|  |
| --- |
| **Ex.No: 5 DATE: 30-08-23**  **CONSTRUCTOR AND METHODS** |

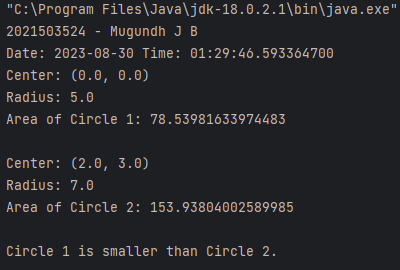
1) Write a program to create a class circle with centre and the radius as instance variables.

* 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

Code:

import java.lang.Math;  
import java.time.LocalDate;  
import java.time.LocalTime;  
  
class Circle {  
 double centerX;  
 double centerY;  
 double radius;  
  
 // Constructor to initialize circle's properties  
 Circle(double x, double y, double r) {  
 centerX = x;  
 centerY = y;  
 radius = r;  
 }  
  
 // Display circle's center and radius  
 void display() {  
 System.*out*.println("Center: (" + centerX + ", " + centerY + ")");  
 System.*out*.println("Radius: " + radius);  
 }  
  
 // Calculate the area of the circle  
 double calculateArea() {  
 return Math.*PI* \* radius \* radius;  
 }  
  
 // Compare areas of two circles and return comparison result  
 static String compareArea(Circle c1, Circle c2) {  
 double area1 = c1.calculateArea();  
 double area2 = c2.calculateArea();  
  
 if (area1 < area2) {  
 return "Circle 1 is smaller than Circle 2.";  
 } else if (area1 > area2) {  
 return "Circle 1 is larger than Circle 2.";  
 } else {  
 return "Circle 1 is equal in area to Circle 2.";  
 }  
 }  
}  
  
public class Circle3524 {  
 public static void main(String[] args) {  
 System.*out*.println("2021503524 - Mugundh J B");  
 System.*out*.println("Date: " + LocalDate.*now*() + " Time: " + LocalTime.*now*());  
 // Create two circles  
 Circle circle1 = new Circle(0, 0, 5);  
 Circle circle2 = new Circle(2, 3, 7);  
 // Display circle information and area for circle 1  
 circle1.display();  
 System.*out*.println("Area of Circle 1: " + circle1.calculateArea());  
 System.*out*.println();  
 // Display circle information and area for circle 2  
 circle2.display();  
 System.*out*.println("Area of Circle 2: " + circle2.calculateArea());  
 System.*out*.println();  
 // Compare and display the comparison result of circle areas  
 System.*out*.println(Circle.*compareArea*(circle1, circle2));  
 }  
}

Output:



2) Write a program to display the use of

• this keyword.

• Default constructor

• Parameterized constructor

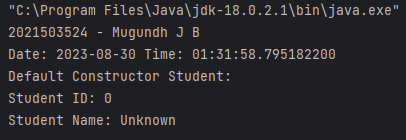
• Pass Object as an argument

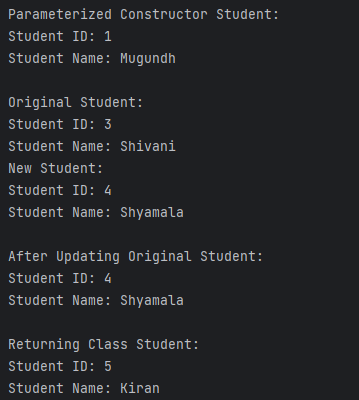
• Return object

Code:

import java.time.LocalDate;  
import java.time.LocalTime;  
  
class Student{  
 int id;  
 String name;  
 // using this keyword  
 Student(){  
 this.id = 0;  
 this.name = "Unknown";  
 }  
 // using this keyword  
 Student(int id, String name){  
 this.id = id;  
 this.name = name;  
 }  
 void display() {  
 System.*out*.println("Student ID: " + id);  
 System.*out*.println("Student Name: " + name);  
 }  
 // Static method to return an object  
 static Student createStudent(int id, String name) {  
 Student student = new Student(id, name);  
 return student;  
 }  
  
 // Method to pass an object as an argument  
 void updateStudent(Student newStudent) {  
 this.id = newStudent.id;  
 this.name = newStudent.name;  
 }  
}  
public class Student3524 {  
 public static void main(String[] args){  
 System.*out*.println("2021503524 - Mugundh J B");  
 System.*out*.println("Date: " + LocalDate.*now*() + " Time: " + LocalTime.*now*());  
  
 // Using default constructor  
 Student defaultStudent = new Student();  
 System.*out*.println("Default Constructor Student:");  
 defaultStudent.display();  
 System.*out*.println();  
  
