Introduction to Java

What is Java?

Java is a general-purpose, high-level, object-oriented programming language.

It was developed by James Gosling at Sun Microsystems in 1995

Principles of Java

James Gosling, creator of Java

There were 5 main goals in the creation of Java.

It must be simple, object-oriented, and familiar.

It must be robust and secure.

It must be architecture-neutral and portable.

It must execute with high performance.

It must be interpreted, threaded, and dynamic.

Origin behind the name Java

James Gosling, Mike Sheridan, and Patrick Naughton initiated the Java language project in June 1991.

Java is the name of a beautiful island in Indonesia. It is also said that the first coffee was named right here after java coffee. The name was chosen by James Gosling during the daytime when he was enjoying a cup of coffee near his office. Java was initially called by the name: OAK.

However, in the wake of the Oak Technologies, the team had officially decided to rename it. The options they considered were Silk, Revolutionary, Dynamic, DNA, and Jolt. Even though Silk was further selected, they decided to go along with Java as it was unique, and a lot of people preferred it.

Sun Microsystems released the first public implementation as Java 1.0 in 1996

Why is Java so popular ?

- Platform independence

- Fundamentally object-oriented

- Easy to learn

- Versatile

Use cases of Java

Building Android Apps

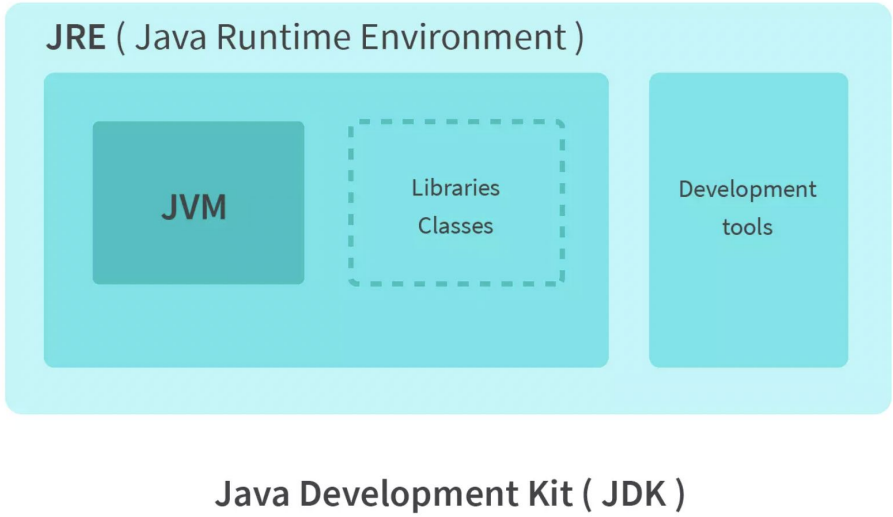
Java web applications

Software tools

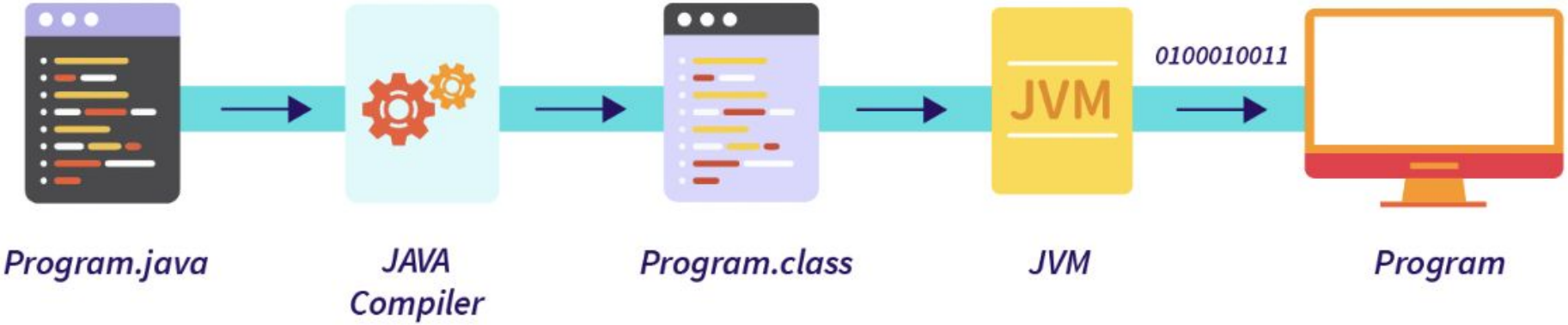
Scientific Applications

... and many more

Recap


JRE (Java Runtime Environment)
JVM
Libraries
Classes
Development
tools
Java Development Kit (JDK)

