Learn To Code

Arrays and Functions

Content

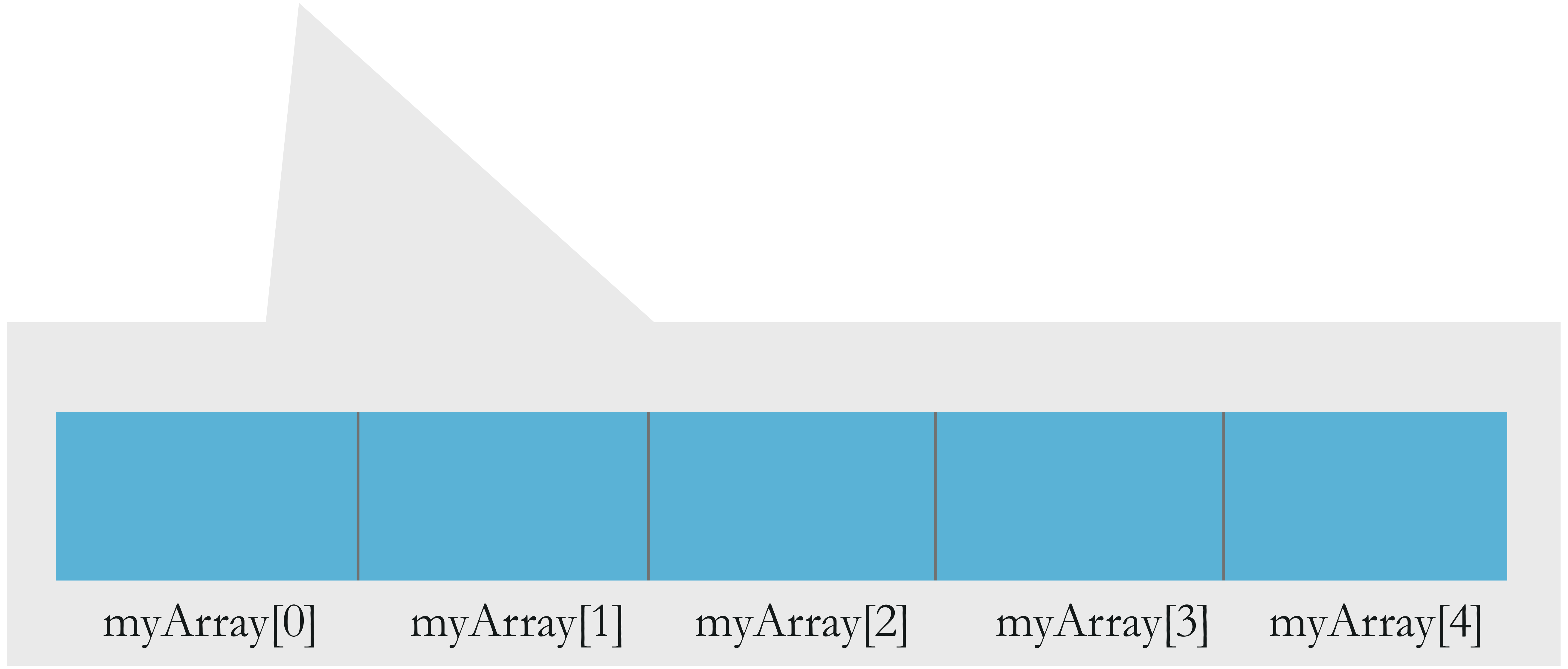
Java Arrays

Java Arrays – 1 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]
```

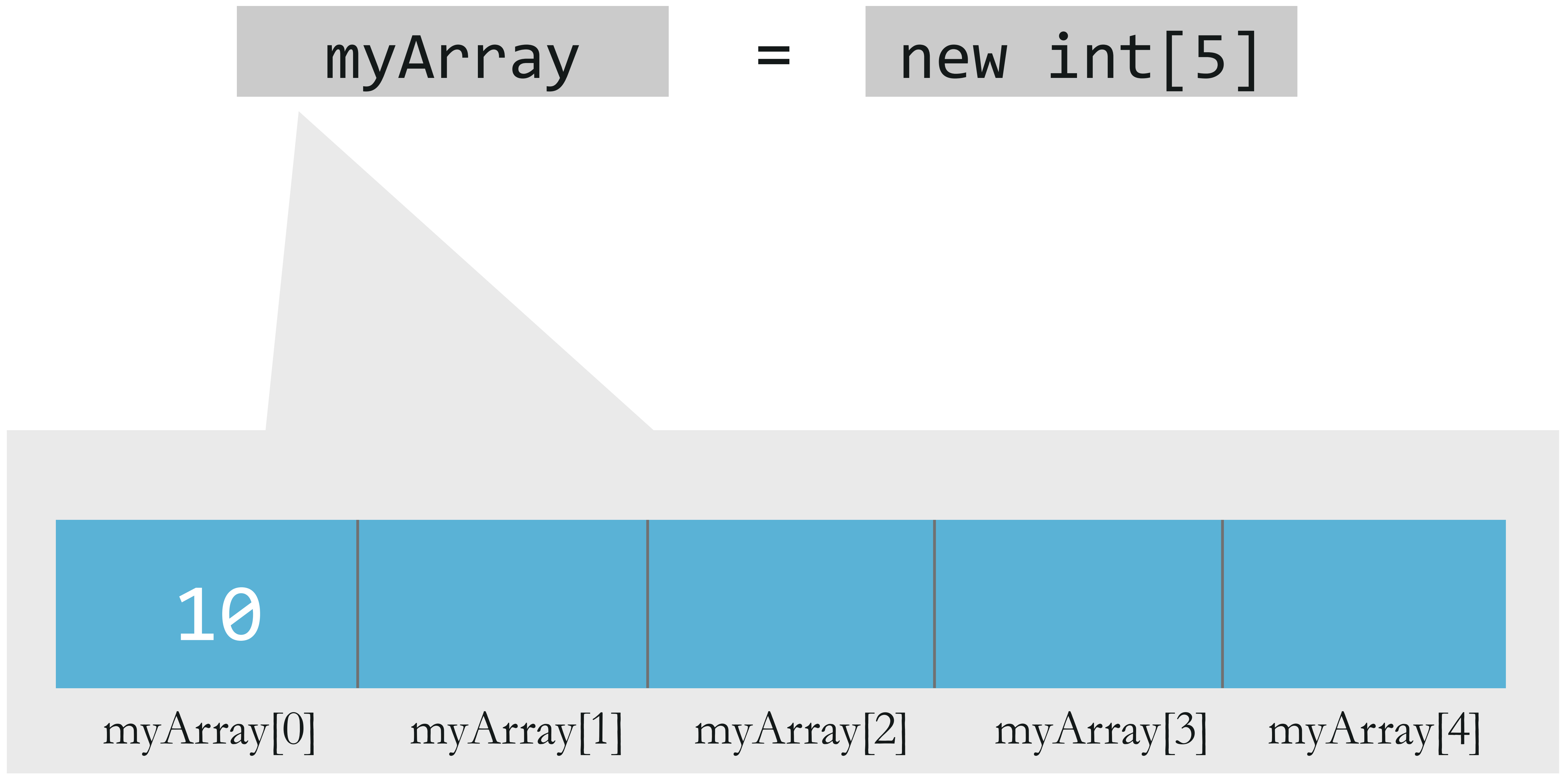


```
myArray[0]  =  10
```

Java Arrays – 1 Dimensional

