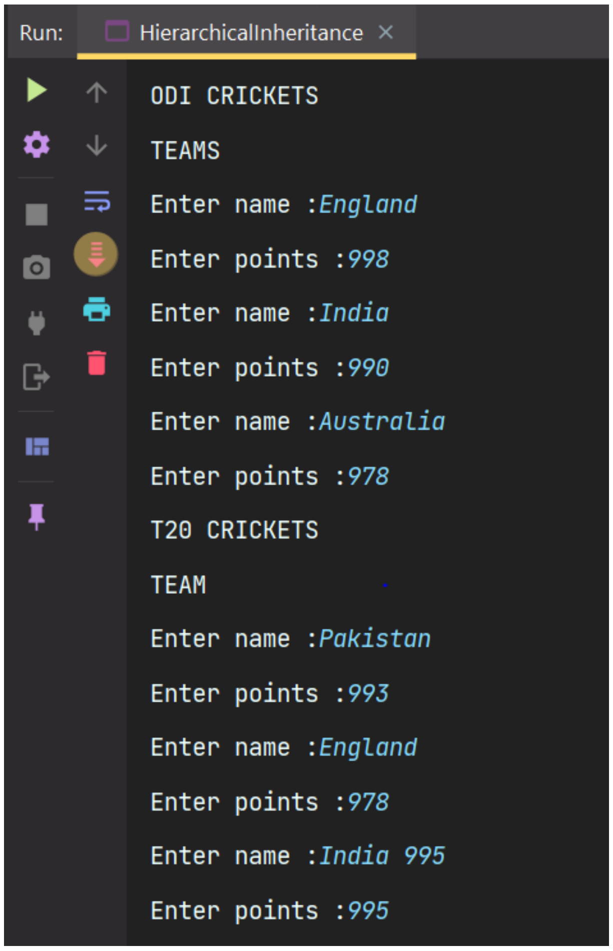
```
    public double getPoints() {
        return points;
    }
    @Override
    public String toString() {
        return "Name :" + name + " Points :" + points;
    }
}
class Cricket{
    protected String format;
    int numOvers ,numberOfDays;
    public Cricket(String format, int numOvers, int numberOfDays) {
        this.format = format;
        this.numOvers = numOvers;
        this.numberOfDays = numberOfDays;
    }
    public String getFormat() {
        return format;
    }
    public int getNumOvers() {
        return numOvers;
    }
    public int getNumberOfDays() {
        return numberOfDays;
    }
}
class TestCricket extends Cricket{
    Pair []teams;
    String name;
    double points;
    private final Scanner input = new Scanner(System.in);
    public Pair getValues(){
        System.out.print("Enter name :");
        name = input.nextLine();
        System.out.print("Enter points :");
        points = input.nextDouble();
        Pair value = new Pair(name,points);
        return value;
    }
    public TestCricket(String format, int numOvers, int numberOfDays) {
        super(format, numOvers, numberOfDays);
        System.out.println("TEST CRICKETS");
        teams = new Pair[3];
        System.out.println("TEAMS");
        for(int i = 0 ; i < 3 ; i++){
            teams[i] = getValues();
            input.nextLine();
        }
    }
    public void sortData(){
        Arrays.sort(teams, Comparator.comparing(Pair :: getPoints).reversed());
    }
    @Override
    public String toString(){
        return teams[0].toString() +" " +  teams[1].toString() + " " +  teams[2].toString() + "\n" ;
    }
}
class ODICricket extends Cricket{
    Pair [] teams;
    String name;
    double points;
    private final Scanner input = new Scanner(System.in);
    public Pair getValues(){
        System.out.print("Enter name :");
        name = input.nextLine();
        System.out.print("Enter points :");
        points = input.nextDouble();
        Pair value = new Pair(name,points);
        return value;
    }
    public ODICricket(String format, int numOvers, int numberOfDays) {
        super(format, numOvers, numberOfDays);
        System.out.println("ODI CRICKETS");
        teams = new Pair[3];
        System.out.println("TEAMS");
        for(int i = 0 ; i < 3 ; i++){
            teams[i] = getValues();
            input.nextLine();
        }
    }
    public void sortData(){
        Arrays.sort(teams, Comparator.comparing(Pair :: getPoints));
    }
    @Override
    public String toString(){
        return teams[0].toString() +" " +  teams[1].toString() + " " +  teams[2].toString() + "\n" ;
    }
}
class T20Cricket extends Cricket{
    Pair [] teams;
    String name;
    double points;
    private final Scanner input = new Scanner(System.in);
    public Pair getValues(){
        System.out.print("Enter name :");
        name = input.nextLine();
        System.out.print("Enter points :");
        points = input.nextDouble();
        Pair value = new Pair(name,points);
        return value;
    }
    public T20Cricket(String format, int numOvers, int numberOfDays) {
        super(format, numOvers, numberOfDays);
        System.out.println("T20 CRICKETS");
        teams = new Pair[3];
```

```
            System.out.println("TEAM");
            for(int i = 0 ; i < 3 ; i++){
                teams[i] = getValues();
                input.nextLine();
            }
        }
        public void sortData() {
            Arrays.sort(teams, Comparator.comparing(Pair::getPoints));
        }
        @Override
        public String toString(){
            return teams[0].toString() +" " +  teams[1].toString() + " " +  teams[2].toString() + "\n" ;
        }
    }
    public class HierarchicalInheritance {
        public static void main(String[] args) {
            TestCricket testCricket = new TestCricket("Test" ,90,5);
            ODICricket odiCricket = new ODICricket("ODI" ,50,1);
            T20Cricket t20Cricket = new T20Cricket("T20" , 20,1);
            testCricket.sortData();
            System.out.print(testCricket);
            odiCricket.sortData();
            System.out.print(odiCricket);
            t20Cricket.sortData();
            System.out.print(t20Cricket);
        }
}
```

