**EX NO: 3 CONSTRUCTORS AND METHODS**

**DATE: 29/08/2023**

**AIM**

To explore the concepts of Control Structures in Java.

**ALGORITHM:**

**STEP 1:**

* Create a class Circle with instance variables for radius, center coordinates (x and y), and a static variable for pi.
* Include a static method compareArea() to compare the areas of two circles and print the result.

**STEP 2:**

* Define a class Circle with instance variables for radius, center coordinates (x and y), and a static variable for pi.
* Implement a constructor to initialize variables, an instance method to calculate area, and a static method to compare two circles based on area.

**STEP 3:**

* Create a class Instancecounter3704 with a private static variable instanceCount and a private non-static variable nonStaticVar.
* Include a default constructor to increment instanceCount and initialize nonStaticVar, and methods to increment nonStaticVar and retrieve both counts.

**STEP 4:**

* Create a class MethodOverloadingDemo3704 with overloaded methods for addition and concatenation based on different parameter lists.
* In the main method, create an object and demonstrate method overloading with various argument types.

**STEP 5:**

* Define a class VariableMethodDemo with private instance and static variables.
* Implement instance and static methods to demonstrate the differences, and create objects to showcase their behavior.

**STEP 6:**

* Create an immutable class Person with private final variables for name and age.
* Include a constructor to set these variables and getter methods; demonstrate that state cannot be changed in the main method.

**STEP 7:**

* Create a class Clock for representing time with instance variables for hours, minutes, and seconds, along with constructors for default time, specified time, and seconds since midnight.
* Include methods for ticking, ticking down, and adding/subtracting clocks; demonstrate these operations in the main method.

1. **Write a program to create a class circle with centre and the radius as instance variables. Initialize and display its variables.**

**CODE**

import java.util.\*;

public class Circle3704 {

public static void main(String[] args){

System.out.println("2021503704");

System.out.println("Bhumisvara");

Scanner in=new Scanner(System.in);

System.out.println("Enter Radius 1:");

int rad1=in.nextInt();

System.out.println("Enter Radius 2:");

int rad2=in.nextInt();

Circle obj1=new Circle();

Circle obj2=new Circle();

obj1.radius=rad1;

obj2.radius=rad2;

float area1=obj1.calculateArea();

float area2=obj2.calculateArea();

System.out.println("Area of circle 1:"+ area1 + " Area of circle 2:"+ area2);

Circle.compareArea(rad1,rad2);

}

}

class Circle

{

int radius;

int x,y;

static float pi=3.14f;

Circle(){

this.radius=0;

this.x=0;

this.y=0;

System.out.println("Center:X="+ x + "Y="+ y + "Radius ="+ radius);

}

float calculateArea(){

float ans;

ans=pi\*radius\*radius;

return ans;

}

static void compareArea(int rad1,int rad2){

if(rad1>rad2){

System.out.println("Circle1 has more area");

}

else if(rad2>rad1){

System.out.println("Circle2 has more area");

}

else{

System.out.println("Equal area");

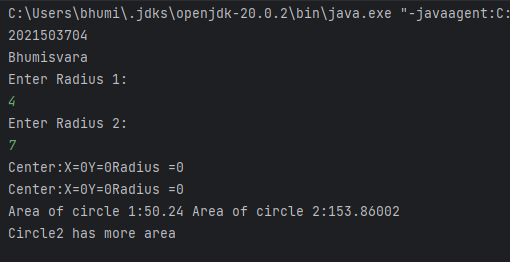
}

return;

}

}

**OUTPUT.**



**2. Write a program to display the use of**

1. **this keyword.**
2. **Default constructor**
3. **Parameterized constructor**
4. **Pass Object as an argument**
5. **Return object**

**CODE.**

import java.util.\*;

public class Circle3704 {

public static void main(String[] args){

System.out.println("2021503704");

System.out.println("Bhumisvara");

