OOPS Lab File 2025

School of Technology, Woxsen University

March 23, 2025

Contents

1 Experiment 1: Java Basic Programming

1.1 1.2: Evaluate Specified Expressions

```java
public class ExpressionEvaluator {
    public static void main(String[] args) {
        System.out.println("Expression 1: (101 + 0) / 3");
        double result1 = (101 + 0) / 3.0;
        System.out.println("Result 1: " + result1);

        System.out.println("\nExpression 2: (3.0e-6 * 10000000.1)");
        double result2 = 3.0e-6 * 10000000.1;
        System.out.println("Result 2: " + result2);

        System.out.println("\nExpression 3: (true && true)");
        boolean result3 = true && true;
        System.out.println("Result 3: " + result3);

        System.out.println("\nExpression 4: (false && true)");
        boolean result4 = false && true;
        System.out.println("Result 4: " + result4);

        System.out.println("\nExpression 5: (false && false) || (true && true)");
        boolean result5 = (false && false) || (true && true);
        System.out.println("Result 5: " + result5);

        System.out.println("\nExpression 6: (false || false) && (true && true)");
        boolean result6 = (false || false) && (true && true);
        System.out.println("Result 6: " + result6);
    }
}
```

Listing 1: ExpressionEvaluator.java

```
Expression 1: (101 + 0) / 3
Result 1: 33.666666666666664

Expression 2: (3.0e-6 * 10000000.1)
Result 2: 30.0000003

Expression 3: (true && true)
Result 3: true

Expression 4: (false && true)
Result 4: false

Expression 5: (false && false) || (true && true)
Result 5: true

Expression 6: (false || false) && (true && true)
Result 6: false
```

1.2 1.3: Fahrenheit to Celsius Converter

```java
import java.util.Scanner;

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

        System.out.print("Input a degree in Fahrenheit: ");
        double fahrenheit = sc.nextDouble();

        double celsius = (fahrenheit - 32) * 5.0 / 9.0;
        System.out.println(fahrenheit + " degree Fahrenheit is equal to " + celsius + "
            in Celsius");

        sc.close();
    }
}
```

Listing 2: TempConverter.java

```
Input a degree in Fahrenheit: 212
212.0 degree Fahrenheit is equal to 100.0 in Celsius
```

2 Experiment 2: Conditional Statement

2.1 2.1: Check Four Integers Equal

```java
import java.util.Scanner;

public class CheckEqual {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter four integers:");
        int a = sc.nextInt();
        int b = sc.nextInt();
        int c = sc.nextInt();
        int d = sc.nextInt();

        if (a == b && b == c && c == d) {
            System.out.println("equal");
        } else {
            System.out.println("not equal");
        }

        sc.close();
    }
}
```

Listing 3: CheckEqual.java

```
Enter four integers:
5 5 5 5
equal
```

2.2 2.2: Check Range of Two Doubles

```java
import java.util.Scanner;

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

        System.out.println("Enter two double values:");
        double num1 = sc.nextDouble();
        double num2 = sc.nextDouble();

        boolean result = (num1 > 0 && num1 < 1) && (num2 > 0 && num2 < 1);
        System.out.println("Both numbers are strictly between 0 and 1: " + result);

        sc.close();
    }
}
```

Listing 4: CheckRange.java

```
Enter two double values:
0.5 0.7
Both numbers are strictly between 0 and 1: true
```

3 Experiment 3: Introduction to Arrays in Java

3.1 3.1: Print 2D Boolean Array

```java
public class BooleanArrayPrinter {
    public static void main(String[] args) {
        boolean[][] array = {{true, false, true}, {false, true, false}};

        for (int i = 0; i < array.length; i++) {
            for (int j = 0; j < array[i].length; j++) {
                System.out.print(array[i][j] ? "t " : "f ");
            }
            System.out.println();
        }
    }
}
```

Listing 5: BooleanArrayPrinter.java

```
t f t
f t f
```

3.2 3.2: Transpose 2D Array

```java
public class ArrayTranspose {
    public static void main(String[] args) {
        int[][] original = {{10, 20, 30}, {40, 50, 60}};
        int[][] transposed = new int[3][2];

        for (int i = 0; i < original.length; i++) {
            for (int j = 0; j < original[i].length; j++) {
                transposed[j][i] = original[i][j];
            }
        }

        for (int i = 0; i < transposed.length; i++) {
            for (int j = 0; j < transposed[i].length; j++) {
                System.out.print(transposed[i][j] + " ");
            }
            System.out.println();
        }
    }
}
```

Listing 6: ArrayTranspose.java

```
10 40
20 50
30 60
```

4 Experiment 4: 2-Dimensional Array in Java

4.1 4.1: Prime Number Condition Array

```java
public class PrimeArray {
    public static boolean isPrime(int n) {
        if (n < 2) return false;
        for (int i = 2; i <= Math.sqrt(n); i++) {
            if (n % i == 0) return false;
        }
        return true;
    }

    public static void main(String[] args) {
        int m = 5;
        boolean[][] A = new boolean[m][m];

        for (int i = 0; i < m; i++) {
            for (int j = 0; j < m; j++) {
                A[i][j] = !(isPrime(i) && isPrime(j));
                System.out.print((A[i][j] ? "t " : "f ") + " ");
            }
            System.out.println();
        }
    }
}
```

Listing 7: PrimeArray.java

```
t  t  t  t  t
t  t  t  t  t
t  t  f  f  t
t  t  f  f  t
t  t  t  t  t
```