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

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

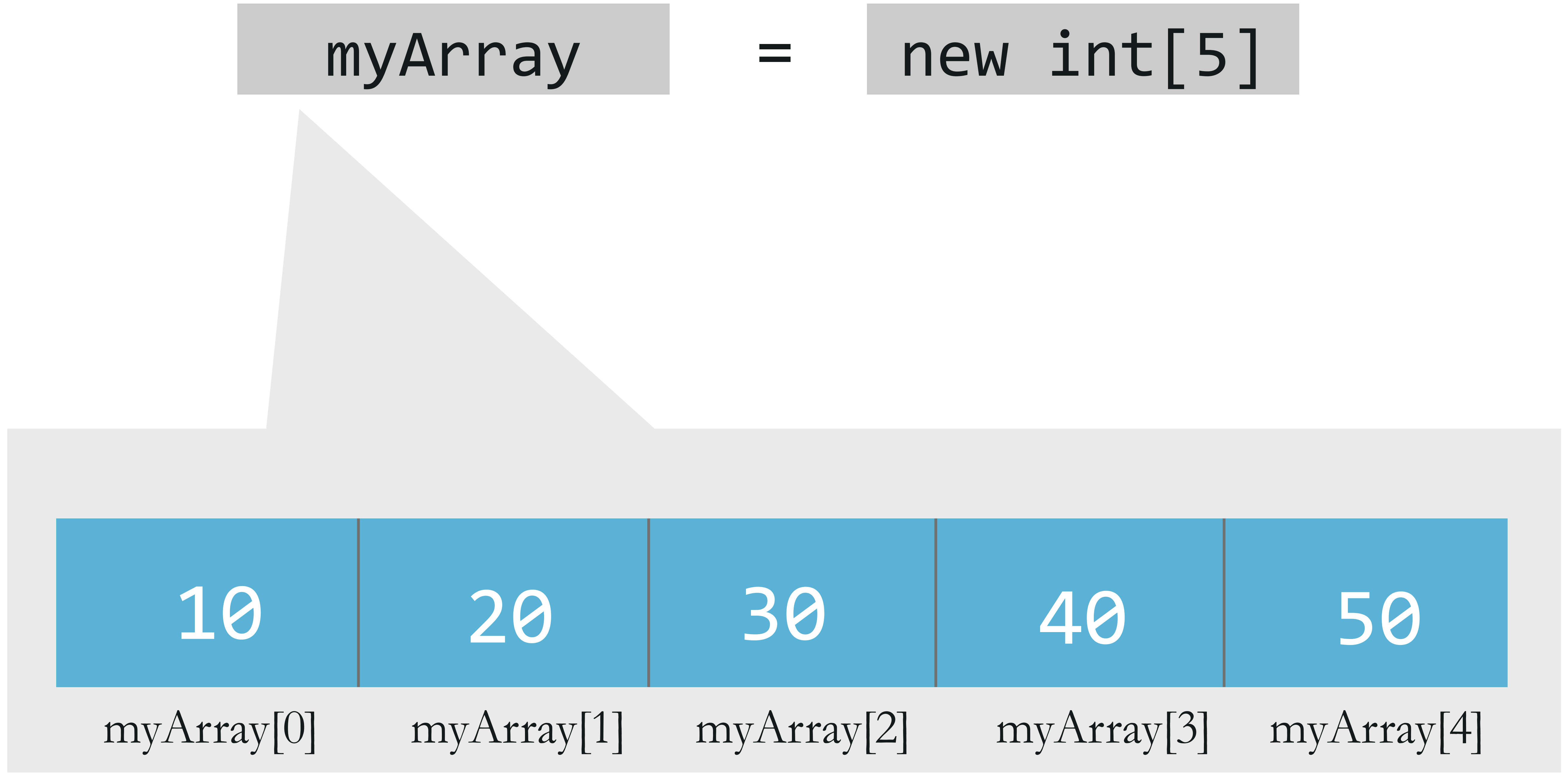


```
myArray[1]  =  20
myArray[2]  =  30
myArray[3]  =  40
myArray[4]  =  50
```

Java Arrays – 1 Dimensional

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

