



MODULE 2: **OBJECT ORIENTED ANALYSIS & DESIGN** **DATA STRUCTURES & ALGORITHMS**

Learn to code arrays and functions

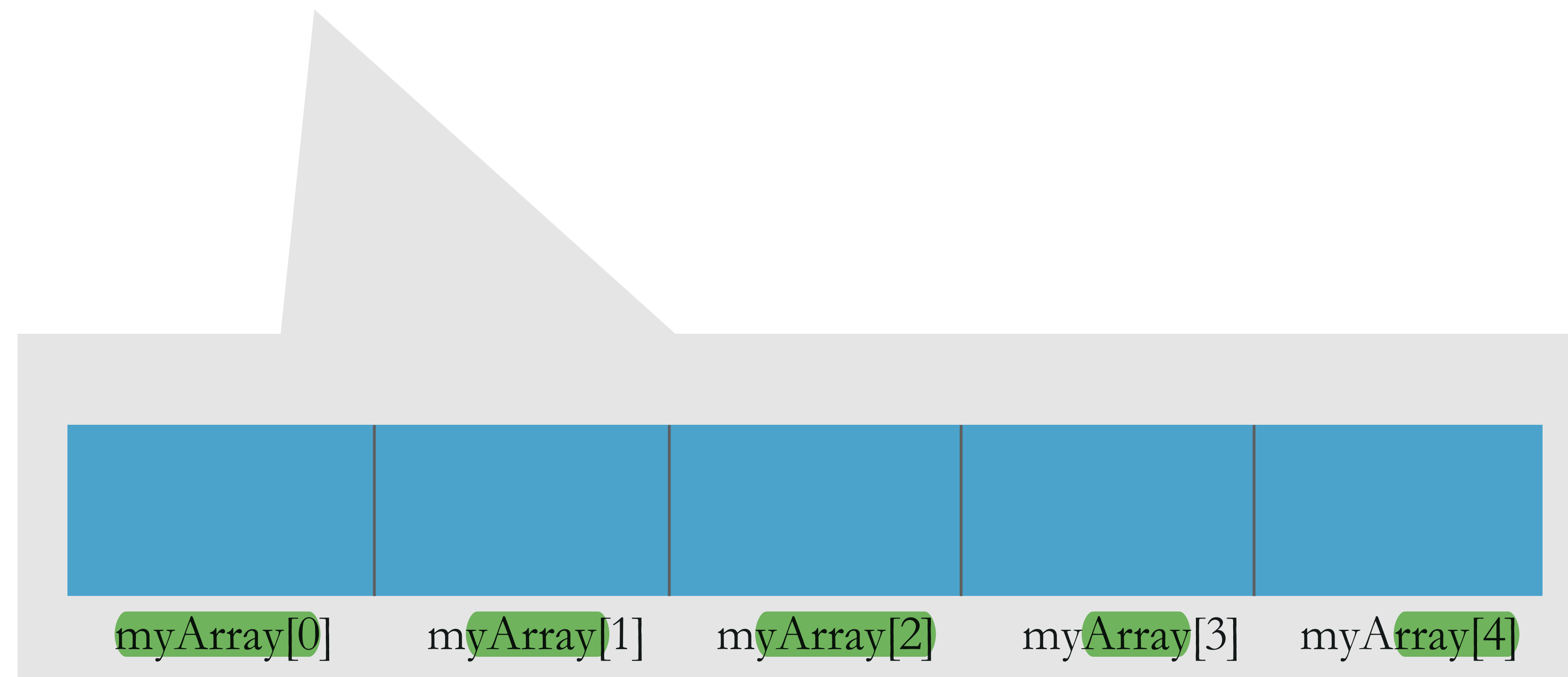
Java Arrays

Java Arrays – One Dimensional

Array is an object which contains fixed number of elements of a similar data type under same name

```
arrayRefVar = new dataType[arraySize];
```

```
myArray = new int[5]
```



