# Java Programming 1 - Week 7 Notes

Hia Al Saleh

October 17th, 2024

## Contents

## 1 Introduction to Arrays

This week's lesson focused on the concept of arrays. Arrays are a type of data structure used to store large amounts of data in one convenient place. Each item in an array is identified by an index number, starting from 0, which makes accessing data straightforward.

### 1.1 Array Usage

Arrays allow us to:

- Store multiple values of the same data type.
- Access any stored value using its index.
- Use them as function parameters and return types.

In Java, arrays can store various types of data, such as integers, strings, or even objects. Here are the most common ways to create arrays:

```
Type[] arrayName = new Type[number of values];
Type[] arrayName = {comma-delimited values};
Type arrayName[] = new Type[number of values];
```

## 2 Working with Arrays in Java

Arrays behave similarly to variables, but with the capability of storing multiple values under the same name. Here are some basic examples of arrays with strings and integers:

### 2.1 String Array Example

```
String[] list = new String[3];
String[] list = {"Good", "Better", "Best"};
String list[] = new String[3];
```

In this example, we define three string elements in an array called `list`. Accessing these elements is done by their index:

- `list[0]` contains `"Good"`.
- `list[1]` contains `"Better"`.
- `list[2]` contains `"Best"`.

### 2.2 Integer Array Example

Similarly, we can define an array that stores integers:

```java
int[] grades = new int[6];
int[] grades = {89, 10, 99, 100, 78, 55};
int grades[] = new int[6];
```

This allows us to store six integer values, such as student grades, and access them using their index.

## 3 Class List Generator Project

One of the week's exercises involved building a class list generator. The goal was to allow a teacher to input the names of students in a class, and then sort and display the list in alphabetical or reverse alphabetical order.

### 3.1 Plan of Action

The plan involves the following steps:

1. Create an array to store the student names.
2. Use a loop to gather student names from the teacher.
3. Ask the user if they want to display the names in alphabetical or reverse alphabetical order.
4. Sort the array and display the result.

### 3.2 Code Implementation

Below is the full implementation of the Class List Generator:

```java
import java.util.Arrays;
import java.util.Scanner;

public class ClassListGenerator {

    public static void main(String[] args) {
        // Initialize Scanner for user input
        Scanner input = new Scanner(System.in);

        // Ask the teacher for the number of students
        System.out.print("Enter the number of students: ")
            ;
        int numberOfStudents = input.nextInt();
        input.nextLine();  // Consume the newline
            character
        // Create an array to store the students' names
        String[] studentNames = new String[
            numberOfStudents];

        // Populate the array with student names
        for (int i = 0; i < numberOfStudents; i++) {
            System.out.print("Enter the name of student "
                + (i + 1) + ": ");
            studentNames[i] = input.nextLine();
        }

        // Ask the user whether to sort the list in
            ascending or descending order
        System.out.print("Enter 1 for alphabetical order
            or 0 for reverse alphabetical order: ");
        int order = input.nextInt();

        // Sort the array alphabetically
        Arrays.sort(studentNames);

        // Display the sorted array in the chosen order
        if (order == 1) {
            System.out.println("Class List in Alphabetical
                Order:");
            for (String name : studentNames) {
                System.out.println(name);
            }
        } else {
            System.out.println("Class List in Reverse
                Alphabetical Order:");
            for (int i = studentNames.length - 1; i >= 0;
                i--) {
                System.out.println(studentNames[i]);
            }
        }
    }
}
```

### 3.3 Explanation of the Code

- `Scanner input = new Scanner(System.in);` initializes the scanner for taking input from the user.
- `String[] studentNames = new String[numberOfStudents];` creates an array of strings to store the student names.
- The `for` loop collects the names of the students one by one.
- `Arrays.sort(studentNames);` sorts the array in alphabetical order.

- Depending on the user's choice, the program either displays the names in alphabetical order or in reverse alphabetical order by looping through the array backwards.

### 3.4 Challenges and Solutions

One challenge in the implementation is ensuring that the correct number of students is processed. The code uses `input.nextLine()` to handle potential issues with the input buffer, especially when dealing with the newline character after `nextInt()`.

Another challenge is handling the sorting order, which is solved by using an `if-else` statement to determine how the array should be printed.

## 4 Testing and Improving the Program

It is essential to test the program with various inputs to ensure it functions correctly. For example, you can test it with:

- A small class size (e.g., 3 students).
- A large class size (e.g., 20 students).
- Edge cases such as no students or identical student names.

During testing, common errors like `ArrayIndexOutOfBoundsException` may occur if the user tries to access an index that does not exist. It's crucial to handle such exceptions by using proper checks before accessing array elements.

## 5 Homework

As a follow-up to this exercise, read Chapter 7 of the textbook. This chapter dives deeper into topics such as:

- Logical operators (`&&`, `||`).
- Typecasting and how it applies to arrays and other data structures.

Next week, we will cover these topics in detail and explore their use in Java programming.