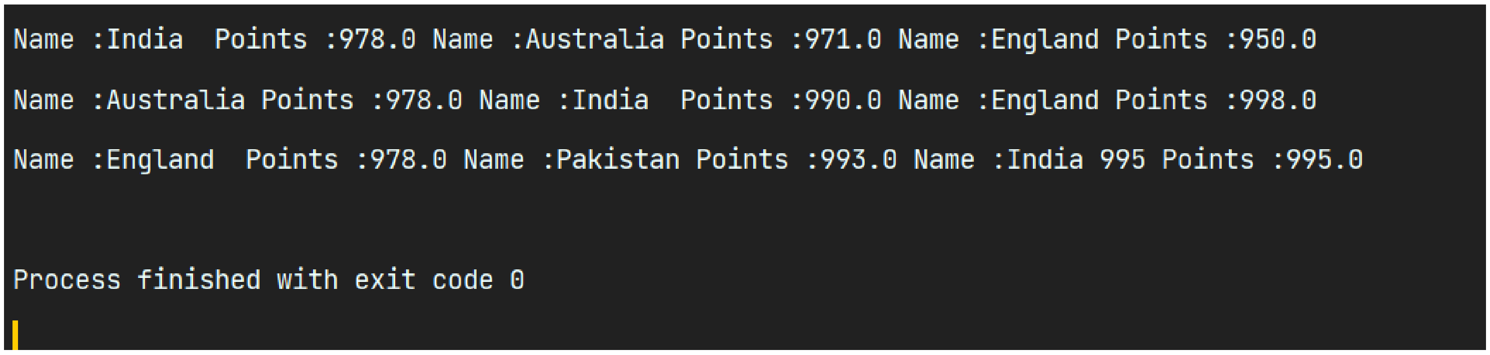
OUTPUT:



```
"C:\Program Files\Java\jdk-16.0.2\bin
TEST CRICKETS
TEAMS
Enter name :India
Enter points :978
Enter name :England
Enter points :950
Enter name :Australia
Enter points :971
```



```
ODI CRICKETS
TEAMS
Enter name :England
Enter points :998
Enter name :India
Enter points :990
Enter name :Australia
Enter points :978
T20 CRICKETS
TEAM
Enter name :Pakistan
Enter points :993
Enter name :England
Enter points :978
Enter name :India 995
Enter points :995
```


```
Name :India  Points :978.0 Name :Australia Points :971.0 Name :England Points :950.0

Name :Australia Points :978.0 Name :India  Points :990.0 Name :England Points :998.0

Name :England  Points :978.0 Name :Pakistan Points :993.0 Name :India 995 Points :995.0


Process finished with exit code 0
```

