Noel Mathew Thomas
2348538

LAB 5

Question 1 (4 Marks)

Create an interface named 'BankInterface' containing abstract methods 'getBalance' and 'getInterestRate'. Deposit amounts of 10000, 150000, and 200000 into banks A, B, and C, respectively. 'BankA', 'BankB', and 'BankC' implement the 'BankInterface' interface, each defining methods 'getBalance' and 'getInterestRate'. Invoke these methods by instantiating objects of each class.

BankA provides an interest rate of 7% based on the balance.
BankB offers an interest rate of 7.4% based on the balance.
BankC provides an interest rate of 7.9% based on the balance.

Display the balance and interest rate of each bank separately.

CODE-

```
package LAB5;

interface BankInterface {

  void getBalance();

  double getInterestRate();

}

// Bank A class that implements bank interface

class BankA implements BankInterface {

  double balance;

  // Bank A constructor that receives initial balance

  BankA(double balance) {
```

```
        this.balance = balance;
    }
    // getBalance from interface ,calculating balance based on interest
    public void getBalance() {
        System.out.println("Bank A interest rate : 7%");
        double final_amount = getInterestRate() + balance;
        System.out.println("Total amount:"+ final_amount);
    }
    // getinterestrate method from interface
    public double getInterestRate() {
        double interest = balance * 0.07;
        return interest;
    }


}
// Bank B class that implements bank interface
class BankB implements BankInterface {
    double balance;
    BankB(double balance) {
```

```
        this.balance = balance;
    }
    public void getBalance() {
        System.out.println("Bank B interest rate : 7.4%");
        double final_amount = getInterestRate() + balance;
        System.out.println("Total amount:"+ final_amount);
    }
    public double getInterestRate() {
        double interest = balance * 0.074;
        return interest;
    }
}
// Bank C class that implements bank interface
class BankC implements BankInterface {
    double balance;
    BankC(double balance) {
        this.balance = balance;
    }
    public void getBalance() {
```

```
        System.out.println("Bank C interest rate : 7.9%");
        double final_amount = getInterestRate() + balance;
        System.out.println("Total amount:"+ final_amount);
    }
    public double getInterestRate() {
        double interest = balance * 0.079;
        return interest;
    }
}
public class Bank {
    public static void main(String[] args) {
        BankA bankA = new BankA(10000);
        BankB bankB = new BankB(150000);
        BankC bankC = new BankC(200000);
        bankA.getBalance();
        bankB.getBalance();
        bankC.getBalance();   }}
```

OUTPUT-

Bank A interest rate : 7%

Total amount:10700.0

Bank B interest rate : 7.4%

Total amount:161100.0

Bank C interest rate : 7.9%

Total amount:215800.0

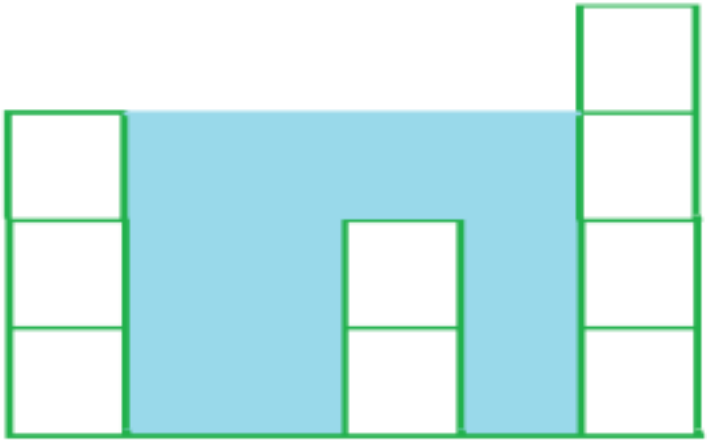
Question 2 (5 Marks)

Imagine a cityscape with an array of blocks, each represented by a non-negative integer indicating its height. During the rainy season, the city faces a unique water conservation challenge. If the width of each block is considered to be 1 unit, we aim to determine the volume of water that can be conserved between these blocks.

As an urban planner, your task is to develop an innovative water conservation system. This involves creating an interface named **WaterConservationSystem** with a method **calculateTrappedWater(int[] blockHeights)**. This method should efficiently calculate and return the total volume of water that can be conserved between the city blocks.