 // Using parameterized constructor  
 Student paramStudent = new Student(1, "Mugundh");  
 System.*out*.println("Parameterized Constructor Student:");  
 paramStudent.display();  
 System.*out*.println();  
  
 // Passing an object as an argument  
 Student originalStudent = new Student(3, "Shivani");  
 Student newStudent = new Student(4, "Shyamala");  
 System.*out*.println("Original Student:");  
 originalStudent.display();  
 System.*out*.println("New Student:");  
 newStudent.display();  
 originalStudent.updateStudent(newStudent);  
  
 System.*out*.println("\nAfter Updating Original Student:");  
 originalStudent.display();  
 System.*out*.println();  
  
 // Returning an object  
 Student returnedStudent = Student.*createStudent*(5, "Kiran");  
 System.*out*.println("Returning Class Student:");  
 returnedStudent.display();  
 }  
}

Output:



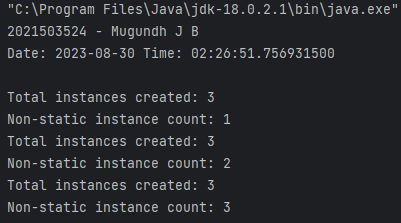


3) Write a program to count the number of instances created for the classusing static variable and the non static variable not visible to all the instances.

Code:

import java.time.LocalDate;  
import java.time.LocalTime;  
  
class Counter {  
 private static int *instanceCount* = 0; // Static variable to count instances  
 private int nonStaticInstanceCount; // Non-static variable not visible to all instances  
  
 public Counter() {  
 *instanceCount*++; // Increment instance count for each new instance  
 nonStaticInstanceCount = *instanceCount*; // Initialize non-static instance count for each new instance  
 }  
  
 public void displayCounts() {  
 System.*out*.println("Total instances created: " + *instanceCount*);  
 System.*out*.println("Non-static instance count: " + nonStaticInstanceCount);  
 }  
}  
  
public class Instance3524 {  
 public static void main(String[] args) {  
 System.*out*.println("2021503524 - Mugundh J B");  
 System.*out*.println("Date: " + LocalDate.*now*() + " Time: " + LocalTime.*now*());  
 System.*out*.println();  
 Counter counter1 = new Counter();  
 Counter counter2 = new Counter();  
 Counter counter3 = new Counter();  
  
 counter1.displayCounts();  
 counter2.displayCounts();  
 counter3.displayCounts();  
 }  
}

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

• By changing number of arguments

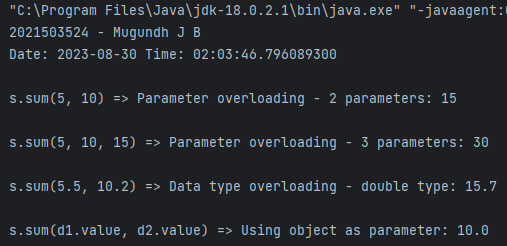
• By changing the data type of the arguments

• Passing object as parameter.

Code:

import java.time.LocalDate;  
import java.time.LocalTime;  
  
class Sum {  
 // Method to sum two integers  
 public int sum(int x, int y) {  
 return (x + y);  
 }  
  
 // Method to sum three integers  
 public int sum(int x, int y, int z) {  
 return (x + y + z);  
 }  
  
 // Method to sum two doubles  
 public double sum(double x, double y) {  
 return (x + y);  
 }  
}  
  
class Data {  
 double value;  
  
 // Constructor to initialize Data object  
 public Data(double value) {  
 this.value = value;  
 }  
}  
  
public class Overloading3524 {  
 public static void main(String[] args) {  
 System.*out*.println("2021503524 - Mugundh J B");  
 System.*out*.println("Date: " + LocalDate.*now*() + " Time: " + LocalTime.*now*());  
 Sum s = new Sum();  
  
 // Parameter overloading with two and three parameters  
 System.*out*.println("\ns.sum(5, 10) => " + "Parameter overloading - 2 parameters: " + s.sum(5, 10));  
 System.*out*.println("\ns.sum(5, 10, 15) => " + "Parameter overloading - 3 parameters: " + s.sum(5, 10, 15));  
  
 // Data type overloading with two doubles  
 System.*out*.println("\ns.sum(5.5, 10.2) => " + "Data type overloading - double type: " + s.sum(5.5, 10.2));  
  
 // Create Data objects  
 Data d1 = new Data(7.5);  
 Data d2 = new Data(2.5);  
  
 // Using object as a parameter  
 double result = s.sum(d1.value, d2.value);  
 System.*out*.println("\ns.sum(d1.value, d2.value) => " + "Using object as parameter: " + result);  
 }  
}

Output:



• Can we overload java main method?