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

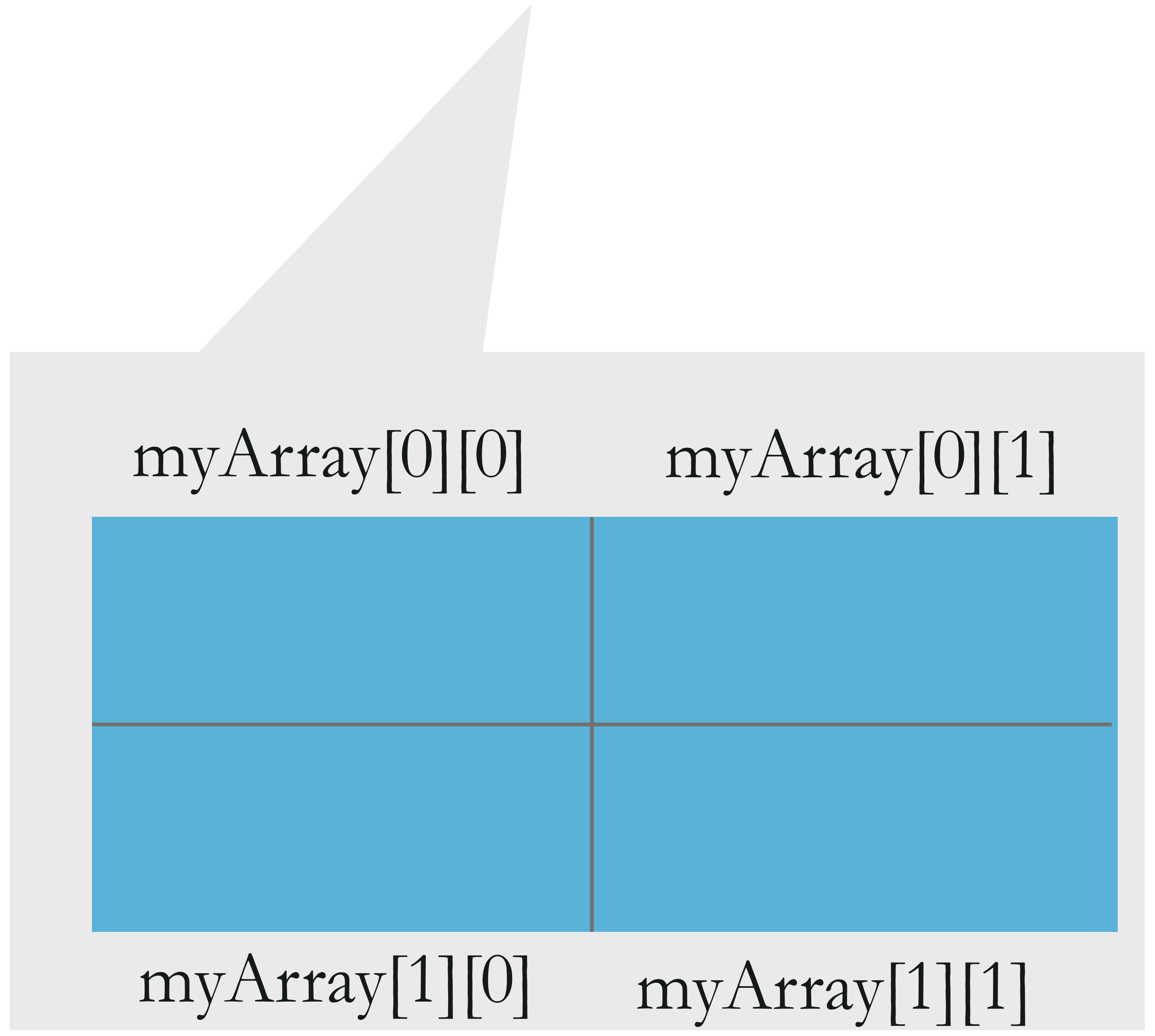


Java Arrays – 2 Dimensional

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]
```



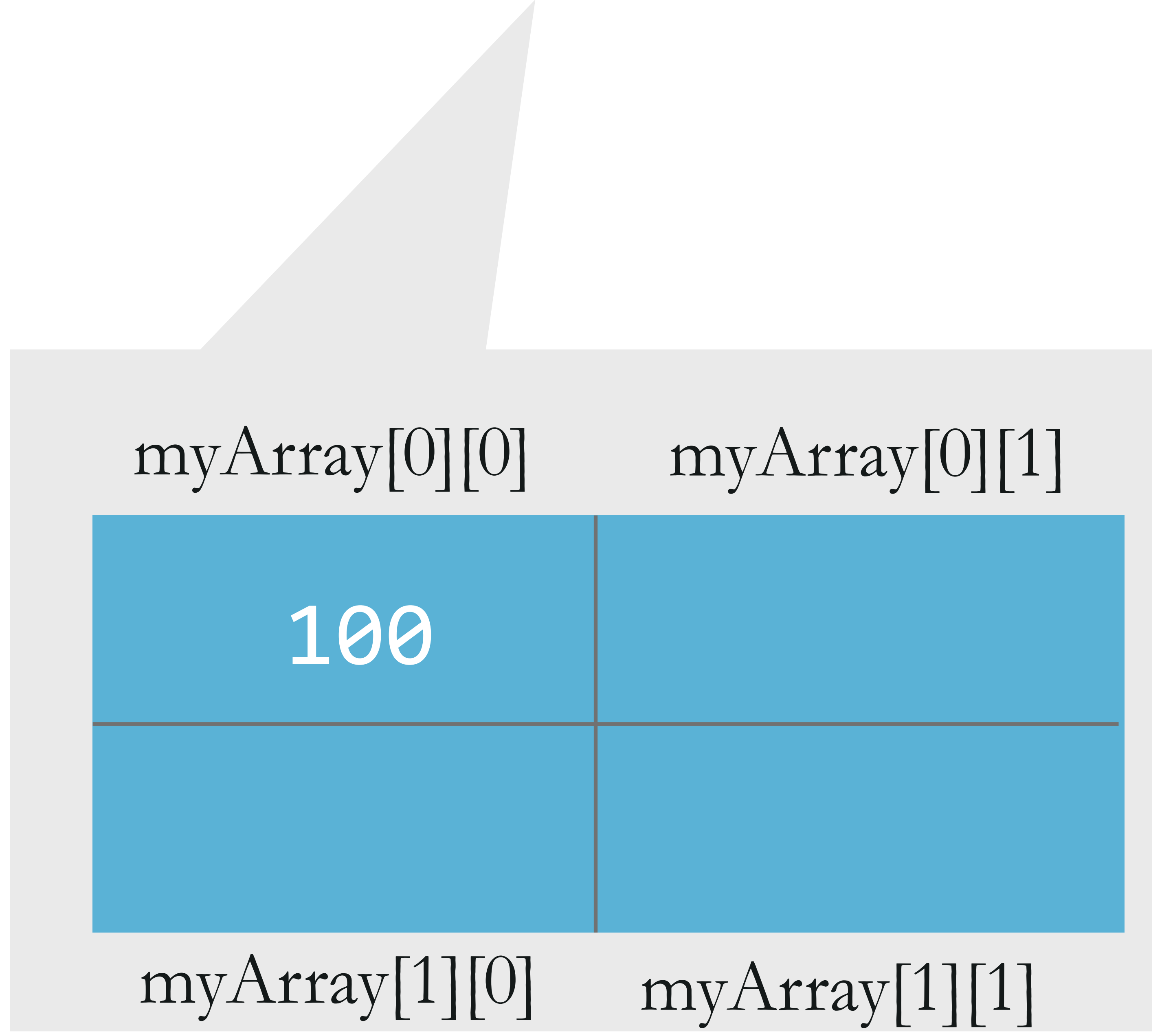
```
myArray[0][0]  =  100
```

