INTRODUCTION TO

JAVA

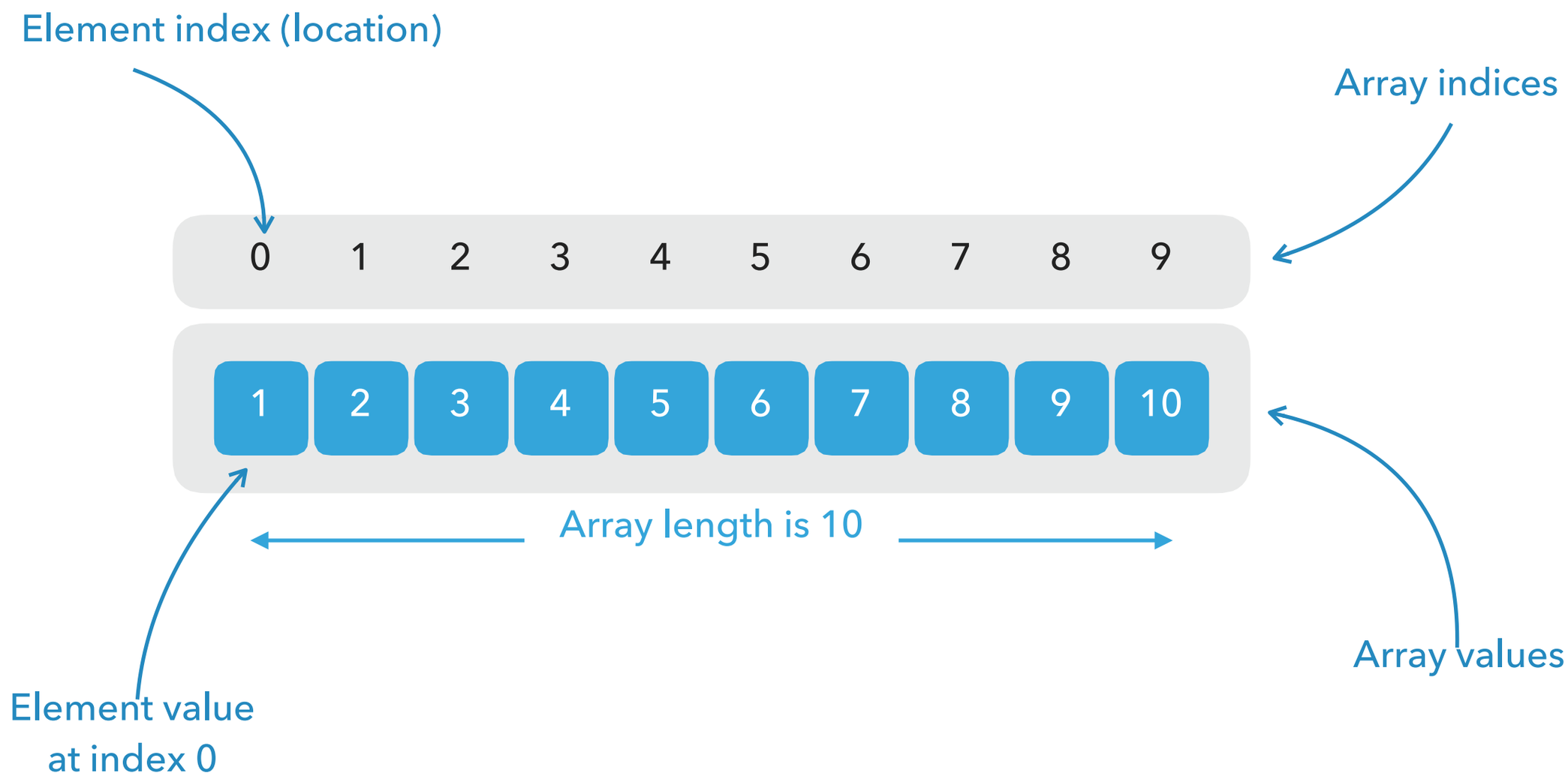
ARRAYS

OVERVIEW

DEFINITION

- An array is a container object that holds a fixed number of values of a single type
- The length of an array is established when the array is created
- After creation, its length is fixed

ARRAYS VISUALISATION



ARRAYS DECLARATION: SYNTAX

- Array declaration without instantiation

```
type[] name;
```

- Array declaration with instantiation

```
type[] name = new type[size];
```

- Array declaration with inline initialization

```
type[] name = {var1,.., varN};
```

ARRAY DECLARATION: INSTANTIATION CODE EXAMPLE

Code

```
int[] leapYears = new int[3];

leapYears[0] = 2020; leapYears[1]= 2016;  leapYears[2] = 2012;

System.out.println("Leap years = " + Arrays.toString(leapYears));
```

Console output

```
Leap years = [2020, 2016, 2012]

Process finished with exit code 0
```

ARRAY DECLARATION: INLINE INITIALIZATION CODE EXAMPLE

Code

```
int[] leapYears = {2020, 2016, 2012};

System.out.println("Leap years = " + Arrays.toString(leapYears));
```

Console output

```
Leap years = [2020, 2016, 2012]

Process finished with exit code 0
```

PROCESSING ARRAYS

WORKING WITH ARRAYS

- When working with arrays, loops are often used because of array iterable nature
- Array contains elements of the single type and size is fixed and known in advance

1. EXAMPLE: PRINTING ARRAY CONTENT

```
public class PrintingArrayDemo {
    public staticvoid main(String[]  args) {
        String[] alphabet = new String[5];

        alphabet[0] = "A";
        alphabet[1] = "B";
        alphabet[2] = "C";
        alphabet[3] = "D";
        alphabet[4] = "E";

        for (int i = 0; i < alphabet.length; i++){
            System.out.println("[" + i + "]: " + alphabet[i]);

        }

    }

}
```

2. EXAMPLE: SUM OF ARRAY ELEMENTS

```
public class SumOfArrayElementsDemo {

    public static void main(String[] args) {

        int[] numbers = {1, 2, 3, 4, 5, 6, 7, 8, 9};
        int sum = 0;

        for (inti  = 0; i < numbers.length; i++) {
            sum += numbers[i];
        }

        System.out.println("Sum = " + sum);

    }
}
```

3. EXAMPLE: FIND SMALLEST ELEMENT IN ARRAY

```
public class SmallestArrayElementDemo {

    public static void main(String[] args) {

        int[] numbers = {61, 97, 4, 37, 12};
        int min = numbers[0];

        for (int i = 0; i < numbers.length; i++) {
            if (numbers[i] < min) {
                min = numbers[i];
            }
        }

        System.out.println("min = " + min);

    }

}
```

ADVANCED ITERATION METHODS

FOR EACH (ENHANCED) LOOP: SUMMARY

- For each loop, also known as enhanced loop, is another way to traverse the array
- There is no use of the index or rather the counter variable
- Data type declared in the foreach must match the data type of the array that you are iterating
- Can access only current element
- Significantly reduces amount of code

FOR EACH (ENHANCED) LOOP: SYNTAX

for each loop declaration

```
type[] name = {var1,.., varN};

for (type item : name) {
    statements...
}
```

Iterator specification

Statement(s) that executed inside of the loop body

FOR EACH (ENHANCED) LOOP: CODE EXAMPLE

```
public class ForEachDemo {
    public staticvoid main(String[]  args) {
        String[] dogBreeds = {
                "Beagle",
                "Golden Retriever",
                "Pug",
                "Shiba Inu"
        };

        for (String breed : dogBreeds) {
            System.out.println(breed);
        }

    }

}
```