To implement this system, design an abstract class called **RainySeasonConservation** that implements the **WaterConservationSystem** interface. This abstract class will serve as a foundation for specific implementations based on various block configurations.

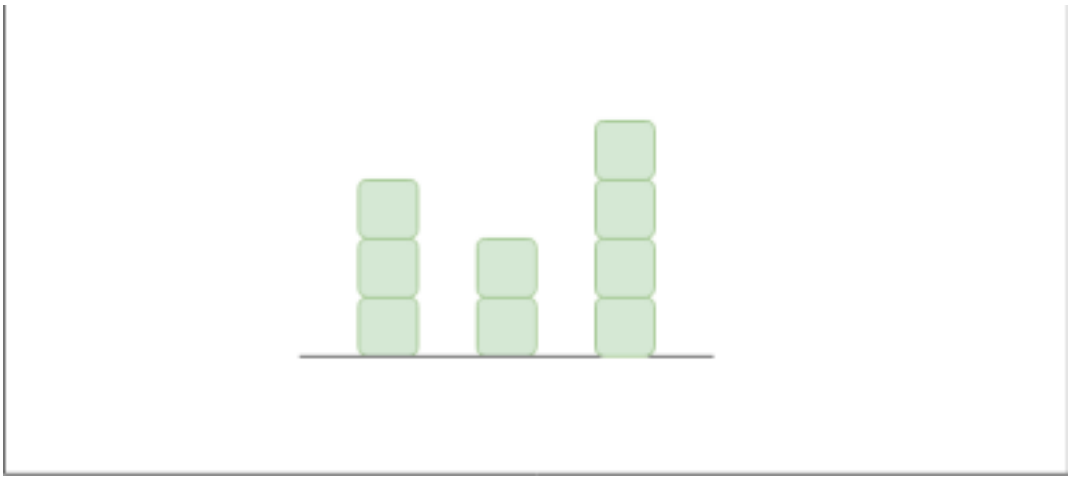
Subsequently, create a class named **CityBlockConservation** that extends **RainySeasonConservation**. Implement the **calculateTrappedWater(int[] blockHeights)** method to address the given block heights, represented by the array **blockHeights[]**.
Test Case 1:



Bars for input {3, 0, 0, 2, 0, 4}
Total trapped water = 3 + 3 + 1 + 3 = 10

Test Case 2:

Input: arr[] = {3, 0, 2, 0, 4}
Output: 7
Explanation: Structure is like below.
We can trap “3 units” of water between 3 and 2,
“1 unit” on top of bar 2 and “3 units” between 2 and 4.



```
package LAB5;
```

```
interface WaterConservationSystem {

    int calculateTrappedWater(int[] blockHeights);

}

abstract class RainySeasonConservation implements
WaterConservationSystem {

}

class CityBlockConservation extends RainySeasonConservation {

    public int calculateTrappedWater(int[] blockHeights) {

        if (blockHeights == null || blockHeights.length <= 2) {

            return 0;

        }

        int n = blockHeights.length;

        int[] l_block_Height = new int[n];

        int[] r_block_Height = new int[n];

        l_block_Height[0] = blockHeights[0];

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

            l_block_Height[i] = Math.max(l_block_Height[i - 1],
blockHeights[i]);

        }

        r_block_Height[n - 1] = blockHeights[n - 1];
```

```
        for (int i = n - 2; i >= 0; i--) {

            r_block_Height[i] = Math.max(r_block_Height[i + 1],
blockHeights[i]);

        }

        int trappedWater = 0;

        for (int i = 0; i < n; i++) {

            int minHeight = Math.min(l_block_Height[i],
r_block_Height[i]);

            trappedWater += Math.max(0, minHeight -
blockHeights[i]);

        }

        return trappedWater;

    }

}

public class Main {

    public static void main(String[] args) {

        WaterConservationSystem c = new
CityBlockConservation();

        int[] blockHeights = {3,0,0,2,0,4};

        int trappedWater =
c.calculateTrappedWater(blockHeights);
```

```
        System.out.println("Trapped Water: " + trappedWater);

    }

}
```

Output-

Trapped Water: 10