Java Arrays – 2 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]
```



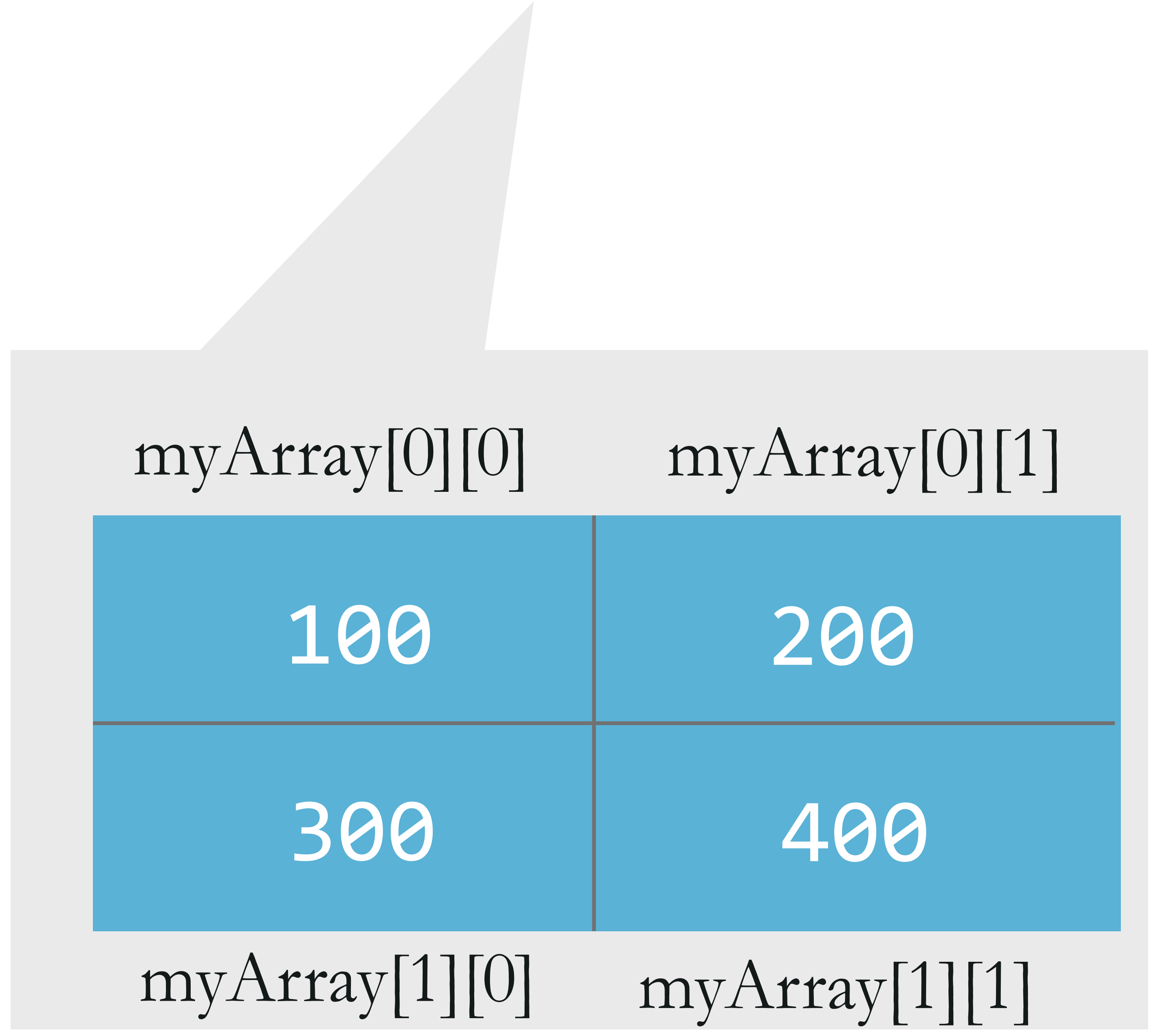
```
myArray[0][1]  =  200
myArray[1][0]  =  300
myArray[1][1]  =  400
```

Java Arrays – 2 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]
```



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...)