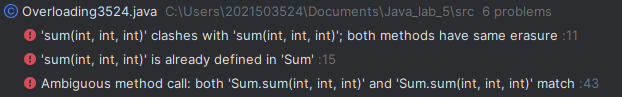
Ans: No, you cannot overload the main method in Java. The main method is the entry point of a Java program, and it must have the exact signature:

public static void main(String[] args)

• Show that the method overloading is not possible by just changing the return type

public int sum(int x, int y, int z) {  
 return (x + y + z);  
}  
public double sum(int x, int y, int z) {  
 return (x + y + z);  
}

Output:

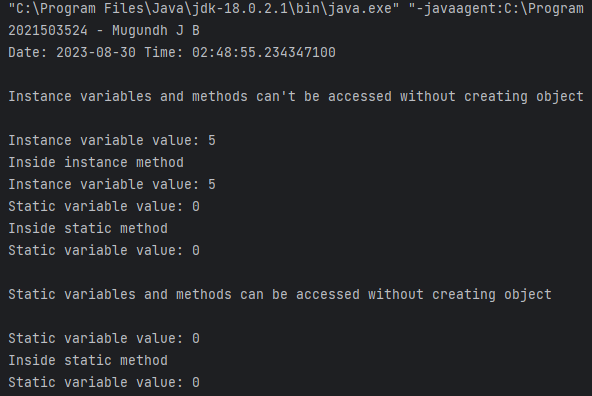


5)

Code:

import java.time.LocalDate;  
import java.time.LocalTime;  
  
class Dummy {  
 // Instance variable  
 int instanceVar;  
 // Static variable  
 static int *staticVar*;  
 public Dummy(int instanceVar) {  
 this.instanceVar = instanceVar;  
 }  
  
 // Instance method  
 void instanceMethod() {  
 System.*out*.println("Inside instance method");  
 System.*out*.println("Instance variable value: " + instanceVar);  
 System.*out*.println("Static variable value: " + *staticVar*);  
 *staticMethod*(); // Calling static method from instance method  
 }  
  
 // Static method  
 static void staticMethod() {  
 System.*out*.println("Inside static method");  
 // System.out.println("Instance variable value: " + instanceVar);  
 // Error: Cannot access instance variable from static method  
 System.*out*.println("Static variable value: " + *staticVar*);  
 }  
}  
  
public class Differences3524 {  
 public static void main(String[] args) {  
 System.*out*.println("2021503524 - Mugundh J B");  
 System.*out*.println("Date: " + LocalDate.*now*() + " Time: " + LocalTime.*now*());  
 Dummy obj = new Dummy(5);  
  
 // Access instance variable and instance method  
 System.*out*.println("\nInstance variables and methods can't be accessed without creating object\n");  
 System.*out*.println("Instance variable value: " + obj.instanceVar);  
 obj.instanceMethod();  
  
 // Access static variable and static method  
 System.*out*.println("\nStatic variables and methods can be accessed without creating object\n");  
 System.*out*.println("Static variable value: " + Dummy.*staticVar*);  
 Dummy.*staticMethod*();  
 }  
}

Output:



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.

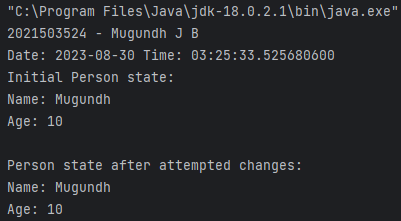
• Show that the state cannot be changed and enhances robustness

Code:

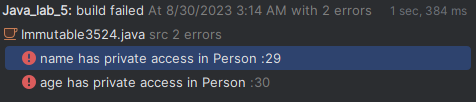
class Person {  
 private final String name;  
 private final int age;  
  
 public Person(String name, int age) {  
 this.name = name;  
 this.age = age;  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public int getAge() {  
 return age;  
 }  
}  
  
public class Immutable3524 {  
 public static void main(String[] args) {

System.*out*.println("2021503524 - Mugundh J B");  
 System.*out*.println("Date: " + LocalDate.*now*() + " Time: " + LocalTime.*now*());  
 try {  
 Person person = new Person("Mugundh", 10);  
  
 System.*out*.println("Initial Person state:");  
 System.*out*.println("Name: " + person.getName());  
 System.*out*.println("Age: " + person.getAge());  
  
 // Intentionally attempt to change the state (will trigger an error)  
 // person.name = "Hello";  
 // person.age = 0;  
  