<u>RESULT:</u>

Thus, the program has been executed successfully.

MULTIPLE INHERITANCE IN JAVA :

- ✓ Multiple Inheritance is a feature of an object-oriented concept, where a class can inherit properties of more than one parent class.

- ✓ The problem occurs when there exist methods with the same signature in both the superclasses and subclass.

- ✓ On calling the method, the compiler cannot determine which class method to be called and even on calling which class method gets the priority.

- ✓ Multiple inheritance is not supported by Java using classes, handling the complexity that causes due to multiple inheritances is very complex.

- ✓ It creates problems during various operations like casting, constructor chaining, etc, and the above all reason is that there are very few scenarios on which we actually need multiple inheritances, so better to omit it for keeping things simple and straightforward.

SOLUTION :

- ✓ Java 8 supports default methods where interfaces can provide a default implementation of methods.

- ✓ And a class can implement two or more interfaces.

- ✓ In case both the implemented interfaces contain default methods with the same method signature, the implementing class should explicitly specify which default method is to be used, or it should override the default method.

4)**AIM :**

To Write a Compile time polymorphism program in java

ALGORITHM:

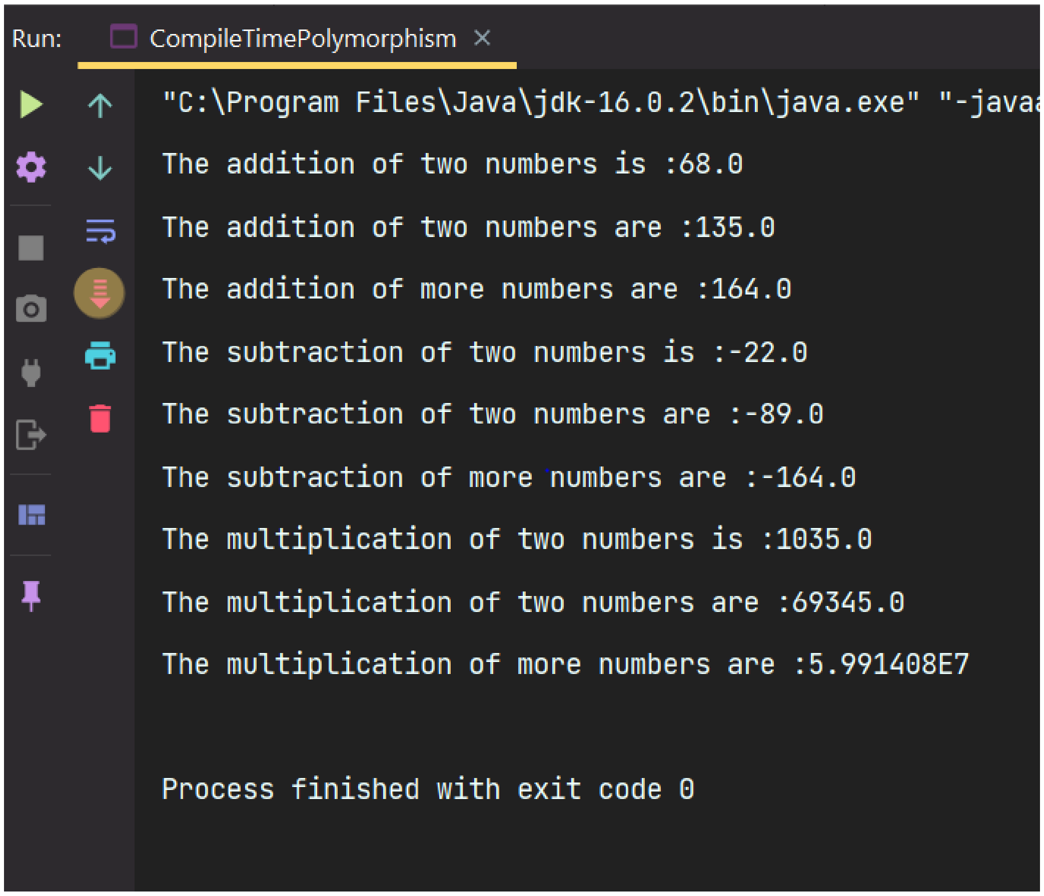
- ✓ Create a add method which excepts two parameters , three parameters ,and varialble length parameters add them and return
- ✓ Similarly create methods for sub,mul in the same as add and return accordingly