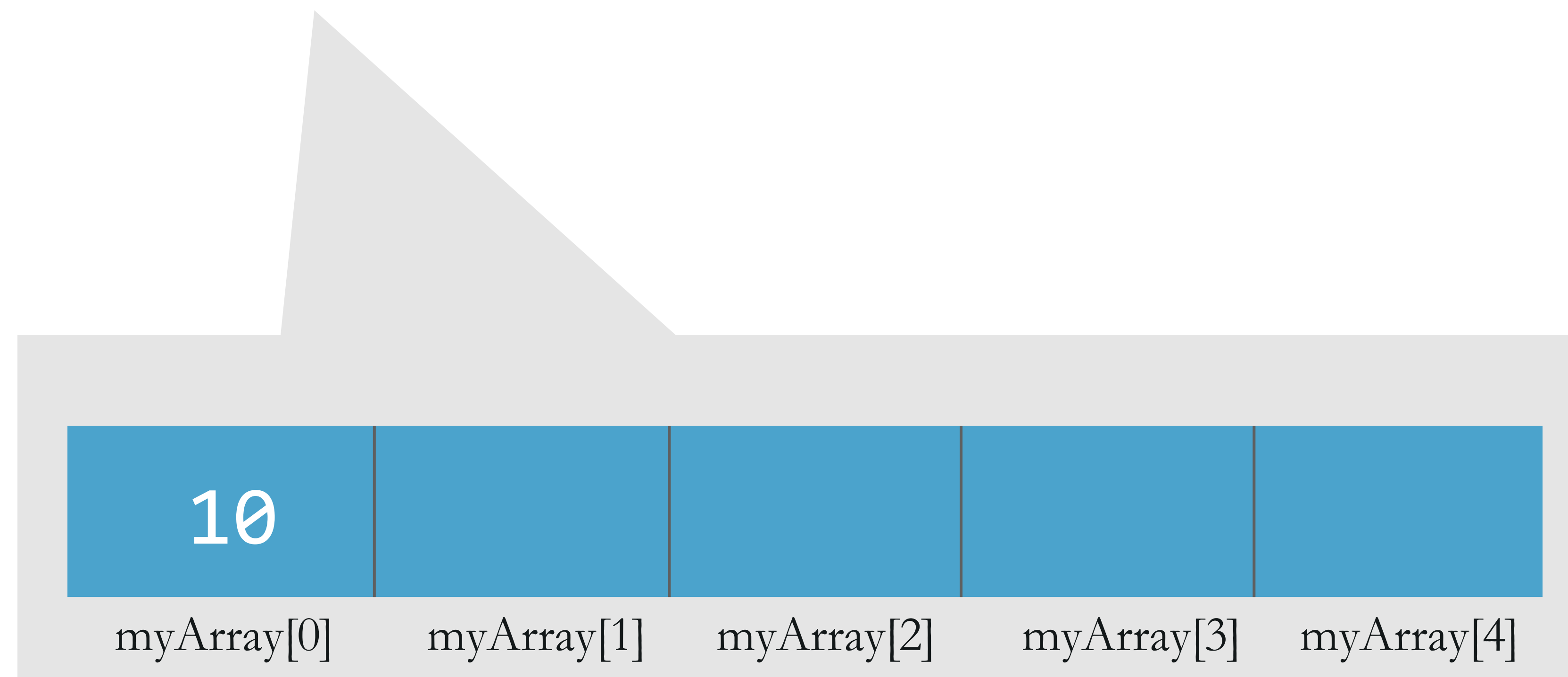
```
myArray[0] = 10
```


Java Arrays – One Dimensional

Array is an object which contains fixed number of elements of a similar data type under same name

```
arrayRefVar = new dataType[arraySize];
```

```
myArray = new int[5]
```



```
myArray[1] = 20
```

```
myArray[2] = 30
```

```
myArray[3] = 40
```