4.2 4.2: K Largest Elements

```java
import java.util.Arrays;
import java.util.Collections;

public class KLargest {
    public static void main(String[] args) {
        Integer[] array = {4, 2, 9, 7, 5, 6, 1, 3};
        int k = 3;

        Arrays.sort(array, Collections.reverseOrder());

        System.out.println("The " + k + " largest elements are:");
        for (int i = 0; i < k; i++) {
            System.out.print(array[i] + " ");
        }
    }
}
```

Listing 8: KLargest.java

```
The 3 largest elements are:
9 7 6
```

5 Experiment 5: Introduction of Class in Java

5.1 5.1: Vehicle and Car Classes

```java
class Vehicle {
    public void drive() {
        System.out.println("Driving a vehicle");
    }
}

class Car extends Vehicle {
    @Override
    public void drive() {
        System.out.println("Repairing a car");
    }
}

public class VehicleTest {
    public static void main(String[] args) {
        Car car = new Car();
        car.drive();
    }
}
```

Listing 9: VehicleTest.java

```
Repairing a car
```

5.2 5.2: Shape and Rectangle Classes

```java
class Shape {
    public double getArea() {
        return 0.0;
    }
}

class Rectangle extends Shape {
    private double length;
    private double width;

    public Rectangle(double length, double width) {
        this.length = length;
        this.width = width;
    }

    @Override
    public double getArea() {
        return length * width;
    }
}

public class ShapeTest {
    public static void main(String[] args) {
        Rectangle rect = new Rectangle(5, 3);
        System.out.println("Rectangle Area: " + rect.getArea());
    }
}
```

Listing 10: ShapeTest.java

```
Rectangle Area: 15.0
```

5.3 5.3: Employee and HRManager Classes

```java
class Employee {
    public void work() {
        System.out.println("Employee is working");
    }
    public double getSalary() {
        return 50000.0;
    }
}

class HRManager extends Employee {
    @Override
    public void work() {
        System.out.println("HR Manager is managing employees");
    }

    public void addEmployee() {
        System.out.println("Adding a new employee");
    }
}

public class EmployeeTest {
    public static void main(String[] args) {
        HRManager hr = new HRManager();
        hr.work();
        System.out.println("Salary: " + hr.getSalary());
        hr.addEmployee();
    }
}
```

Listing 11: EmployeeTest.java

```
HR Manager is managing employees
Salary: 50000.0
Adding a new employee
```

6 Experiment 6: Java Class with Real-life Applications

6.1 6.1: BankAccount and SavingsAccount

```java
class BankAccount {
    protected double balance;

    public BankAccount() {
        this.balance = 0;
    }

    public void deposit(double amount) {
        balance += amount;
        System.out.println("Deposited: " + amount + ", New Balance: " + balance);
    }

    public void withdraw(double amount) {
        if (balance >= amount) {
            balance -= amount;
            System.out.println("Withdrawn: " + amount + ", New Balance: " + balance);
        } else {
            System.out.println("Insufficient funds");
        }
    }
}

class SavingsAccount extends BankAccount {
    @Override
    public void withdraw(double amount) {
        if (balance - amount < 100) {
            System.out.println("Cannot withdraw: Minimum balance of 100 required");
        } else {
            super.withdraw(amount);
        }
    }
}

public class BankTest {
    public static void main(String[] args) {
        SavingsAccount account = new SavingsAccount();
        account.deposit(500);
        account.withdraw(300);
        account.withdraw(150);
    }
}
```

Listing 12: BankTest.java

```
Deposited: 500.0, New Balance: 500.0
Withdrawn: 300.0, New Balance: 200.0
Cannot withdraw: Minimum balance of 100 required
```

6.2 6.2: K Largest Elements

```java
import java.util.Arrays;
import java.util.Collections;

public class KLargest {
    public static void main(String[] args) {
        Integer[] array = {4, 2, 9, 7, 5, 6, 1, 3};
        int k = 3;

        Arrays.sort(array, Collections.reverseOrder());

        System.out.println("The " + k + " largest elements are:");
        for (int i = 0; i < k; i++) {
            System.out.print(array[i] + " ");
        }
    }
}
```

Listing 13: KLargest.java

```
The 3 largest elements are:
9 7 6
```

6.3 6.3: Vehicle Class Hierarchy

```java
class Vehicle {
    protected String make, model;
    protected int year;
    protected String fuelType;

    public Vehicle(String make, String model, int year, String fuelType) {
        this.make = make;
        this.model = model;
        this.year = year;
        this.fuelType = fuelType;
    }

    public double calcFuelEfficiency() { return 0.0; }
    public double calcDistanceTravelled(double time, double speed) { return time *
        speed; }
    public double getMaxSpeed() { return 0.0; }
}

class Truck extends Vehicle {
    public Truck(String make, String model, int year, String fuelType) {
        super(make, model, year, fuelType);
    }
    @Override
    public double calcFuelEfficiency() { return 10.5; }
    @Override
    public double getMaxSpeed() { return 80.0; }
}

public class VehicleHierarchy {
    public static void main(String[] args) {
        Truck truck = new Truck("Ford", "F150", 2020, "Diesel");
        System.out.println("Truck Fuel Efficiency: " + truck.calcFuelEfficiency());
        System.out.println("Distance Travelled: " + truck.calcDistanceTravelled(2, 60));
        System.out.println("Max Speed: " + truck.getMaxSpeed());
    }
}
```

Listing 14: VehicleHierarchy.java

```
Truck Fuel Efficiency: 10.5
Distance Travelled: 120.0
Max Speed: 80.0
```