PROGRAM CODE :

```
package Java.Lab.lab3;

public class CompileTimePolymorphism {
   @SafeVarargs
   public static <T extends Number> T add(T ...b){
      Double sum = 0.00;
      for(int i = 0;  i < b.length ; i++){
         sum += b[i].doubleValue();
      }
      return (T) sum;
   }
   public static<T extends Number>  T add(T a, T b){
      Double sum = a.doubleValue() + b.doubleValue();
      return (T)sum;
   }
   public static<T extends Number> T add(T a , T b ,  T c){
      Double sum = a.doubleValue() + b.doubleValue() + c.doubleValue();
      return (T)sum;
   }

   public static<T extends Number>  T sub(T a, T b){
      Double sum = a.doubleValue() - b.doubleValue();
      return (T)sum;
   }
   public static<T extends Number> T sub(T a , T b ,  T c){
        Double sum = a.doubleValue() - b.doubleValue() - c.doubleValue();
        return (T)sum;
    }
    @SafeVarargs
    public static <T extends Number> T sub(T ...b){
        Double sum = 0.00;
        for(int i = 0 ; i < b.length ; i++){
            sum -= b[i].doubleValue();
        }
        return (T) sum;
    }
    public static<T extends Number>  T mul(T a, T b){
        Double sum = a.doubleValue() * b.doubleValue();
        return (T)sum;
    }
    public static<T extends Number> T mul(T a , T b ,  T c){
        Double sum = a.doubleValue() * b.doubleValue() * c.doubleValue();
        return (T)sum;
    }
    @SafeVarargs
    public static <T extends Number> T mul(T ...b){
        Double sum = 1.00;
        for(int i = 0 ; i < b.length ; i++){
            sum *= b[i].doubleValue();
        }
        return (T) sum;
    }
    public static void main(String[] args) {
        System.out.println("The addition of two numbers is :" + add(23,45));
        System.out.println("The addition of two numbers are :" + add(23,45,67));
        System.out.println("The addition of more numbers are :" + add(23,45,67,12,8,9));
        System.out.println("The subtraction of two numbers is :" + sub(23,45));
        System.out.println("The subtraction of two numbers are :" + sub(23,45,67));
        System.out.println("The subtraction of more numbers are :" + sub(23,45,67,12,8,9));
        System.out.println("The multiplication of two numbers is :" + mul(23,45));
```

```
            System.out.println("The multiplication of two numbers are :" + mul(23,45,67));
            System.out.println("The multiplication of more numbers are :" + mul(23,45,67,12,8,9));
        }
}
```

OUTPUT:



```
"C:\Program Files\Java\jdk-16.0.2\bin\java.exe" "-java
The addition of two numbers is :68.0
The addition of two numbers are :135.0
The addition of more numbers are :164.0
The subtraction of two numbers is :-22.0
The subtraction of two numbers are :-89.0
The subtraction of more numbers are :-164.0
The multiplication of two numbers is :1035.0
The multiplication of two numbers are :69345.0
The multiplication of more numbers are :5.991408E7


Process finished with exit code 0
```

RESULT:

Thus, the program has been executed successfully.

