Arrays
Overview

- Often, we need to store and manipulate several variables; a program will use an array to store a collection of variables of the same type.
- In this module we will learn how to store several variables inside an array.
- You can think of an array as a collection of variables in which each variable occupies a specific position.
- Example:
  - Store your friends’ phone numbers inside an array named *besties*.
Learning Objectives

- To be able to declare an array
- To be able to initialize an array with the `new` keyword
- To be able to initialize an array with an initializer list
- To be able to access array values
- To be able to modify array values
- To be able to traverse an array using the `for`-loop
- To be able to traverse an array using the enhanced `for`-loop
- To be able to solve problems using arrays
Modeling with arrays

- **Arrays** are used to store **several elements of the same type**
- Arrays are a type of data structure
- An array is like a list in real life: list of students, list of songs (playlist), etc.
- An array has a **fixed length** (the number of elements that can be stored in it)
- Each element in an array has a **position or index**:
  - The first element is at index **0**
  - The last element is at index **(length of the array) - 1**

Element type decided with variable declaration

Index numbering similar to strings
Declaring and Creating Arrays

- Arrays are an object type
- To declare an array, you write

  ```java
  TypeOfElements[] arrayName = new TypeOfElements[length];
  ```

Example:

```java
int[] myArray = new int[6]
```
Create an array of integers of length 6

```java
Student[] studentsArray = new Student[10]
```
Create an array of Student of length 10
Default Values

- When we initialize an array with `new`, Java will make each element some “default” value depending on the type.

Example: `int[] myArray = new int[6]`
Integers in the array will start as 0.

`Student[] studentsArray = new Student[10]`
Strings (or any other object type) in the array will start as null.

doubles start as 0.0, Booleans start as false.
To declare an array using an *initializer list*, you write

```
TypeOfElements[ ] arrayName = {element1, element2, ...};
```

Example:

```java
int[] myArray = {1, 2, 4, 5, 6};
```

What is the length of `myArray`?

```
1 2 4 5 6
```

```java
String[] names = {"Kayla", "Malik", "Serena"};
```

```
"Kayla" "Malik" "Serena"
```

*Initializer list* only works on the same line you are declaring the array.
Array length

- The **length** of the array **cannot be changed after initialization**
- To get the length you write `arrayName.length`

```java
int[] myArray = {1, 2, 4, 5, 6};
myArray.length ➔ 5

String[] names = new String[10];
names.length ➔ 10
```
Accessing Array Values

- Elements inside an array are **accessed** based on their **position** or **index**
- To access the element at position $i$ you write $\text{arrayName}[i]$
  
  myArray[3] ➔ 11  
  myArray[0] ➔ 45  
  myArray[myArray.length - 1] ➔ 3

- Trying to access an element at a **position** $< 0$ or $\geq \text{array.length}$ will raise an **Error** (ArrayIndexOutOfBoundsException)
Modifying Array Values

- To assign a new value at the position $i$ you write
  \[ \text{arrayName}[i] = \text{new\_value}; \]

```java
int[] myArray = {1, 2, 4, 5, 6};
myArray[0] = 5; \Rightarrow \{5, 2, 4, 5, 6\}
myArray[4] = 10; \Rightarrow \{5, 2, 4, 5, 10\}
```

- Trying to modify an element at a position $< 0$ or $\geq \text{array\_length}$ will raise an Error (ArrayIndexOutOfBoundsException)
Traversing Arrays

- Often, we need to iterate through all the elements inside an array:
  - To find a specific value
  - To perform a computation: for example, the sum of all the elements
  - Many more reasons
- We use a **loop** to **iterate** through an array
Array iteration with a For-loop

- Start the **loop control variable** at 0 (the smallest position)
- Keep looping as long as the control loop variable is within the bounds of the array (< `array.length`)
- Increment the loop control variable between each iteration
- Use the **loop control variable** to access the **elements** inside the array
Array iteration with the Enhanced For-loop

- The enhanced for loop is called the for each loop
- The for each loop does not use an index to traverse an array
- The for each loop uses a variable that will refer to each value in the array from the first position (0) to the last (arrayname.length - 1)
- The iteration stops after the variable reaches the last element
- To use the for each loop you write

```java
for (TypeOfElements variable : arrayName){
    //body
}
```
Array iteration with the Enhanced For-loop

String[] names = {"Kayla", "Malik", "Serena"};

for(String name : names){
    System.out.println(name);
}

Will print:

"Kayla" // first value of name
"Malik"  // second value of name
"Serena" // third value of name