Discussion-5: Working with Arrays and ArrayLists

Discussion Topic:

Identify three real-life scenarios in which an array could be used for storing information. Provide a sample code segment that illustrates how to store data in an array for one of your outlined scenarios. Be sure to reply to two of your classmates' posts, providing constructive feedback on the scenarios identified by your classmates.

My Post:

Hello class,

Before I give real-life scenarios, I would like to define what an array is in the context of Java programming. An array is a container object that holds a fixed number of values of a single type.

Characteristics of Java Arrays:

- Fixed Size: Once defined, the size of an array cannot be changed.
- Ordered: Arrays store elements in a sequential order which means that elements can be accessed by their index in constant time.
- Efficiency: Accessing any element in an array is a constant time operation. Arrays have a very low memory overhead because they store a single type of data.
- Single Type: Java arrays are typed, which means they can only store elements of the same data type as declared in the array declaration.

Arrays are different from arrayList which are lists and are part of the collection interface. An interface in Java is a reference type, similar to a class, that can contain only constants, default methods, static methods, and nested types (Tutorials Point, n. d.). In the case of the Collection interface, it includes methods like add(), remove(), get(), and size(), among others (Oracle Doc., n.d.). This allows different types of list classes, such as ArrayList, LinkedList, and others class type such as Set, to use those methods.
Note that arrays are not part of the Collection interface. In other words, they do not have methods associated with them.

Below are three real-live scenarios where an array could be used for storing data:

Scenario-1:
An array can be used to keep track of the quantities of different products in a store. For example, each element of the array represents the quantity of a specific product.

```
public class Main {
    public static void main(String[] args) {
        // Stores product quantities
        int[] quantities = new int[4];
        // Storing product quantities
        quantities[0] = 50;
        quantities[1] = 30;
```

```
        quantities[2] = 20;
        quantities[3] = 40;
        // Prints the product quantities
        for (int i = 0; i < quantities.length; i++) {
            System.out.println("Product " + (i + 1) + " Quantity: " + quantities[i]);
        }
    }
}
```

Output

```
Product 1 Quantity: 50
Product 2 Quantity: 30
Product 3 Quantity: 20
Product 4 Quantity: 40
```

Scenario-2:

An array can be used to store and modify daily temperatures.

```
public class Main {
    public static void main(String[] args) {
        // Stores daily temperatures
        int[] temperatures = {68, 70, 75, 72, 69, 71, 73};

        // Prints initial temperatures
        System.out.println("Initial daily temperatures:");
        printTemperatures(temperatures);

        // Modifies temperatures
        modifyTemperature(temperatures, 2, 78);
        modifyTemperature(temperatures, 5, 74);

        // Prints updated temperatures
        System.out.println("\nUpdated daily temperatures:");
        printTemperatures(temperatures);
    }

    // Method to print all temperatures
    public static void printTemperatures(int[] temperatures) {
        String[] days = {"Monday", "Tuesday", "Wednesday", "Thursday",
                         "Friday","Saturday", "Sunday"};
        for (int i = 0; i < temperatures.length; i++) {
            System.out.println(days[i] + ": " + temperatures[i] + "°F");
        }
    }

    // Method to modify a temperature
    public static void modifyTemperature(int[] temperatures, int dayIndex,
                                         int newTemperature) {
        if (dayIndex >= 0 && dayIndex < temperatures.length) {
            temperatures[dayIndex] = newTemperature;
```

```
        } else {
            System.out.println("Invalid day index!");
        }
    }
}
```

Ouput

```
Initial daily temperatures:
Monday: 68°F
Tuesday: 70°F
Wednesday: 75°F
Thursday: 72°F
Friday: 69°F
Saturday: 71°F
Sunday: 73°F

Updated daily temperatures:
Monday: 68°F
Tuesday: 70°F
Wednesday: 78°F
Thursday: 72°F
Friday: 69°F
Saturday: 74°F
Sunday: 73°F
```

Scenario-3:

An array can be used to store and sort students' grades in a particular class.

```
import java.util.Arrays;

public class Main{
    public static void main(String[] args) {
        // Two-dimensional array to store student IDs and their grades
        int[][] studentGrades = {
            {1, 85}, // {Student ID, Grade}
            {2, 90},
            {3, 78},
            {4, 92},
            {5, 88}
        };
        // Prints initial student grades
        System.out.println("Initial student grades:");

        printAllGrades(studentGrades);
        // Sorting grades from lowest to highest
        sortGrades(studentGrades);

        // Printing sorted student grades
        System.out.println("\nSorted student grades:");
```

```
        printAllGrades(studentGrades);
    }

    // Prints all grades
    public static void printAllGrades(int[][] studentGrades) {
        for (int i = 0; i < studentGrades.length; i++) {
            System.out.println("Student ID " + studentGrades[i][0]
                               + ": " + studentGrades[i][1]);
        }
    }
    // Sorts grades
    public static void sortGrades(int[][] studentGrades) {
        Arrays.sort(studentGrades, (a, b) -> Integer.compare(a[1], b[1]));
    }
}
```

Output

```
Initial student grades:
Student ID 1: 85
Student ID 2: 90
Student ID 3: 78
Student ID 4: 92
Student ID 5: 88

Sorted student grades:
Student ID 3: 78
Student ID 1: 85
Student ID 5: 88
Student ID 2: 90
Student ID 4: 92
```

-Alex

References:

Oracle Doc. (n.d.). *Collection (Java SE 21)* [Java Platform, Standard Edition Java API Specification]. Oracle. Retrieved from https://docs.oracle.com/en/java/javase/21/docs/api/java.base/java/util/Collection.html

Tutorials Point. (n.d.). *Java interfaces*. Tutorials Point. Retrieved from https://www.tutorialspoint.com/java/java_interfaces.htm