Java(25/7/2024)

1. 1. Bank is a class that provides method to get the rate of interest. But, rate of interest may differ according to banks. For example, SBI, ICICI and AXIS banks are providing 8.4%, 7.3% and 9.7% rate of interest. Write a Java program for above scenario.

Sample Input SBI, 8.4

Sample Output

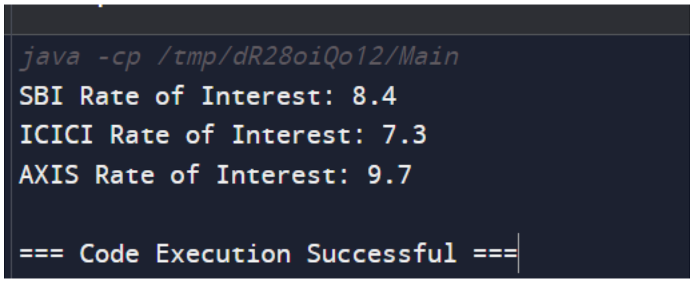
Test case

1. SBI, 8.3
2. ICICI, 7.3
3. AXIS, 9.7
4. SBI, 8.6
5. AXIX, 7.6

Code:

```
class Bank {
   double getRateOfInterest() {
      return 0.0;
   }
}
class SBI extends Bank {
   double getRateOfInterest() {
      return 8.4;
   }
}
class ICICI extends Bank {
   double getRateOfInterest() {
      return 7.3;
   }
}
class AXIS extends Bank {
```

```
            double getRateOfInterest() {
              return 9.7;
            }
          }
public class Main {
            public static void main(String[] args) {
              Bank b1 = new SBI();
              Bank b2 = new ICICI();
              Bank b3 = new AXIS();
System.out.println("SBI Rate of Interest: " + b1.getRateOfInterest());
              System.out.println("ICICI Rate of Interest: " + b2.getRateOfInterest());
              System.out.println("AXIS Rate of Interest: " + b3.getRateOfInterest());
            }
          }
```



```
java -cp /tmp/dR28oiQo12/Main
SBI Rate of Interest: 8.4
ICICI Rate of Interest: 7.3
AXIS Rate of Interest: 9.7

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

2. Bring out the situation in which member names of a subclass hide members by the same name in the super class. How it can be resolved? Write Suitable code in Java and

Implement above scenario with the Parametrized Constructor (accept int type parameter) of the Super Class can be called from Sub Class Using super () and display the input values provided.

Sample Input : 100, 200

Sample Output : 100, 200

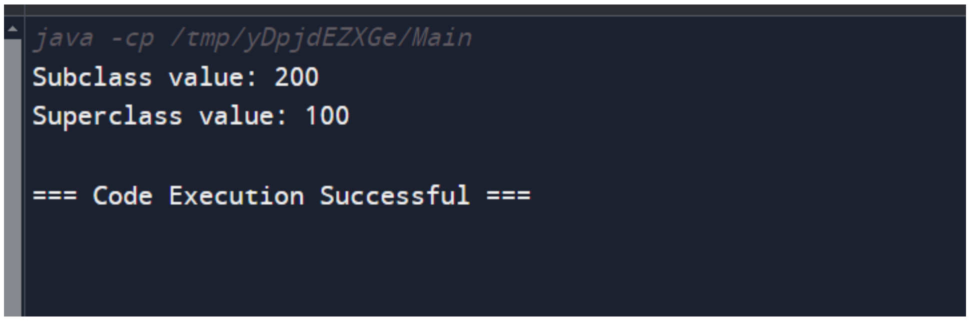
Test Cases

1. 10, 20

2. -20, -30
3. 0, 0
4. EIGHT FIVE
5. 10.57, 12.58

Code:

```
class SuperClass {
    int value;
SuperClass(int value) {
        this.value = value;
    }
}
class SubClass extends SuperClass {
    int value;
SubClass(int subValue, int superValue) {
        super(superValue);
        this.value = subValue;
    }
 void displayValues() {
        System.out.println("Subclass value: " + this.value);
        System.out.println("Superclass value: " + super.value);
    }
}
public class Main {
    public static void main(String[] args) {
        SubClass obj = new SubClass(200, 100);
        obj.displayValues();
    }
}
```



```
java -cp /tmp/yDpjdEZXGe/Main
Subclass value: 200
Superclass value: 100

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

3. Display Multiplication table for 5 and 10 using various stages of life cycle of the thread by generating a suitable code in Java.

Sample Input 5, 10

5 X 1 = 5

5 X 2 =10

....

10 X 1 =10

10 X 2 = 20

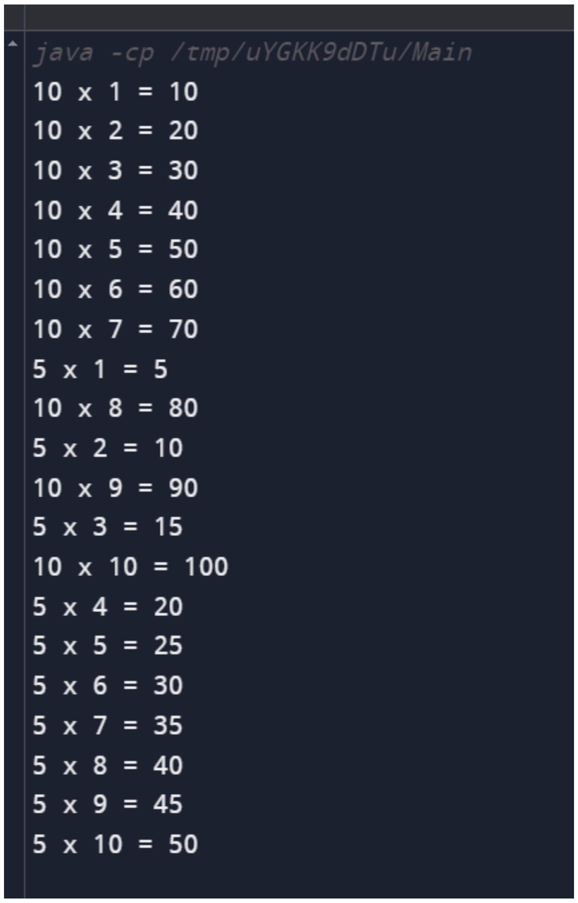
....