5)AIM :

To Write a polymorphism program in java

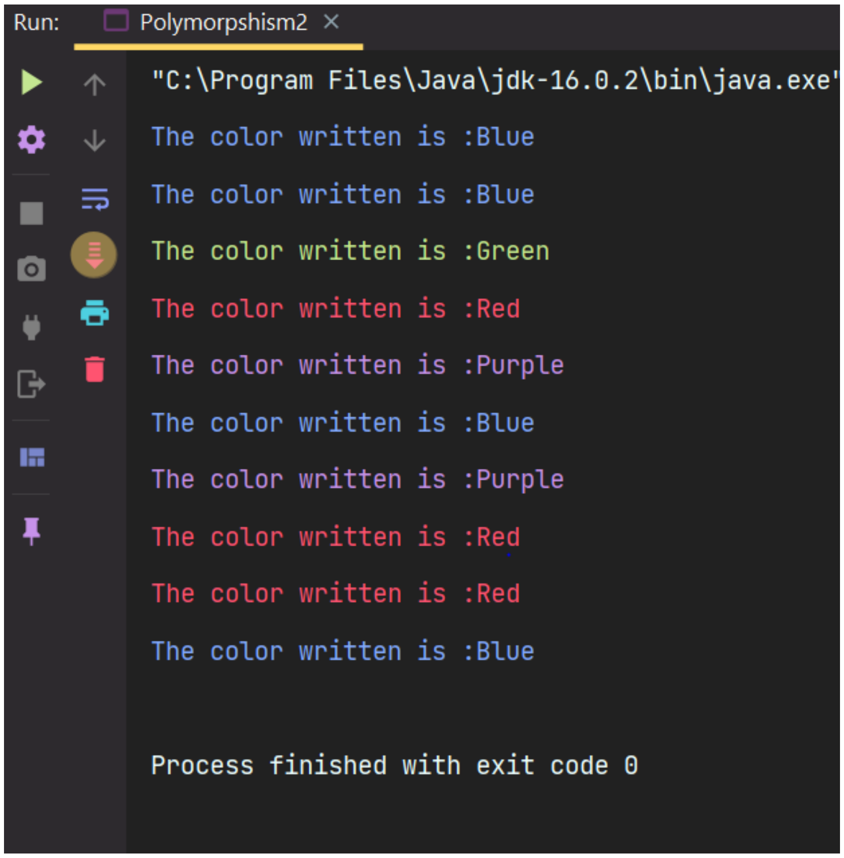
ALGORITHM:

- ✓ Create a class Color with fields like colorString,color
- ✓ Create constructor which intializes the variables and getter methods
- ✓ Create a class Green that extends from Programming Language in which the constructor Call the super and intializes the variables and getter methods which inturn prints the content in that particular color
- ✓ Create a class for other color like Blue , Purple , Red , Cyan which inherits from Color class and prints the content in that particular color.