```

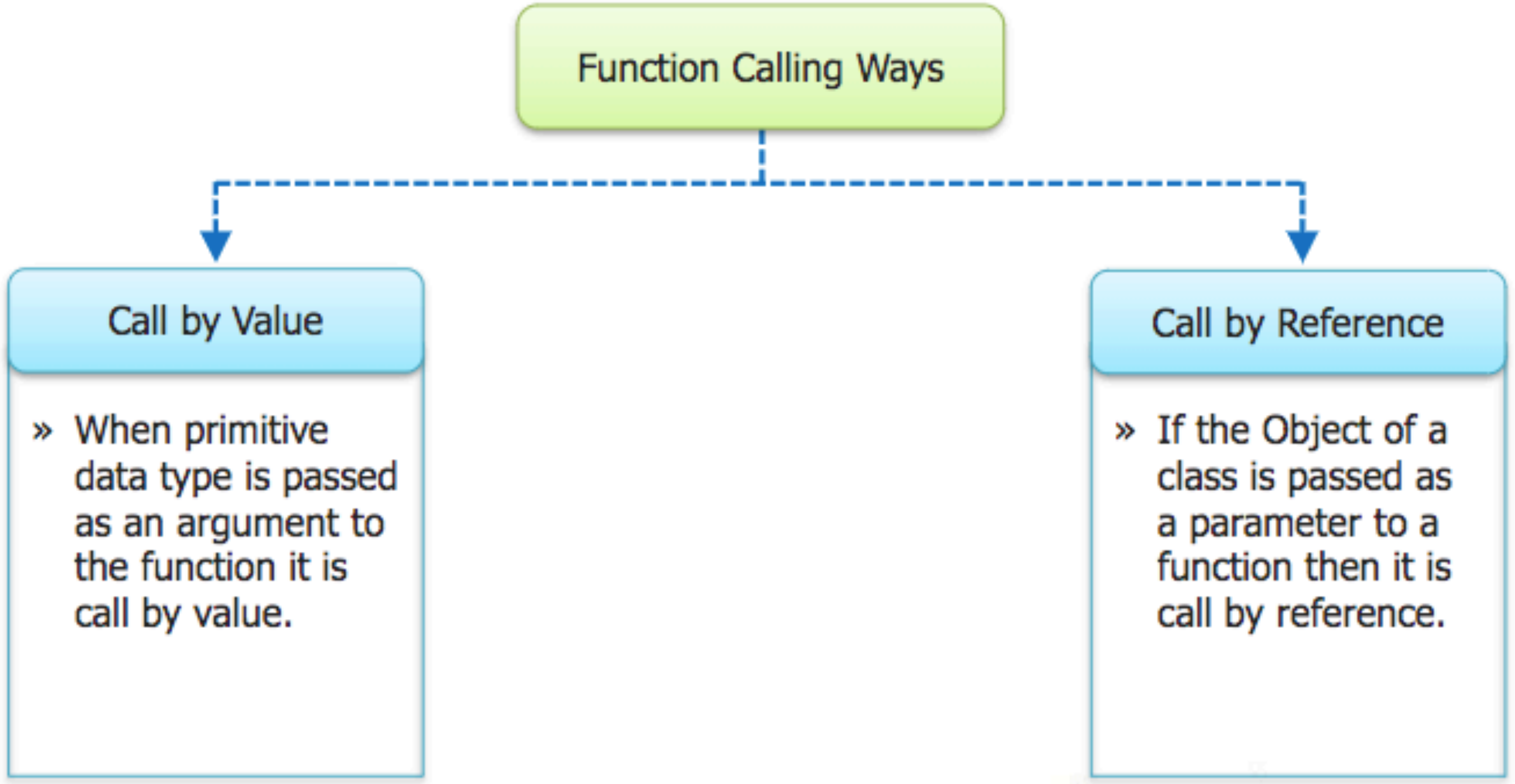
✓ To use a method it needs to be invoked or called

✓ 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

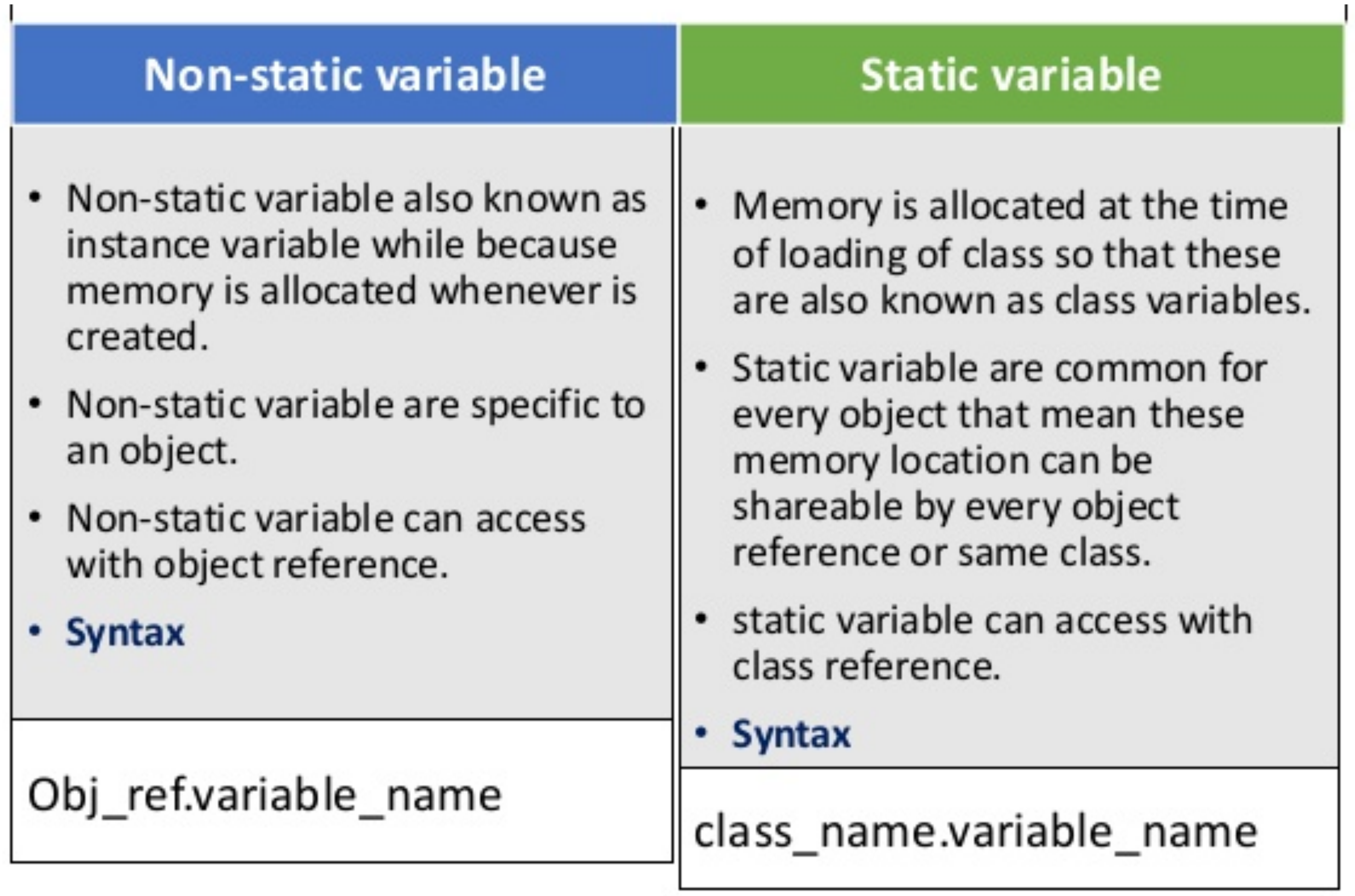


» When primitive data type is passed as an argument to the function it is call by value.

» If the Object of a class is passed as a parameter to a function then it is call by reference.

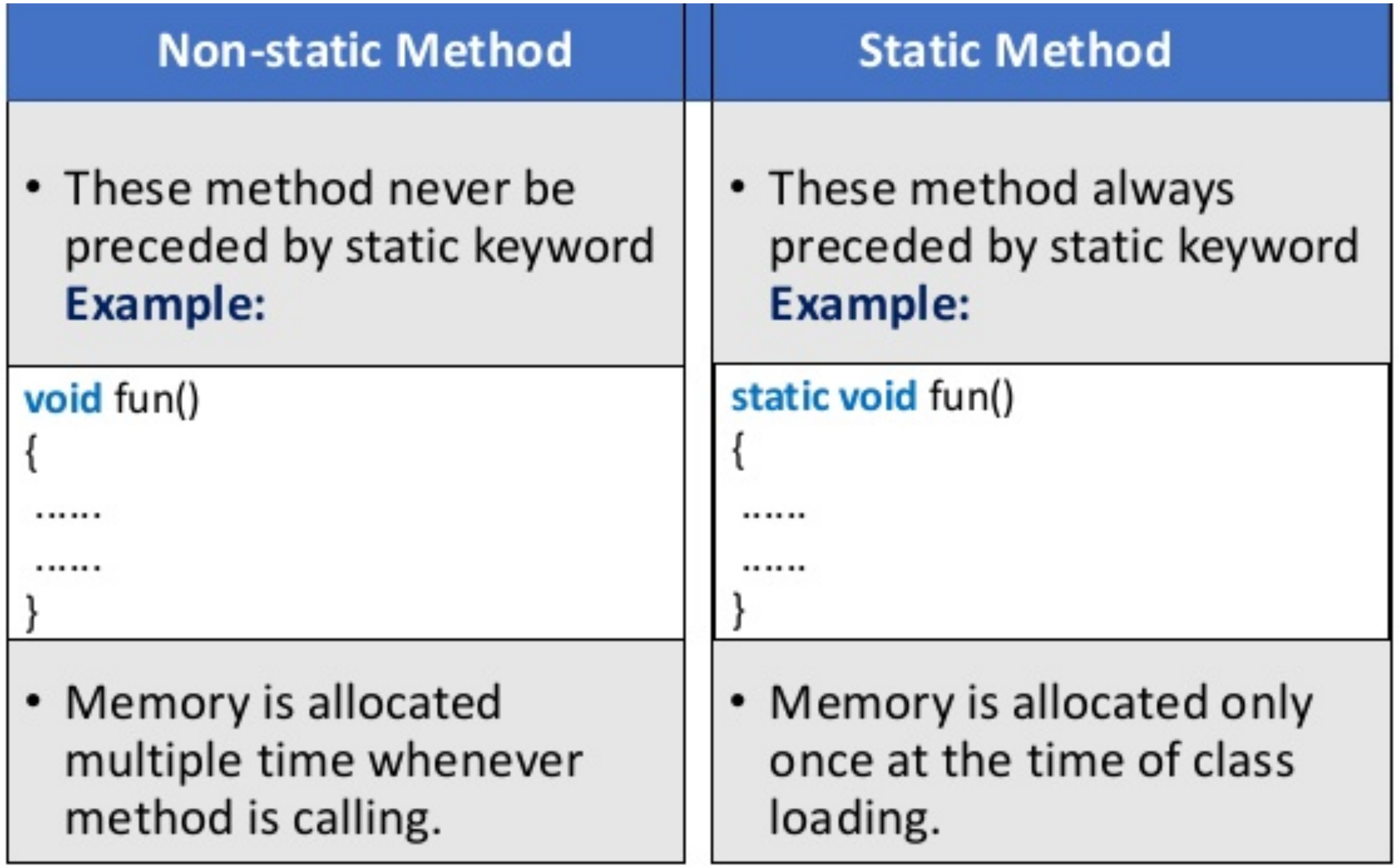
Static Vs Non-Static

Static vs Non Static



Non-static variable	Static variable