Scanner in=new Scanner(System.in);

System.out.println("Enter Radius 1:");

int rad1=in.nextInt();

System.out.println("Enter Radius 2:");

int rad2=in.nextInt();

Circle obj1=new Circle(rad1);

Circle obj2=new Circle(rad2);

float area1=obj1.calculateArea();

float area2=obj2.calculateArea();

System.out.println("Area of circle 1:"+ area1 + " Area of circle 2:"+ area2);

Circle res =Circle.compareObj(obj1,obj2);

System.out.println("Compare radius:"+ res.radius+ " " + "With area" + " "+res.area);

}

}

class Circle

{

int radius;

float area;

int x,y;

static float pi=3.14f;

Circle(int radius){

this.radius=radius;

this.x=0;

this.y=0;

this.area=0;

System.out.println("Radius ="+ radius);

}

float calculateArea(){

float ans;

ans=pi\*radius\*radius;

this.area=ans;

return ans;

}

static int compareArea(float rad1,float rad2){

if(rad1>rad2){

return 1;

}

else if(rad2>rad1){

return 2;

}

else{

return 0;

}

}

static Circle compareObj(Circle obj1,Circle obj2){

obj1.calculateArea();

obj2.calculateArea();

int x=compareArea(obj1.area,obj2.area);

if(x==1)

return obj1;

else if(x==2)

return obj2;

else{

System.out.println("Equal area");

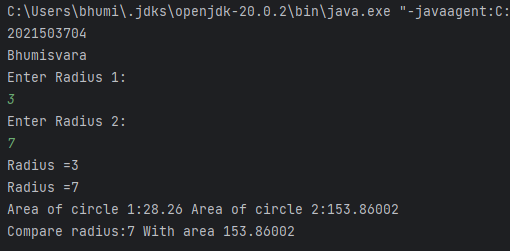
return obj1;

}

}

}

**OUTPUT.**



**3. Write a program to count the number of instances created for the class using static variable and the non static variable not visible to all the instances.**

**CODE.**

public class Instancecounter3704{

private static int instanceCount = 0;

private int nonStaticVar;

public Instancecounter3704() {

instanceCount++;

nonStaticVar = 0;

}

public void incrementNonStaticVar() {

nonStaticVar++;

}

public static int getInstanceCount() {

return instanceCount;

}

public int getNonStaticVar() {

return nonStaticVar;

}

public static void main(String[] args) {

System.out.println("2021503704");

System.out.println("Bhumisvara");

Instancecounter3704 obj1 = new Instancecounter3704();

obj1.incrementNonStaticVar();

Instancecounter3704 obj2 = new Instancecounter3704();

obj2.incrementNonStaticVar(); System.out.println("InstanceCount:"+Instancecounter3704.getInstanceCount());

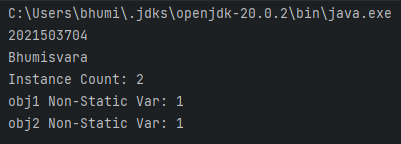
System.out.println("obj1Non-StaticVar: " + obj1.getNonStaticVar());

System.out.println("obj2Non-StaticVar: " + obj2.getNonStaticVar());

}

}

**OUTPUT**



4. **Write a program that implements method overloading(multiple methods in the same class can have the same name but different parameter lists) based on the following conditions**

1. **By changing number of arguments**
2. **By changing the data type of the arguments**
3. **Can we overload java main method?**
4. **Show that the method overloading is not possible by just changing the return type**
5. **Passing object as parameter**.