 // Display state after the attempted changes  
 System.*out*.println("\nPerson state after attempted changes:");  
 System.*out*.println("Name: " + person.getName());  
 System.*out*.println("Age: " + person.getAge());  
 } catch (Exception e) {  
 System.*out*.println("Error: " + e.getMessage());  
 }  
 }  
}

Output:



When attempting to change:



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: o default(with no parameters passed; initialize the represented time to12:0:0) o a constructor with three parameters: hours, minutes, and seconds. o 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: o 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). o get-methods : getHours(), getMinutes(), getSeconds() with no parameters that return the corresponding values. o set-methods : setHours(), setMinutes(), setSeconds() with one parameter each that setup the corresponding instance variables. o method tick() with no parameters that increments the time stored in a Clock object by one second. o 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. o Add an instance method tickDown() which decrements the time stored in a Clock object by one second. o 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()

Code:

import java.util.Scanner;  
class Clock {  
 private int hours, minutes, seconds;  
  
 public Clock() {  
 this(12, 0, 0); // Set default time to 12:00:00  
 }  
  
 public Clock(int hours, int minutes, int seconds) {  
 setClock(hours, minutes, seconds);  
 }  
  
 public Clock(int secondsSinceMidnight) {  
 this(secondsSinceMidnight / 3600, (secondsSinceMidnight % 3600) / 60, secondsSinceMidnight % 60);  
 }  
  
 // Set time values while checking for validity  
 public void setClock(int hours, int minutes, int seconds) {  
 if (hours >= 0 && hours < 24 && minutes >= 0 && minutes < 60 && seconds >= 0 && seconds < 60) {  
 this.hours = hours;  
 this.minutes = minutes;  
 this.seconds = seconds;  
 } else throw new IllegalArgumentException("Invalid time values.");  
 }  
  
 // Get and set methods for hours, minutes, and seconds  
 public int getHours() { return hours; }  
 public int getMinutes() { return minutes; }  
 public int getSeconds() { return seconds; }  
 public void setHours(int hours) { if (hours >= 0 && hours < 24) this.hours = hours; else throw new IllegalArgumentException("Invalid hours value."); }  
 public void setMinutes(int minutes) { if (minutes >= 0 && minutes < 60) this.minutes = minutes; else throw new IllegalArgumentException("Invalid minutes value."); }  
 public void setSeconds(int seconds) { if (seconds >= 0 && seconds < 60) this.seconds = seconds; else throw new IllegalArgumentException("Invalid seconds value."); }  
  
 // Increment time by one second  
 public void tick() { seconds++; if (seconds >= 60) { seconds = 0; minutes++; if (minutes >= 60) { minutes = 0; hours = (hours + 1) % 24; } } }  
  
 // Decrement time by one second  
 public void tickDown() { seconds--; if (seconds < 0) { seconds = 59; minutes--; if (minutes < 0) { minutes = 59; hours = (hours + 23) % 24; } } }  
  
 // Add time from another Clock instance  
 public void addClock(Clock other) { seconds += other.seconds; minutes += other.minutes; hours += other.hours; normalizeTime(); }  
  
 // Subtract time from another Clock instance and return the difference as a new Clock  
 public Clock subtractClock(Clock other) {  
 int totalThis = hours \* 3600 + minutes \* 60 + seconds, totalOther = other.hours \* 3600 + other.minutes \* 60 + other.seconds;  
 return new Clock(Math.*abs*(totalThis - totalOther));  
 }  
  
 // Normalize the time to ensure valid values  
 private void normalizeTime() { minutes += seconds / 60; seconds %= 60; hours = (hours + minutes / 60) % 24; minutes %= 60; }  
  
 // String representation of the time in HH:MM:SS format  
 public String toString() { return String.*format*("%02d:%02d:%02d", hours, minutes, seconds); }  
}  
  
// Demonstrate the Clock class with user input  
public class Clock3524 {  
 public static void main(String[] args) {

System.*out*.println("2021503524 - Mugundh J B");  
 System.*out*.println("Date: " + LocalDate.*now*() + " Time: " + LocalTime.*now*());  
 Scanner scanner = new Scanner(System.*in*);  
 System.*out*.print("Enter seconds since midnight for first Clock: ");  
 int sec1 = scanner.nextInt();  
 Clock c1 = new Clock(sec1);  
 System.*out*.println("First Clock: " + c1);  
  
 // Create a second Clock instance  
 Clock c2 = new Clock(15, 30, 45);  
 System.*out*.println("Second Clock: " + c2);  
  
 // Calculate and display the difference between the two Clock instances  
 System.*out*.println("Difference Clock (Third Clock): " + c1.subtractClock(c2));  
  
 scanner.close();  
 }  
}

Output:

