

**JAVA PROGRAMMING**

EXPERIMENT-1



**NAME : SIDDIQSHA A**

**DEPARTMENT : CSE-AIML**

**DATE : 23.08.2024**

EXPERIMENT-1

**Q1.Write a Java program to find out the number of objects created of a Class**

**AIM:**

To write a Java program that counts the number of objects created for a particular class using a static counter variable.

**ALGORITHM:**

1. **Start**
2. Initialize a static variable objectCount to 0.
3. Create a class MyClass.
4. In the constructor of MyClass, increment objectCount by 1 each time an object is created.
5. Define a static method getObjectCount() that returns the value of objectCount.
6. In the main method:
7. Create multiple objects of MyClass.
8. Print the value returned by getObjectCount().
9. **End\**

**CODE:**

class MyClass {

private static int objectCount = 0;

public MyClass() {

objectCount++;

}

public static int getObjectCount() {

return objectCount;

}

}

public class ObjectCounter {

public static void main(String[] args) {

MyClass obj1 = new MyClass();

MyClass obj2 = new MyClass();

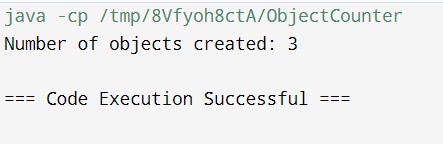
MyClass obj3 = new MyClass();

System.out.println("Number of objects created: " + MyClass.getObjectCount());

}

}

**OUTPUT:**



**RESULT:**

The program successfully tracks and displays the number of objects created for the class.

**Q2.** **Write a Java program to calculate difference between Two Time Periods without using inbuilt functions**

**AIM:**

To write a Java program that calculates the difference between two time periods (hours, minutes, and seconds) without using any inbuilt functions.

**ALGORITHM:**

1. **Start**
2. Define a Time class with attributes hours, minutes, and seconds.
3. Implement a constructor to initialize the time attributes.
4. Create a static method difference() that takes two Time objects as parameters.
5. Subtract the seconds, minutes, and hours of the first time from the second.
6. If seconds are negative, add 60 to seconds and subtract 1 from minutes.
7. If minutes are negative, add 60 to minutes and subtract 1 from hours.
8. Return the difference as a new Time object.
9. In the main method:
10. Create two Time objects for the start and stop times.
11. Call the difference() method and store the result.
12. Print the difference.
13. **End**

**CODE:**

class TimePeriod {

int hours;

int minutes;

int seconds;

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

this.hours = hours;

this.minutes = minutes;

this.seconds = seconds;

}

public static TimePeriod difference(TimePeriod start, TimePeriod end) {

TimePeriod diff = new TimePeriod(0, 0, 0);

if (end.seconds < start.seconds) {

end.seconds += 60;

end.minutes--;

}

diff.seconds = end.seconds - start.seconds;

if (end.minutes < start.minutes) {

end.minutes += 60;

end.hours--;

}

diff.minutes = end.minutes - start.minutes;

diff.hours = end.hours - start.hours;

return diff;

}

}

public class Main {

public static void main(String[] args) {

TimePeriod start = new TimePeriod(5, 45, 30);

TimePeriod end = new TimePeriod(8, 20, 10);

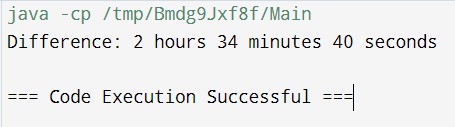
TimePeriod diff = TimePeriod.difference(start, end);

System.out.println("Difference: " + diff.hours + " hours " + diff.minutes + " minutes " + diff.seconds + " seconds");

}

}

**OUTPUT:**



**RESULT:**

The program correctly calculates and displays the time difference between the two given time periods

**Q3.** **Write a Java Program to Check Automorphic Number.**

**AIM:**

To write a Java program that checks if a given number is an Automorphic number (a number whose square ends in the number itself).

**ALGORITHM:**

1. **Start**
2. Accept an integer input number.
3. Calculate the square of the number and store it in square.
4. Convert number and square to strings (numStr and squareStr).
5. Check if squareStr ends with numStr.
6. If true, print that the number is Automorphic.
7. Else, print that the number is not Automorphic.
8. **End**

**CODE:**

import java.util.Scanner;

public class AutomorphicNumberChecker {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

int number = scanner.nextInt();

scanner.close();

int square = number \* number;

String numStr = Integer.toString(number);

String squareStr = Integer.toString(square);

if (squareStr.endsWith(numStr)) {

System.out.println(number + " is an Automorphic number.");

} else {

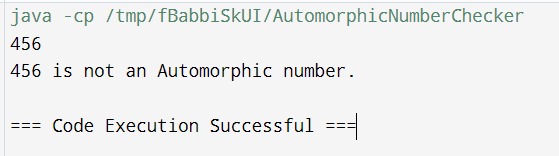
System.out.println(number + " is not an Automorphic number.");

}

}

}

**OUTPUT:**



**RESULT:**

The program successfully determines and displays whether the given number is Automorphic or not.