**CODE**

public class MethodOverloadingDemo3704 {

public int add(int a, int b) {

return a + b;

}

public double add(double a, double b) {

return a + b;

}

public int add(int a, double b) {

return (int) (a + b);

}

public double add(double a, int b) {

return a + b;

}

public String concatenate(String str1, String str2) {

return str1 + str2;

}

public static void main(String[] args) {

System.out.println("2021503704");

System.out.println("Bhumisvara");

MethodOverloadingDemo3704 demo = new MethodOverloadingDemo3704();

System.out.println("Method 1 (int): " + demo.add(5, 10));

System.out.println("Method 2 (double): " + demo.add(3.5, 2.5));

System.out.println("Method 3 (int + double): " + demo.add(8, 3.7));

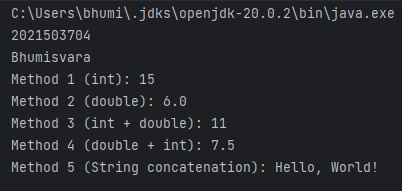
System.out.println("Method 4 (double + int): " + demo.add(2.5, 5));

System.out.println("Method 5 (String concatenation): " + demo.concatenate("Hello, ", "World!"));

}

}

**OUTPUT**



**5. Write a program that show the differences of**

1. **Instancevariables**
2. **Instancemethods**
3. **staticvariable**
4. **staticmethods**

**CODE**

public class VariableMethodDemo {

private int instanceVar;

private static int staticVar;

public VariableMethodDemo(int instanceVar) {

this.instanceVar = instanceVar;

}

public void instanceMethod() {

System.out.println("Inside instanceMethod()");

System.out.println("Instance variable: " + instanceVar);

System.out.println("Static variable: " + staticVar);

}

public static void staticMethod() {

System.out.println("Inside staticMethod()");

System.out.println("Static variable: " + staticVar) }

public static void main(String[] args) {

System.out.println("2021503704");

System.out.println("Bhumisvara");

VariableMethodDemo obj1 = new VariableMethodDemo(10);

VariableMethodDemo obj2 = new VariableMethodDemo(20);

staticVar = 100;

obj1.instanceMethod();

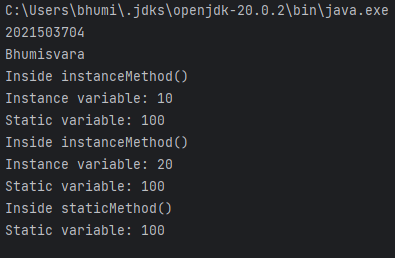
obj2.instanceMethod();

staticMethod();

}

}

**OUTPUT**



**6. Write a program to create an immutable class Person(state cannot be changed)**

1. **Define private final fields of name and age**
2. **Define a constructor to set the fields and a getter method to display the values.**
3. **Show that the state cannot be changed and enhances robustness**

**CODE**

public final class Person {

private final String name;

private final 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 static void main(String[] args) {

System.out.println("2021503704");

Person person = new Person("Bhumisvara", 21);

System.out.println("Name: " + person.getName());

System.out.println("Age: " + person.getAge());

System.out.println("Name: " + person.getName());

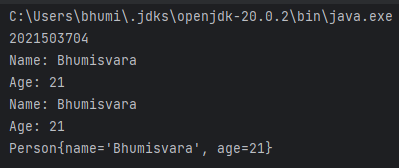
System.out.println("Age: " + person.getAge());

System.out.println(person);

}

}

**OUTPUT**



**7. Write a Java class Clock for dealing with the day time represented by hours, minutes, and seconds. Your class must have the following features:**

**• Three instance variables for the hours (range 0- 23), minutes(range 0- 59),and seconds(range 0-59).**

**• Three constructors:**

* **default(with no parameters passed; initialize the represented time to12:0:0)**
* **a constructor with three parameters: hours, minutes, and seconds.**
* **a constructor with one parameter: the value of time in seconds since midnight (it should be converted into the time value in hours, minutes,and seconds)**

**CODE**

import java.util.\*;

class Clock{

private int hours;

private int minutes;

private int seconds;

public Clock() {

this.hours = 12;

this.minutes = 0;

this.seconds = 0;

}

public Clock(int hours, int minutes, int seconds) {

this.hours = hours;

this.minutes = minutes;

this.seconds = seconds;