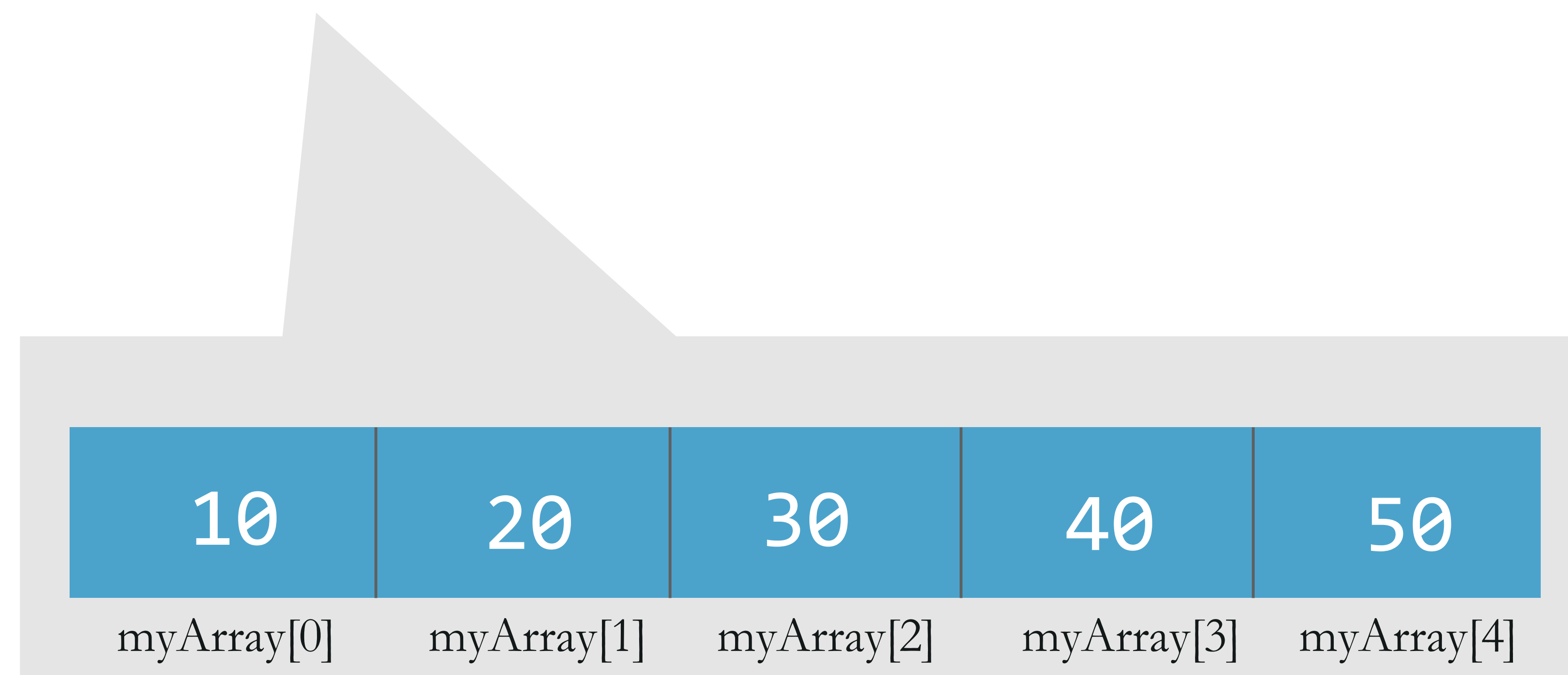
```
myArray[4] = 50
```


Java Arrays – One Dimensional

Array is an object which contains fixed number of elements of a similar data type under same name

```
arrayRefVar = new dataType[arraySize];
```

```
myArray = new int[5]
```