**Q4. Write a java program to Print numbers in given pattern**

**1**

**212**

**32123**

**4321234**

**543212345**

**AIM:**

To write a Java program that prints numbers in a specific pyramid pattern

**ALGORITHM:**

1. **Start**
2. Set the value of n (number of rows).
3. Loop i from 1 to n:
4. Loop j from n to i (print spaces).
5. Loop j from i down to 1 (print decreasing numbers).
6. Loop j from 2 to i (print increasing numbers).
7. Move to the next line after each row is printed.
8. **End**

**CODE:**

public class NumberPattern {

public static void main(String[] args) {

int n = 5;

for (int i = 1; i <= n; i++) {

for (int j = 1; j <= n - i; j++) {

System.out.print(" ");

}

for (int j = i; j > 0; j--) {

System.out.print(j);

}

for (int j = 2; j <= i; j++) {

System.out.print(j);

}

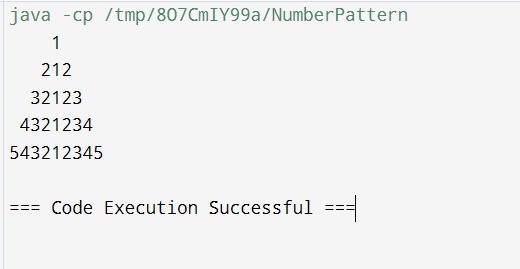
System.out.println();

}

}

}

**OUTPUT:**



**RESULT:**

The program successfully prints the numbers in the desired pyramid pattern.

**Q5.** **Create a custom wrapper class named MyInteger to encapsulate an int value. Implement the following methods:**

* **Constructor: A constructor that takes an int value as a parameter and initializes the encapsulated value.**
* **getValue(): Returns the encapsulated int value.**
* **isEven(): Returns true if the encapsulated value is even, false otherwise.**
* **isOdd(): Returns true if the encapsulated value is odd, false otherwise.**
* **isPrime(): Returns true if the encapsulated value is a prime number, false otherwise.**

**parseMyInteger(String s): Converts a String to a MyInteger object**.

**AIM:**

To create a custom wrapper class MyInteger in Java that encapsulates an integer value and provides methods to check if the value is even, odd, or prime, and to parse a string into a MyInteger object.

**ALGORITHM:**

1. **Start**
2. Create a class MyInteger with a private int attribute value.
3. Implement a constructor that initializes value with the provided parameter.
4. Define methods:
5. getValue(): Return the encapsulated value.
6. isEven(): Return true if value is even, else false.
7. isOdd(): Return true if value is odd, else false.
8. isPrime(): Check divisibility from 2 to the square root of value. If divisible, return false. Else, return true.
9. parseMyInteger(String s): Convert the string s to an integer and return a new MyInteger object with this value.
10. In the main method:
11. Create an instance of MyInteger and demonstrate the functionality of each method.
12. **End**

**CODE:**

public class MyInteger {

private int value;

public MyInteger(int value) {

this.value = value;

}

public int getValue() {

return value;

}

public boolean isEven() {

return value % 2 == 0;

}

public boolean isOdd() {

return value % 2 != 0;

}

public boolean isPrime() {

if (value <= 1) return false;

if (value <= 3) return true;

if (value % 2 == 0 || value % 3 == 0) return false;

for (int i = 5; i \* i <= value; i += 6) {

if (value % i == 0 || value % (i + 2) == 0) return false;

}

return true;

}

public static MyInteger parseMyInteger(String s) {

return new MyInteger(Integer.parseInt(s));

}

public static void main(String[] args) {

MyInteger num = MyInteger.parseMyInteger("29");

System.out.println("Value: " + num.getValue());

System.out.println("Is Even: " + num.isEven());

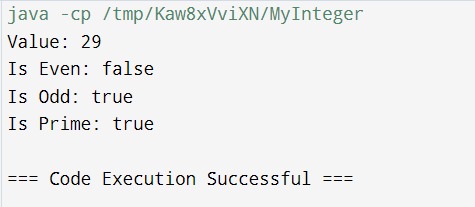
System.out.println("Is Odd: " + num.isOdd());

System.out.println("Is Prime: " + num.isPrime());

}

}

OUTPUT:



**RESULT:**

The custom wrapper class MyInteger is successfully created with all required functionalities.

**Q6. Write a Java program that reads an integer from the user using a Scanner. Convert the integer to a String using the Integer.toString() method and print the length of the String.**

**AIM:**

To write a Java program that reads an integer from the user, converts it to a string, and prints the length of the string.

**ALGORITHM:**

1. **Start**
2. Use the Scanner class to read an integer input from the user.
3. Convert the integer to a string using Integer.toString().
4. Calculate and print the length of the string.
5. **End**

**CODE:**

import java.util.Scanner;

public class IntegerToStringLength {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

int number = scanner.nextInt();

scanner.close();

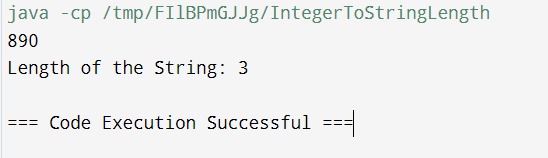
String numberStr = Integer.toString(number);

System.out.println("Length of the String: " + numberStr.length());

}

}

OUTPUT:



**RESULT:**

The program successfully reads an integer, converts it to a string, and correctly displays the length of the string.