PROGRAM CODE:

```
package Java.Lab.lab3;
class Color{
   protected String colorString,color;
   public Color(String colorString, String color) {
      this.colorString = colorString;
      this.color = color;
   }
   public String getColorString() {
      return colorString;
   }
   public String getColor() {
      return color;
   }
   @Override
   public String toString() {
      return color + "The color written is :" + colorString + " Ansi value is :" + getColor();
   }
}
class Green  extends Color{
   public Green(String colorString, String color) {
        super(colorString, color);
    }
    @Override
    public String getColorString() {
        return super.getColorString();
    }
    @Override
    public String getColor() {
        return super.getColor();
    }
    @Override
    public String toString() {
        return color + "The color written is :" + colorString;
    }
}
class Blue extends Color{
    public Blue(String colorString, String color) {
        super(colorString, color);
    }
    @Override
    public String getColorString() {
        return super.getColorString();
    }
    @Override
    public String getColor() {
        return super.getColor();
    }
    @Override
    public String toString() {
        return color + "The color written is :" + colorString;
    }
}
class Purple extends Color{
    public Purple(String colorString, String color) {
        super(colorString, color);
    }
    @Override
    public String getColorString() {
        return super.getColorString();
    }
    @Override
    public String getColor() {
        return super.getColor();
    }
    @Override
    public String toString() {
        return color + "The color written is :" + colorString;
    }
}
class Red extends Color{
    public Red(String colorString, String color) {
        super(colorString, color);
    }
    @Override
    public String getColorString() {
        return super.getColorString();
    }
    @Override
    public String getColor() {
        return super.getColor();
    }
    @Override
    public String toString() {
        return color + "The color written is :" + colorString;
    }
}
class Cyan extends Color{
    public Cyan(String colorString, String color) {
        super(colorString, color);
    }
    @Override
    public String getColorString() {
        return super.getColorString();
    }
    @Override
    public String getColor() {
        return super.getColor();
    }
    @Override
    public String toString() {
        return color + "The color written is :" + colorString;
    }
}
public class Polymorpshism2 {
    public static void main(String[] args) {
        Color color = new Color("Black","\u001B[30m");
        for(int i = 0 ; i < 10 ; i++){
            int randomNumber = (int)((Math.random()*5)+1);
            switch (randomNumber){
                case 1 -> {
                    color = new Green("Green","\u001B[32m");
                    System.out.println(color);
                }
                case 2 -> {
                    color = new Purple("Purple","\u001B[35m");
                    System.out.println(color);
                }
                case 3 -> {
                    color = new Blue("Blue" , "\u001B[34m");
                    System.out.println(color);
                }
                case 4 -> {
                    color = new Red("Red" , "\u001B[31m");
                    System.out.println(color);
                }
                case 5 -> {
                    color = new Cyan("Cyan" , "\u001B[36m");
                    System.out.println(color);
                }
                default -> {
                    System.out.println("Invalid Color!!!!");
                }
            }
        }
    }
}
```

OUTPUT:



```
"C:\Program Files\Java\jdk-16.0.2\bin\java.exe'
The color written is :Blue
The color written is :Blue
The color written is :Green
The color written is :Red
The color written is :Purple
The color written is :Blue
The color written is :Purple
The color written is :Red
The color written is :Red
The color written is :Blue

Process finished with exit code 0
```

RESULT:

Thus, the program has been executed successfully.

Early Binding vs Late Binding

COMPILE TIME POLYMORPHISM	RUN TIME POLYMORPHSIM
✓ In Compile time Polymorphism, the call is resolved by the compiler. ✓ It is also known as Static binding, Early binding and overloading as well. ✓ Method overloading is the compile-time polymorphism where more than one methods share the same name with different parameters or signature and different return type. ✓ It is achieved by function overloading and operator overloading. ✓ It provides fast execution because the method that needs to be executed is known early at the compile time ✓ Compile time polymorphism is less flexible as all things execute at compile time.	✓ In Run time Polymorphism, the call is not resolved by the compiler. ✓ It is also known as Dynamic binding, Late binding and overriding as well. ✓ Method overriding is the runtime polymorphism having same method with same parameters or signature, but associated in different classes. ✓ It is achieved by overiding since all the objects are refernces no need to create pointers and virtual functions like c++ ✓ It provides slow execution as compare to early binding because the method that needs to be executed is known at the runtime. ✓ Run time polymorphism is more flexible as all things execute at run time.