DELHI TECHNICAL CAMPUS
GREATER NOIDA
*Affiliated to GGSIPU and approved by AICTE & COA*

# PRACTICAL FILE

**SESSION: 2024-25**

# OBJECT ORIENTED PROGRAMMING (AIML 252)
# II Year, IV Sem

**Submitted to:**
Name: Mr. Aman Kumar
Designation: Asst. Professor

**Submitted by:**
Name:Kush Gupta
Enrollment No.:03718011623
Sec:AIML

**EXPERIMENT – 1**

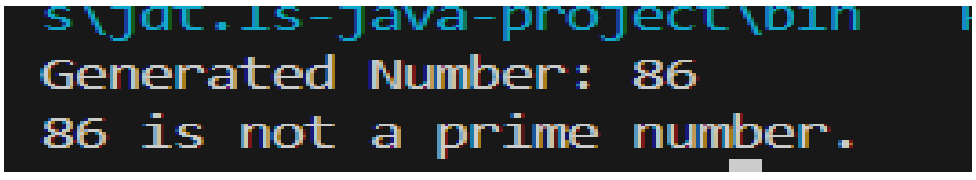
**(a)**

**AIM: Generate a random number up to 100 and print whether it is prime or**

**not. CODE:**

```
import java.util.Random;
public class PrimeCheck {
    public static void main(String[] args) {
     // Generate a random number up to 100
     Random random = new Random();
     int number = random.nextInt(101); // Random number between 0 and 100

     System.out.println("Generated Number: " + number);

     // Check if the number is prime if
     (isPrime(number)) {
           System.out.println(number + " is a prime number.");
     } else {
           System.out.println(number + " is not a prime number.");
       }
    }

    // Function to check if a number is prime
    public static boolean isPrime(int num) {
     if (num <= 1) {
           return false; // 0 and 1 are not prime
       }
       for (int i = 2; i <= Math.sqrt(num); i++)
           { if (num % i == 0) {
               return false; // Divisible by a number other than 1 and itself
           }
       }
     return true; // Prime number
    }
}
```

**OUTPUT:**



**(b)**

**AIM: Design a program to generate first 10 terms of Fibonacci series. CODE:**

```
public class FibonacciSeries {
    public static void main(String[] args) {
     int n = 10; // Number of terms to generate int
     firstTerm = 0;
     int secondTerm = 1;

     System.out.println("First " + n + " terms of the Fibonacci series:");

     // Print the first two terms System.out.print(firstTerm +
     ", " + secondTerm);

     // Generate the remaining terms for
     (int i = 3; i <= n; i++) {
          int nextTerm = firstTerm + secondTerm;
          System.out.print(", " + nextTerm);

          // Update terms for the next iteration
          firstTerm = secondTerm; secondTerm =
          nextTerm;
       }
    }
}
```

**OUTPUT:**

```
First 10 terms of the Fibonacci series:
0, 1, 1, 2, 3, 5, 8, 13, 21, 34
```

**(a)**

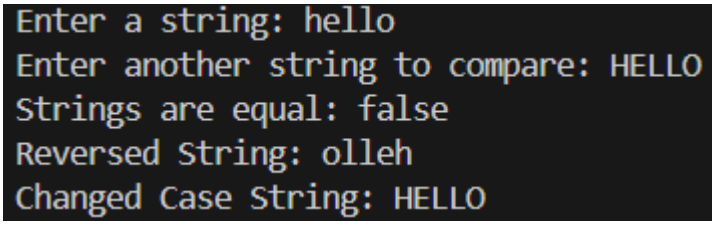
**AIM: Design a class that performs String operations (Equal, Reverse the string, change case).**

**CODE:**

```
import java.util.Scanner;
class StringOperations {
   private String str;
   public StringOperations(String str)
      { this.str = str;
   }

   public boolean isEqual(String anotherStr)
      { return str.equals(anotherStr);
   }

   public String reverseString() {
      return new StringBuilder(str).reverse().toString();
   }

   public String changeCase() {
      StringBuilder changedStr = new StringBuilder();
      for (char ch : str.toCharArray()) {
         if (Character.isUpperCase(ch)) {
            changedStr.append(Character.toLowerCase(ch));
         } else if (Character.isLowerCase(ch)) {
            changedStr.append(Character.toUpperCase(ch));
         } else {
            changedStr.append(ch);
         }
      }
      return changedStr.toString();
   }
   public static void main(String[] args)
      { Scanner scanner = new
      Scanner(System.in);

      System.out.print("Enter a string: ");
```

```
        String inputString = scanner.nextLine();

        StringOperations strOps = new StringOperations(inputString);

        System.out.print("Enter another string to compare: ");
        String compareString = scanner.nextLine();

        System.out.println("Strings are equal: " + strOps.isEqual(compareString));
        System.out.println("Reversed String: " + strOps.reverseString());
        System.out.println("Changed Case String: " + strOps.changeCase());

        scanner.close();
    }
}
```