}

public Clock(int s) {

this.hours = s / 3600;

s%= 3600;

this.minutes = s / 60;

this.seconds = s % 60;

}

public int getHours() {

return hours;

}

public int getMinutes() {

return minutes;

}

public int getSeconds() {

return seconds;

}

public void setHours(int hours) {

this.hours = hours;

}

public void setMinutes(int minutes) {

this.minutes = minutes;

}

public void setSeconds(int seconds) {

this.seconds = seconds;

}

public void setClock(int s) {

this.hours = s/ 3600;

s %= 3600;

this.minutes = s / 60;

this.seconds = s % 60;

}

public void tick() {

seconds++;

if (seconds == 60) {

seconds = 0;

minutes++;

if (minutes == 60) {

minutes = 0;

hours++;

if (hours == 24) {

hours = 0;

}

}

}

}

public void tickDown() {

seconds--;

if (seconds < 0) {

seconds = 59;

minutes--;

if (minutes < 0) {

minutes = 59;

hours--;

if (hours < 0) {

hours = 23;

}

}

}

}

public void addClock(Clock otherClock) {

int totalSeconds = this.hours \* 3600 + this.minutes \* 60 + this.seconds

+ otherClock.hours \* 3600 + otherClock.minutes \* 60 + otherClock.seconds;

this.hours = totalSeconds / 3600;

totalSeconds %= 3600;

this.minutes = totalSeconds / 60;

this.seconds = totalSeconds % 60;

}

public Clock subtractClock(Clock otherClock) {

int thisTotalSeconds = this.hours \* 3600 + this.minutes \* 60 + this.seconds;

int otherTotalSeconds = otherClock.hours \* 3600 + otherClock.minutes \* 60 + otherClock.seconds;

int diffSeconds = thisTotalSeconds - otherTotalSeconds;

if (diffSeconds < 0) {

diffSeconds += 86400; // Adding a day's worth of seconds (24 \* 60 \* 60)

}

return new Clock(diffSeconds);

}

}

public class Clock3704 {

public static void main(String[] args) {

System.out.println("2021503704");

System.out.println("Bhumisvara");

Scanner scanner = new Scanner(System.in);

System.out.print("Enter seconds since midnight for the first clock: ");

int secondsSinceMidnight1 = scanner.nextInt();

Clock clock1 = new Clock(secondsSinceMidnight1);

System.out.print("Enter hours, minutes, and seconds for the second clock: ");

int hours2 = scanner.nextInt();

int minutes2 = scanner.nextInt();

int seconds2 = scanner.nextInt();

Clock clock2 = new Clock(hours2, minutes2, seconds2);

System.out.println("First Clock:");

printClock(clock1);

System.out.println("Second Clock:");

printClock(clock2);

Clock thirdClock = clock1.subtractClock(clock2);

System.out.println("Third Clock (Difference):");

printClock(thirdClock);

}

public static void printClock(Clock clock) {

System.out.println("Hours: " + clock.getHours());

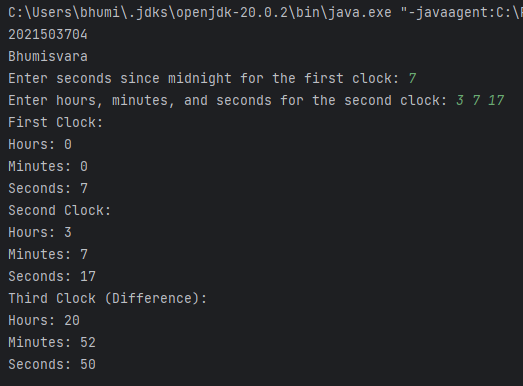
System.out.println("Minutes: " + clock.getMinutes());

System.out.println("Seconds: " + clock.getSeconds());

}

}

**OUTPUT**



**RESULT**

Constructors and methods have been implemented successfully.