|  |
| --- |
| Exp: 06 Date: 30/08/2023  JAVA PROGRAMMING LAB 4 |

Name: VIJAI SURIA M

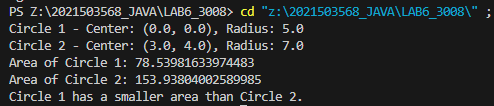
Reg No. : 2021503568

**Constructors and methods**

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

* Modify the exercise to have a constructor in class circle to initialize its variables.
* Modify the exercise to define the instance method calculateArea() to calculate the area and a static method compareArea() to compare the area of the circle and declare the result as smaller than or larger than or equal

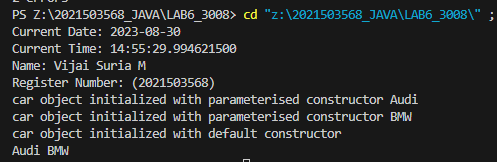
|  |
| --- |
| class Circle {  double centerX;  double centerY;  double radius;  public Circle(double centerX, double centerY, double radius) {  this.centerX = centerX;  this.centerY = centerY;  this.radius = radius;  }  public double calculateArea() {  return Math.PI \* radius \* radius;  }  public static String compareArea(Circle circle1, Circle circle2) {  double area1 = circle1.calculateArea();  double area2 = circle2.calculateArea();  if (area1 < area2) {  return "Circle 1 has a smaller area than Circle 2.";  } else if (area1 > area2) {  return "Circle 1 has a larger area than Circle 2.";  } else {  return "Circle 1 and Circle 2 have equal areas.";  }  }  }  public class CircleArea3568 {  public static void main(String[] args) {  Circle circle1 = new Circle(0, 0, 5);  Circle circle2 = new Circle(3, 4, 7);  System.out.println("Circle 1 - Center: (" + circle1.centerX + ", " + circle1.centerY + "), Radius: " + circle1.radius);  System.out.println("Circle 2 - Center: (" + circle2.centerX + ", " + circle2.centerY + "), Radius: " + circle2.radius);  System.out.println("Area of Circle 1: " + circle1.calculateArea());  System.out.println("Area of Circle 2: " + circle2.calculateArea());  System.out.println(Circle.compareArea(circle1, circle2));  }  } |



2. Write a program to display the use of

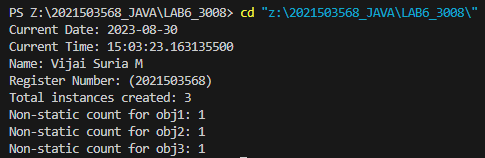
* this keyword.
* Default constructor
* Parameterized constructor
* Pass Object as an argument
* Return object

|  |
| --- |
| import java.time.LocalDate;  import java.time.LocalTime;  class Car {  public String myCar;  Car(){  this.myCar="";  System.out.println("car object initialized with default constructor " + myCar);  }  Car(String myCar){  this.myCar=myCar;  System.out.println("car object initialized with parameterised constructor " + myCar);  }  }  public class Question2 {  public static Car mergeObject(Car obj1, Car obj2){  Car result = new Car();  result.myCar = obj1.myCar + " " + obj2.myCar;  return result;  }  public static void main(String[] args) {  System.out.println("Current Date: " + LocalDate.now());  System.out.println("Current Time: " + LocalTime.now());  System.out.println("Name: Vijai Suria M \nRegister Number: (2021503568)");  Car obj1 = new Car("Audi");  Car obj2 = new Car("BMW");  Car result = mergeObject(obj1,obj2);  System.out.println(result.myCar);  }  } |



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.