**OUTPUT:**



**(b)**

**AIM: Find the average and sum of array of N numbers entered by user.**

**CODE:**

```
import java.util.Scanner;

public class ArraySumAverage {
   public static void main(String[] args)
      { Scanner scanner = new
      Scanner(System.in);

      System.out.print("Enter the number of elements: ");
      int n = scanner.nextInt();

      int[] numbers = new int[n];
      int sum = 0;

      System.out.println("Enter " + n + " numbers:");
      for (int i = 0; i < n; i++) {
         numbers[i] = scanner.nextInt();
         sum += numbers[i];
      }

      double average = (double) sum / n;

      System.out.println("Sum: " + sum);
      System.out.println("Average: " + average);

      scanner.close();
   }
}
```

**OUTPUT:**

```
Enter the number of elements: 4
Enter 4 numbers:
12
13
14
15
Sum: 54
Average: 13.5
```

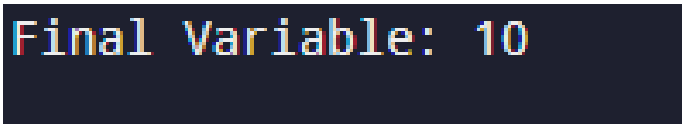
# EXPERIMENT – 3 (A)

**AIM: Demonstrate the use of final keyword with data member, function and class.**

**CODE:**

**1)**
```
final class FinalClass {
    final int finalVariable = 10;

    final void display() {
        System.out.println("Final Variable: " + finalVariable);
    }
}

public class Main {
    public static void main(String[] args)
        { FinalClass obj = new FinalClass();
        obj.display();
    }
}
```

**OUTPUT:**




**CODE:**

**2)**
```
class BaseClass {
    final int finalVariable = 10;
    void display() {
        System.out.println("Final Variable in BaseClass: " + finalVariable);
```

```
    }
}
class SubClass extends BaseClass
    { void modifyFinalVariable() {
        finalVariable = 20;     }
}
public class Main {
    public static void main(String[] args)
        { SubClass subObj = new SubClass();
        subObj.display();
        subObj.modifyFinalVariable();
    }
}
```

**OUTPUT:**

```
ERROR!
/tmp/vlp7aYO489/Main.java:15: error: cannot assign a value to final variable finalVariable
        finalVariable = 20; // Attempting to modify a final variable
        ^
1 error
```

**(B)**

**AIM: Design a program to demonstrate multi-threading using Thread Class.**

**CODE:**

```
class MyThread extends Thread {
    private String threadName;

    public MyThread(String name)
        { this.threadName = name;
    }
  @Override
    public void run() {
        for (int i = 0; i < 5; i++)
            { System.out.println(threadName + " - Count: " + i);
            try {
                Thread.sleep(500);
            } catch (InterruptedException e)
                { System.out.println(threadName + "
                interrupted.");
            }
        }
        System.out.println(threadName + " has finished execution.");
    }
}
public class MultiThreadingExample
    { public static void main(String[] args) {
        MyThread thread1 = new MyThread("Thread 1");
        MyThread thread2 = new MyThread("Thread 2");
```

```
        thread1.start()
        ;
        thread2.start()
        ; try {
            thread1.join();
            thread2.join();
        } catch (InterruptedException e)
            { System.out.println("Main thread interrupted.");
        }
        System.out.println("Main thread has finished execution.");
    }
}
```

**OUTPUT:**

```
Thread 2 - Count: 0
Thread 1 - Count: 0
Thread 2 - Count: 1
Thread 1 - Count: 1
Thread 2 - Count: 2
Thread 1 - Count: 2
Thread 2 - Count: 3
Thread 1 - Count: 3
Thread 2 - Count: 4
Thread 1 - Count: 4
Thread 2 has finished execution.
Thread 1 has finished execution.
```