• Non-static variable also known as instance variable while because memory is allocated whenever is created. • Non-static variable are specific to an object. • Non-static variable can access with object reference. • **Syntax**	• Memory is allocated at the time of loading of class so that these are also known as class variables. • Static variable are common for every object that mean these memory location can be shareable by every object reference or same class. • static variable can access with class reference. • **Syntax**
Obj_ref.variable_name	class_name.variable_name

Static vs Non Static

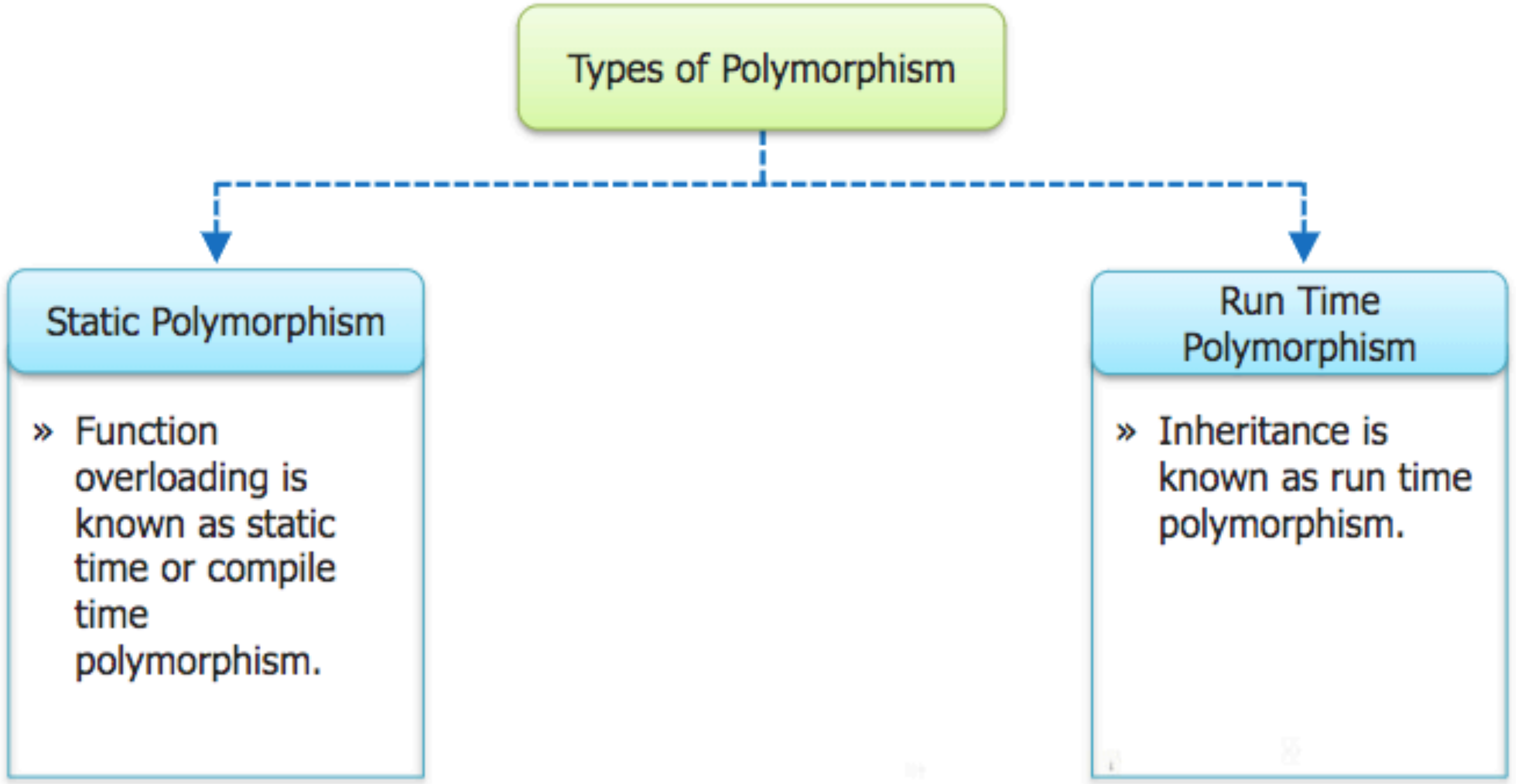


Non-static Method	Static Method
• These method never be preceded by static keyword **Example:**	• These method always preceded by static keyword **Example:**
`void fun() { ...... ...... }`	`static void fun() { ...... ...... }`
• Memory is allocated multiple time whenever method is calling.	• Memory is allocated only once at the time of class loading.