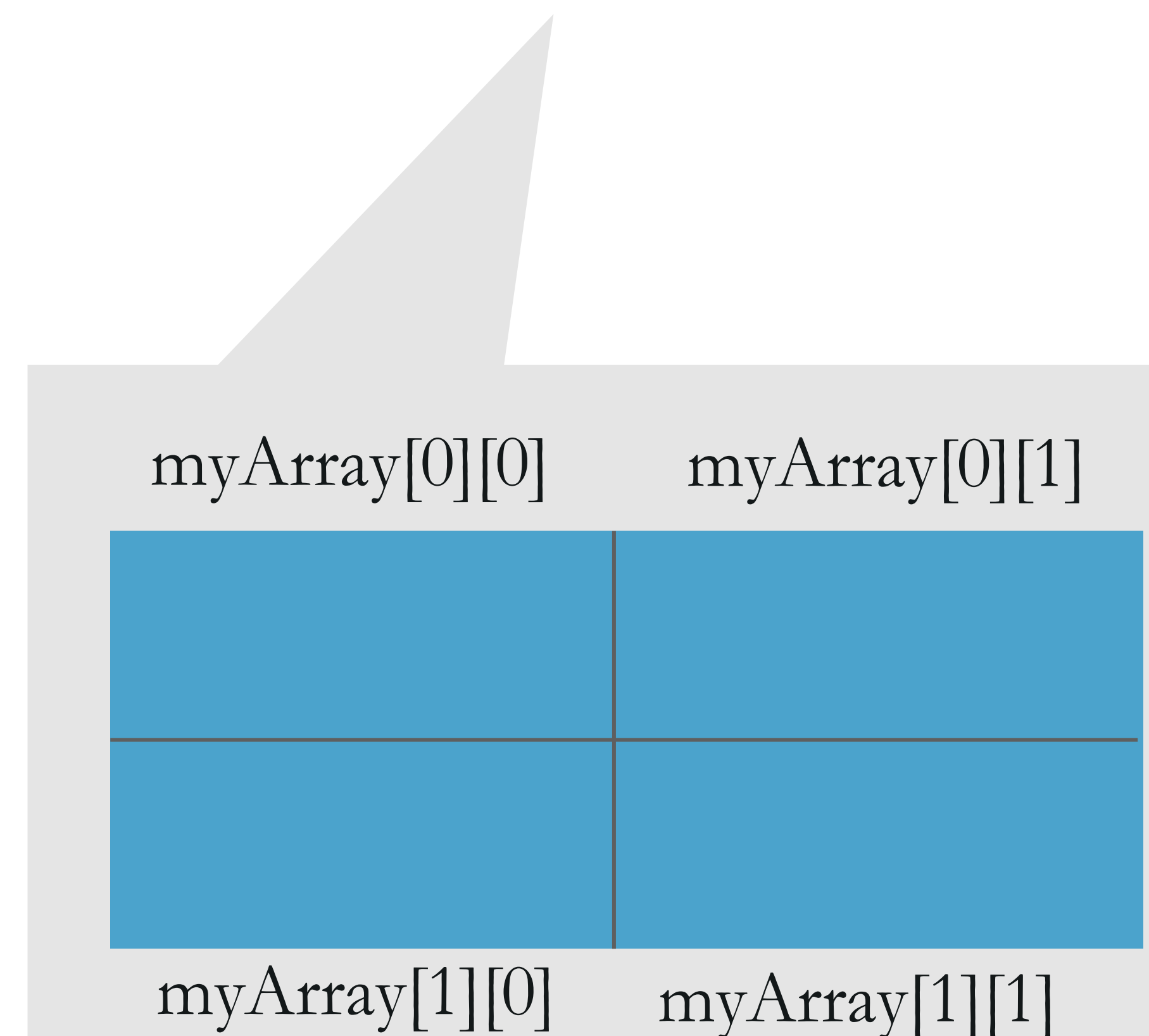


Java Arrays – Two Dimensional

Like a 1D array, a 2D array is also a collection of data cells, all of the same type, which can be given a single name

```
datatype[][] arrayRefVar = new dataType[row][col];
```

```
int[][] myArray = new int[2][2]
```




```
myArray[0][0] = 100
```


Java Arrays – Two Dimensional

Like a 1D array, a 2D array is also a collection of data cells, all of the same type, which can be given a single name

```
datatype[][] arrayRefVar = new dataType[row][col];
```

```
int[][] myArray = new int[2][2]
```



myArray[0][0]	myArray[0][1]
100	
myArray[1][0]	myArray[1][1]

```
myArray[0][1] = 200
```

```
myArray[1][0] = 300
```

```
myArray[1][1] = 400
```


Java Arrays – Two Dimensional

Like a 1D array, a 2D array is also a collection of data cells, all of the same type, which can be given a single name

```
dataType[][] arrayRefVar = new dataType[row][col];
```

```
int[][] myArray = new int[2][2]
```

myArray[0][0]	myArray[0][1]
100	200
300	400
myArray[1][0]	myArray[1][1]

Java Classes

A class may contain

```
class <class_name>{  
    field;  
    method;  
}
```

Fields

Methods

Constructors

Blocks

Nested Class & Interface

Java Methods

Java Methods

A method is a set of **code** that is **grouped together** to perform a **specific operation**

A **method** must be written **inside a class**

Each method has its own **signature**

Java provides **two types** of methods

Pre Defined or **Standard**
Library Methods

User Defined Methods

Java User Defined Methods

To use a method, you need to perform two steps:

Method Initialization

Method Invocation

Java Methods

Method Initialization

```
modifier returnType nameOfMethod (Parameter List)
{
    // method body
}
```

- ✓ A method can be parameterized or non-parameterized
- ✓ Method definition consists of a method header and a method body
- ✓ You can **Overload Method** i.e. Provide same name to more than one method but their data type or parameter list must be different