## EXPERIMENT – 4 (A)

## AIM: Demonstrate the use of key word try ,catch finally ,throw and throws

## CODE:

```
import java.util.Scanner;

public class ExceptionDemo {

    // Method to demonstrate the use of throw and throws
    public static void checkAge(int age) throws Exception {
        if (age < 18) {
            // Throwing an exception explicitly
            throw new Exception("Age must be 18 or older.");
        } else {
            System.out.println("Age is valid.");
        }
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        try {
            System.out.print("Enter your age: ");
            int age = scanner.nextInt();

            // Calling the method that might throw an exception
            checkAge(age);

            // Simulate another operation
            System.out.println("Proceeding with the next steps...");

        } catch (Exception e) {
            // Handling the exception thrown in the try block
            System.out.println("Caught an exception: " + e.getMessage());
        } finally {
            // This block is always executed, regardless of an exception
            System.out.println("Finally block: Cleaning up resources...");
            scanner.close();  // Closing scanner
        }

        System.out.println("Program continues after try-catch-finally block.");
    }
     }
```

**OUTPUT:**

```
Output
Enter your age: 98
Age is valid.
Proceeding with the next steps...
Finally block: Cleaning up resources...
Program continues after try-catch-finally block.

=== Code Execution Successful ===
```

```
Output
Enter your age: lp
Caught an exception: null
Finally block: Cleaning up resources...
Program continues after try-catch-finally block.

=== Code Execution Successful ===
```

## EXPERIMENT – 4 (B)

## AIM: Design a program to demonstrate and create the game of ‘tic tac toe’

## CODE:

```
import java.util.Scanner;

public class TicTacToe {
    static char[][] board = new char[3][3]; // Game board
    static char currentPlayer = 'X'; // Start with 'X'

    // Initialize the board with empty spaces
    public static void initializeBoard() {
        for (int i = 0; i < 3; i++) {
            for (int j = 0; j < 3; j++) {
                board[i][j] = ' ';
            }
        }
    }

    // Display the board
    public static void displayBoard() {
        for (int i = 0; i < 3; i++) {
            for (int j = 0; j < 3; j++) {
                System.out.print(board[i][j]);
                if (j < 2) System.out.print("|");
            }
            System.out.println();
            if (i < 2) System.out.println("-----");
        }
    }

    // Check for a winner
    public static boolean checkWin() {
        for (int i = 0; i < 3; i++) {
            // Check rows and columns
            if ((board[i][0] == currentPlayer && board[i][1] == currentPlayer && board[i][2] == currentPlayer)
||
                (board[0][i] == currentPlayer && board[1][i] == currentPlayer && board[2][i] == currentPlayer))
{
                return true;
            }
        }
        // Check diagonals
        if ((board[0][0] == currentPlayer && board[1][1] == currentPlayer && board[2][2] == currentPlayer) ||
            (board[0][2] == currentPlayer && board[1][1] == currentPlayer && board[2][0] == currentPlayer))
```

```
{
            return true;
        }
        return false;
    }

    // Check if the board is full (draw)
    public static boolean checkDraw() {
        for (int i = 0; i < 3; i++) {
            for (int j = 0; j < 3; j++) {
                if (board[i][j] == ' ') return false;
            }
        }
        return true;
    }

    // Main method to start the game
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        initializeBoard();
        displayBoard();

        while (true) {
            System.out.println("Player " + currentPlayer + "'s turn:");
            System.out.print("Enter row (0-2): ");
            int row = scanner.nextInt();
            System.out.print("Enter column (0-2): ");
            int col = scanner.nextInt();

            // Validate move
            if (row >= 0 && row < 3 && col >= 0 && col < 3 && board[row][col] == ' ') {
                board[row][col] = currentPlayer;
                displayBoard();

                if (checkWin()) {
                    System.out.println("Player " + currentPlayer + " wins!");
                    break;
                }

                if (checkDraw()) {
                    System.out.println("It's a draw!");
                    break;
                }

                // Switch player
                currentPlayer = (currentPlayer == 'X') ? 'O' : 'X';
            } else {
                System.out.println("Invalid move, try again.");
```

```
        }
      }

      scanner.close();
   }
}
```

**OUTPUT:**

Output

```
 | |
-----
 | |
-----
 | |
Player X's turn:
Enter row (0-2): 1
Enter column (0-2): 2
 | |
-----
 | |X
-----
 | |
Player O's turn:
Enter row (0-2): 1
Enter column (0-2): 1
 | |
-----
 |O|X
-----
 | |
Player X's turn:
Enter row (0-2): 2
Enter column (0-2): 3
Invalid move, try again.
Player X's turn:
Enter row (0-2): 1
Enter column (0-2): 3
Invalid move, try again.
Player X's turn:
```