Polymorphism

Polymorphism

Polymorphism means the system behaves differently in different programming context.


Types of Polymorphism

Static Polymorphism

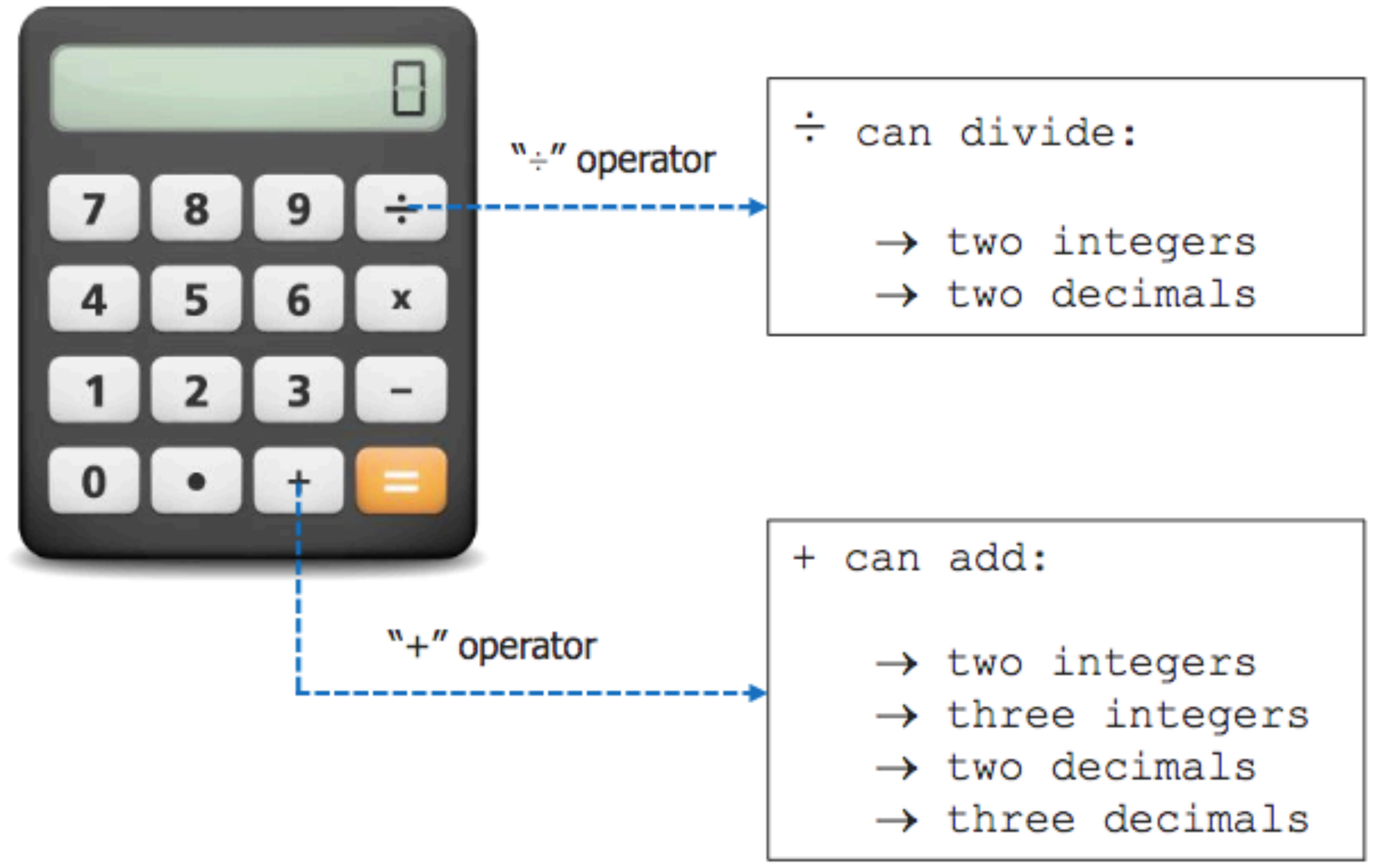
- » Function overloading is known as static time or compile time polymorphism.

Run Time Polymorphism

- » Inheritance is known as run time polymorphism.

Polymorphism

Functions by different operators in a Calculator.



Thank You