Java Methods

Method Invocation

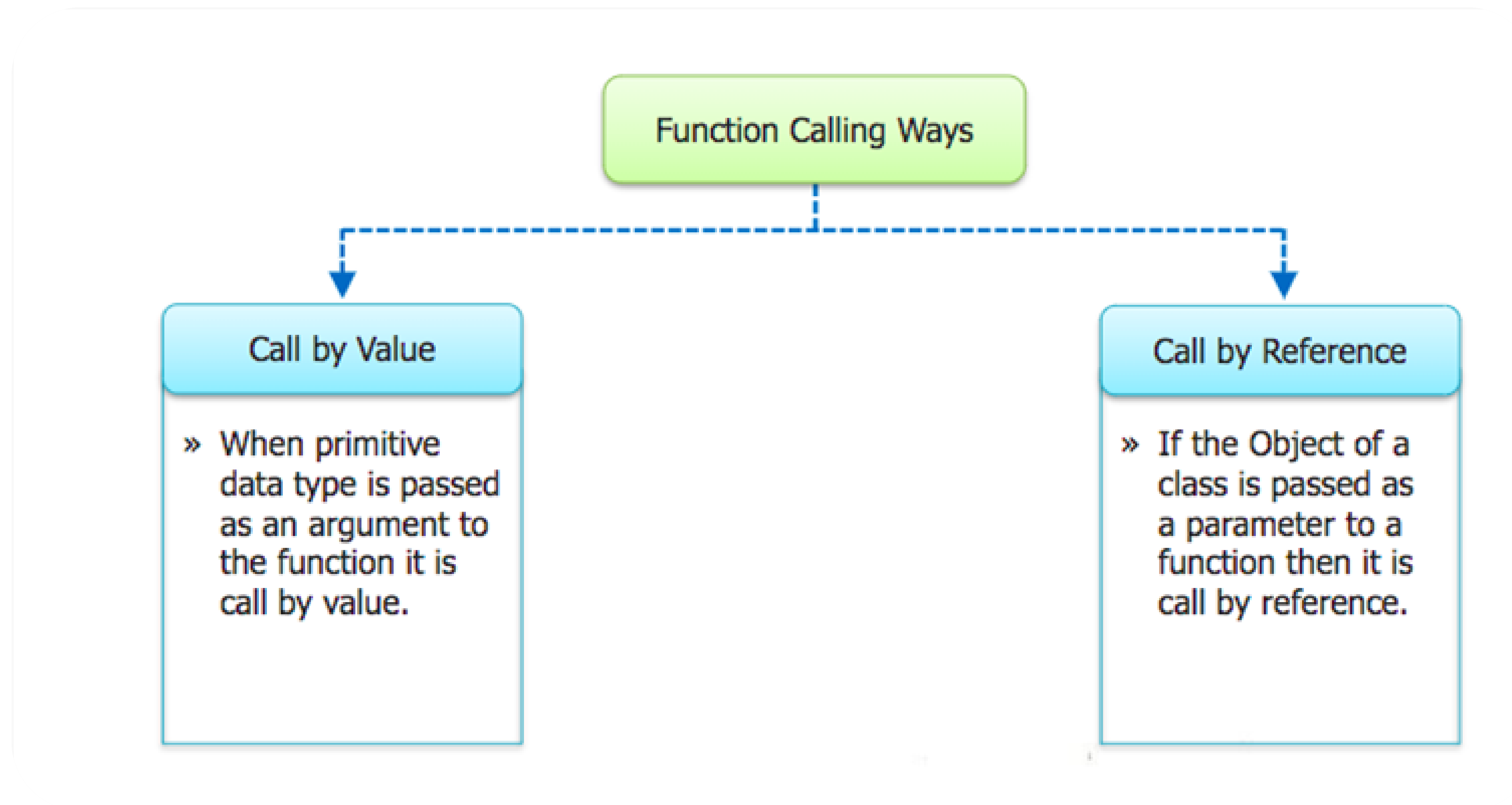
```
methodName()
```

```
methodName(parameter1, parameter2...)
```

- ✓ To use a method it needs to be invoked or called
- ✓ When a program invokes a method, the program control gets transferred to the called method
- ✓ A method can be called in two ways:
 - Call by Value
 - Call by Reference