Test Cases:

1. 10, 20
2. -10, -30
3. 0, 0
4. SIX, SIX
5. 9.8, 9.6

Code:

```
class MultiplicationTable extends Thread {
    private int number;
MultiplicationTable(int number) {
        this.number = number;
    }
public void run() {
        for (int i = 1; i <= 10; i++) {
            System.out.println(number + " x " + i + " = " + number * i);
        }
    }
}
public class Main {
    public static void main(String[] args) {
        MultiplicationTable table5 = new MultiplicationTable(5);
        MultiplicationTable table10 = new MultiplicationTable(10);
        table5.start();
        table10.start();
    }
}
```



```
java -cp /tmp/uYGKK9dDTu/Main
10 x 1 = 10
10 x 2 = 20
10 x 3 = 30
10 x 4 = 40
10 x 5 = 50
10 x 6 = 60
10 x 7 = 70
5 x 1 = 5
10 x 8 = 80
5 x 2 = 10
10 x 9 = 90
5 x 3 = 15
10 x 10 = 100
5 x 4 = 20
5 x 5 = 25
5 x 6 = 30
5 x 7 = 35
5 x 8 = 40
5 x 9 = 45
5 x 10 = 50
```

4. Using the concepts of thread with implementing Runnable interface in Java to generate Fibonacci series.

Sample Input : 5

Sample Output : 0 1 1 2 3 …..

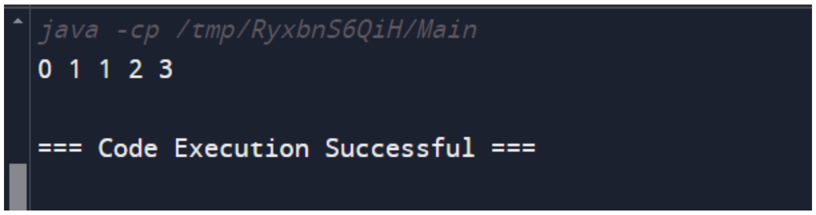
Test Cases

1. 7
2. -10
3. 0
4. EIGHT FIVE
5. 12.65

Code:

```
class FibonacciRunnable implements Runnable {
    private int count;
FibonacciRunnable(int count) {
        this.count = count;
    }
    public void run() {
        int a = 0, b = 1;
        System.out.print(a + " " + b);
        for (int i = 2; i < count; i++) {
            int next = a + b;
            System.out.print(" " + next);
            a = b;
            b = next;
        }
        System.out.println();
    }
}
```

```
public class Main {
    public static void main(String[] args) {
        int n = 5; // Example input
        Thread thread = new Thread(new FibonacciRunnable(n));
        thread.start();
    }
}
```



```
java -cp /tmp/RyxbnS6QiH/Main
0 1 1 2 3

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

5. Generate a Java code to find the sum of N numbers using array and throw ArrayIndexOutOfBoundsException when the loop variable beyond the size N.

Sample Input : 5

1 2 3 4 5

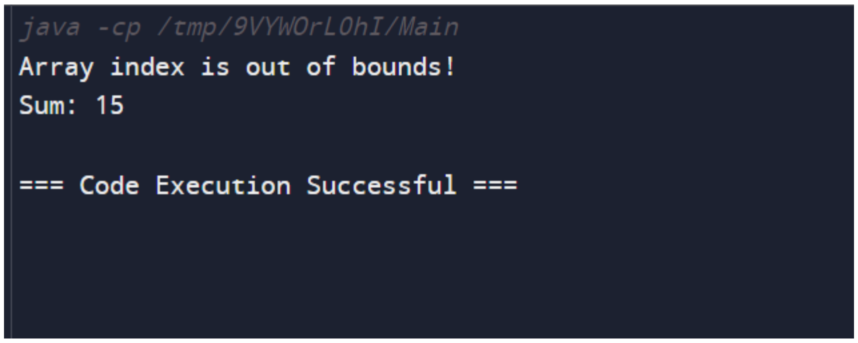
Sample Output : 15

Test Cases

1. 4, 10
2. -10
3. 0
4. EIGHT SEVEN
5. 12.68

Code:

```
public class Main {
    public static void main(String[] args) {
        int[] numbers = {1, 2, 3, 4, 5};
        int sum = 0;
        try {
            for (int i = 0; i <= numbers.length; i++) {
                sum += numbers[i];
            }
        } catch (ArrayIndexOutOfBoundsException e) {
            System.out.println("Array index is out of bounds!");
        }
        System.out.println("Sum: " + sum);
    }
}
```



```
java -cp /tmp/9VYWOrLOhI/Main
Array index is out of bounds!
Sum: 15

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