CST8132 Object Oriented Programming

Exercise 2 – Arrays, Loops, Memory Maps

Due in the lab period of week 3 (Jan 25 to Jan 29)

Overview

- In this lab exercise you will practice using loops with 1-dimensional arrays, 2-dimensional arrays, as well as drawing memory maps.
- Create a project in Eclipse using the provided starting code for classes DemoLauncher, ArrayDemo, and Goldfish.
    - Classes DemoLauncher, and Goldfish are provided fully.
    - Class ArrayDemo will need loops created to complete the indicated tasks.
- You can use enhanced for loops or normal for loops, just keep in mind that the enhanced for loop cannot be used to assign values into array elements.

DemoLauncher

```java
public class DemoLauncher {
    public static void main(String[] args) {
        ArrayDemo2 demo = new ArrayDemo2();
        demo.demonstrateTask1();
        demo.demonstrateTask2();
        demo.demonstrateTask3();
        demo.demonstrateTask4();
    }
}
```

Goldfish

```java
public class Goldfish {

    private double weight;

    public double getWeight(){
        return weight;
    }

    public void setWeight(double weight){
        this.weight = weight;
    }
}
```

```java
public class ArrayDemo {

    public void demonstrateTask1(){
        System.out.println("Demonstrate Task 1");

        int[] numbers = null; // note that numbers references nothing (null) initially
        numbers = new int[5]; // a reference to an Array object containing int values is assigned

        System.out.println("Array of primitives before assigning values");
        // TODO: loop to print the values in the array elements on one line separated by spaces

        // TODO: loop to assign values 1 to 5 into the array elements

        System.out.println("\nArray of primitives after assigning values");
        // TODO: loop to again print the values on one line separated by spaces

        // TODO: Draw a memory map for int[] numbers = new int[5]; at this point
    }

    public void demonstrateTask2(){
        System.out.println("\n\nDemonstrate Task 2");

        Goldfish[] fishes = new Goldfish[5];

        System.out.println("\nArray of references before assigning references");
        // TODO: loop to print the reference value within each array element separated by a space

        // TODO: loop to initialize each array element with a Goldfish object, each with
        // weights of 1, 2, 3, 4, 5

        System.out.println("\nArray of references after assigning references to Goldfish");
        // TODO: loop to print the each referenced objects weight separated by a space

        // TODO: Draw a memory map for Goldfish[] fishes = new Goldfish[5]; at this point

    }

    public void demonstrateTask3(){
        System.out.println("\nDemonstrate Task 3");

        int[][] numbers = new int[5][5];

        System.out.println("\n2D Array of primitives before assigning values");
        // TODO: use nested loops to display the 25 default int values in the array elements,
        // 5 per row, 5 rows separated by spaces (Tip: Put a println within the outer loop
        // below the inner loop)

        // TODO: use nested loops to initialize each element with the multiplication of
        // the loop couters + 1, e.g. element[0][0] should have a value of 1, element[4][4]
        // should have a value of 25

        System.out.println("\n2D Array of primitives after assigning values");
        // TODO: use nested loops to display the 25 default int values in the array elements,
        // 5 per row, 5 rows separated by spaces

        // TODO: Draw a memory map for int[][] numbers = new int[5][5]; at this point

    }

    // BONUS (4) for this exercise. 2 points for code, 2 points for correct memory map.
    // Note: Task 4 leads into Assignment 1, the Battlefield that uses a 2D array of references
    //       to objects instantiated from class Square.
    public void demonstrateTask4(){

        System.out.println("Demonstrate Task 4");

        Goldfish[][] fishes = new Goldfish[5][5];

        System.out.println("\n2D Array of references before assigning references");
        // TODO: use nested loops to display the 25 default reference values in the array
        // elements, 5 per row, 5 rows separated by spaces

        // TODO: use nested loops to store a reference to 25 Goldfish objects, each one with a
        // weight set to be the multiplication of the loop counters + 1, for example the Goldfish
        // object  referenced by element[0][0] would have a weight of 1, while the Goldfish object
        // referenced by element[4][4] would have a weight of 25.


        System.out.println("\n2D Array of references after assigning references to Goldfish");
        // TODO: use nested loops to display the weight of the 25 referenced Goldfish objects,
        // 5 per row, 5 rows separated by spaces.

        // Draw a memory map for Goldfish[][] fishes = new Goldfish[5][5]; at this point

    }
}
```

Grading Guide:

Task 1 completed correctly (1D Array Primitives)	1
Task 1 memory map correct	1
Task 2 completed correctly (1D Array References) Task 2 memory map correct	2 2
Task 3 completed correctly (2D Array Primitives)	2
Task 3 memory map correct	2
Task 4 completed correctly (2D Array References)	2 (Bonus)
Task 4 memory map correct	2 (Bonus)
Total	10 (14 with bonus)

Sample Program Output

```
Demonstrate Task 1
Array of primitives before assigning values
0 0 0 0 0
Array of primitives after assigning values
1 2 3 4 5

Demonstrate Task 2

Array of references before assigning references
null null null null null
Array of references after assigning references to Goldfish objects
1.0 2.0 3.0 4.0 5.0

Demonstrate Task 3

2D Array of primitives before assigning values
0 0 0 0 0
0 0 0 0 0
0 0 0 0 0
0 0 0 0 0
0 0 0 0 0

2D Array of primitives after assigning values
1 2 3 4 5
2 4 6 8 10
3 6 9 12 15
4 8 12 16 20
5 10 15 20 25

Demonstrate Task 4

2D Array of references before assigning references
null null null null null
null null null null null
null null null null null
null null null null null
null null null null null

2D Array of references after assigning references to Goldfish objects
1.00 2.00 3.00 4.00 5.00
2.00 4.00 6.00 8.00 10.00
3.00 6.00 9.00 12.00 15.00
4.00 8.00 12.00 16.00 20.00
5.00 10.00 15.00 20.00 25.00
```