Method Value Vs Reference

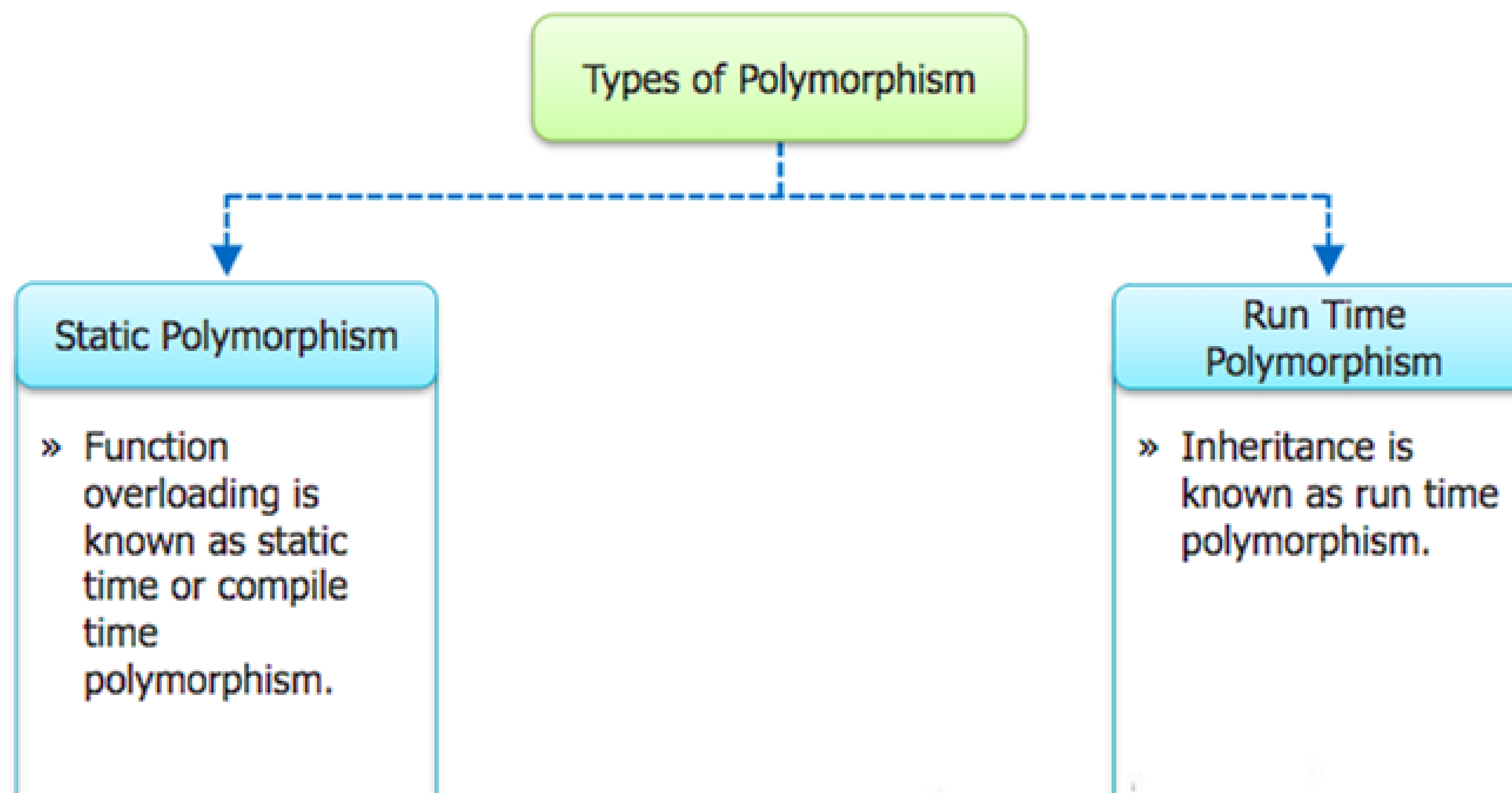
Value Vs Reference



Polymorphism

Polymorphism

Polymorphism means the system behaves differently in different programming context.



Polymorphism

Functions by different operators in a Calculator.



"÷" operator

÷ can divide:

- two integers
- two decimals

"+" operator

+ can add:

- two integers
- three integers
- two decimals
- three decimals

Thank You!