|  |
| --- |
| import java.time.LocalDate;  import java.time.LocalTime;  class InstanceCounter {  private static int instanceCount = 0;  private int nonStaticCount = 0;    public InstanceCounter() {  instanceCount++;  nonStaticCount++;  }    public static int getInstanceCount() {  return instanceCount;  }    public int getNonStaticCount() {  return nonStaticCount;  }  }  public class InstanceCounter3568 {  public static void main(String[] args) {  System.out.println("Current Date: " + LocalDate.now());  System.out.println("Current Time: " + LocalTime.now());  System.out.println("Name: Vijai Suria M \nRegister Number: (2021503568)");  InstanceCounter obj1 = new InstanceCounter();  InstanceCounter obj2 = new InstanceCounter();  InstanceCounter obj3 = new InstanceCounter();    System.out.println("Total instances created: " + InstanceCounter.getInstanceCount());  System.out.println("Non-static count for obj1: " + obj1.getNonStaticCount());  System.out.println("Non-static count for obj2: " + obj2.getNonStaticCount());  System.out.println("Non-static count for obj3: " + obj3.getNonStaticCount());  }  } |



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

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

5. Write a program that show the differences of

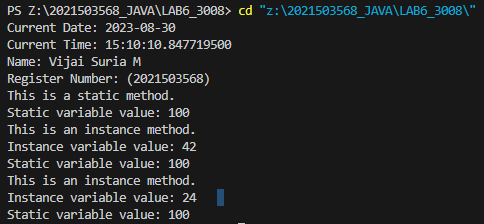
a. Instancevariables

b. Instancemethods

c. staticvariable

d. staticmethods

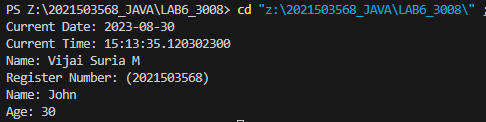
|  |
| --- |
| import java.time.LocalDate;  import java.time.LocalTime;  class Example {  int instanceVar;  static int staticVar;  void instanceMethod() {  System.out.println("This is an instance method.");  System.out.println("Instance variable value: " + instanceVar);  System.out.println("Static variable value: " + staticVar);  }  static void staticMethod() {  System.out.println("This is a static method.");  // Instance variables cannot be accessed directly in a static method.  // System.out.println("Instance variable value: " + instanceVar); // This will cause an error.  System.out.println("Static variable value: " + staticVar);  }  }  public class Question5 {  public static void main(String[] args) {  System.out.println("Current Date: " + LocalDate.now());  System.out.println("Current Time: " + LocalTime.now());  System.out.println("Name: Vijai Suria M \nRegister Number: (2021503568)");  Example.staticVar = 100;  Example.staticMethod();  Example obj1 = new Example();  obj1.instanceVar = 42;  obj1.instanceMethod();  Example obj2 = new Example();  obj2.instanceVar = 24;  obj2.instanceMethod();  }  } |



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

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

|  |
| --- |
| import java.time.LocalDate;  import java.time.LocalTime;  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;  }  }  public class Question6 {  public static void main(String[] args) {  System.out.println("Current Date: " + LocalDate.now());  System.out.println("Current Time: " + LocalTime.now());  System.out.println("Name: Vijai Suria M \nRegister Number: (2021503568)");  Person person = new Person("John", 30);  System.out.println("Name: " + person.getName());  System.out.println("Age: " + person.getAge());  }  } |



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)

Instance methods:

* A set-method method setClock() with one parameter seconds since midnight (to be converted into the time value in hours, minutes, and seconds as above).
* get-methods : getHours(), getMinutes(), getSeconds() with no parameters that return the corresponding values.
* set-methods : setHours(), setMinutes(), setSeconds() with one parameter each that setup the corresponding instance variables.
* method tick() with no parameters that increments the time stored in a Clock object by one second.
* method addClock() accepting an object of type Clock as a parameter.The method should add the time represented by the parameter class to the time represented in the current class.
* Add an instance method tickDown() which decrements the time stored in a Clock object by one second.
* Add an instance method subtractClock() that takes one Clock parameter and returns the difference between the time represented in the current

Clock object and the one represented by the Clock parameter. Difference of time should be returned as an clock object.

Write a separate class Clock Demo with a main() method. The program should:

* Instantiate a Clock object first Clock using one integer seconds since

midnight obtained from the keyboard.

* Print both clock object
* Create a reference thirdClock that should reference to object of difference of first Clock and second Clock by calling the method subtractClock()