Recap


JVM
0100010011
Program.java
JAVA
Compiler
Program.class
JVM
Program

Java Arrays

Java array is a data structure that stores data of the same type in a sequential manner. An array takes a contiguous section of the memory.

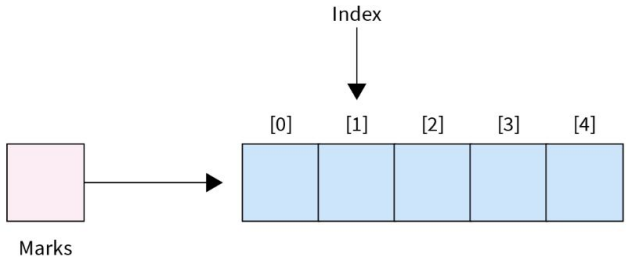
```java
class Main {
    public static void main(String[] args) {
        // Array with initial values (size determined automatically)
        int[] arr1 = {4, 8, 9, 1, 0};

        // Empty array with size = 5. Initially all values are 0.
        int[] arr2 = new int[5];

        // Alternate declarations
        int []arr3 = new int[5];
        int arr4[] = new int[5];
        int [] arr5 = new int[5];
    }
}
```

How do arrays work?

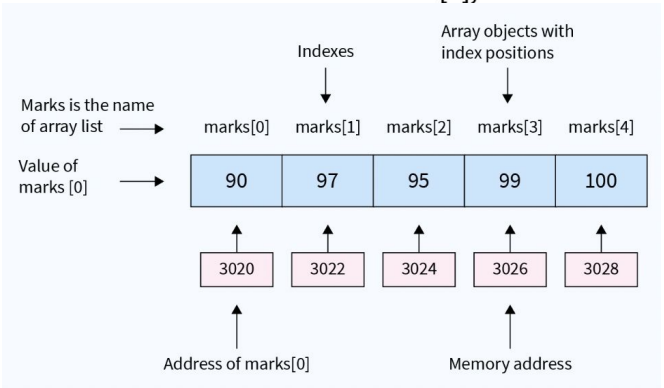
- Indexing and Memory management

```
int mark1 = 78;
int mark2 = 84;
int mark3 = 63;
int mark4 = 91;
int mark5 = 75;
```

```
marks[0] = 78;
marks[1] = 84;
marks[2] = 63;
marks[3] = 91;
marks[4] = 75;
```



int marks = new int[5];



How to initialize an array in Java?

```
datatype[] arrayName = new datatype[length]; OR
datatype arrayName[] = new datatype[length];

//[length] = Length of the array
```

```
// Both will create an int array with a length of 5.

int[] myArray = new int[5];
int myArray1[] = new int[5];
```

What is ArrayList

ArrayList is a class in java.util package which implements dynamic-sized arrays in Java. ArrayList dynamically grows and shrinks in size on the addition and removal of elements respectively.

Unlike built-in arrays, ArrayLists are resizable which means they can grow and shrink dynamically as we add and remove elements. It is relatively slower than built-in arrays but can be of great use in lots of array manipulation programs.

Syntax:

```
public class ArrayList<E>
extends AbstractList<E>
implements List<E>, RandomAccess, Cloneable, java.io.Serializable
```

Important features of Arraylist

Dynamic Resizing

Ordered

Index based

Object based

Not Synchronized

Operations in ArrayList

Declare an ArrayList of different Types

Add Element

Get Element

Add Element at a specific Index

Set Element at a specific Index

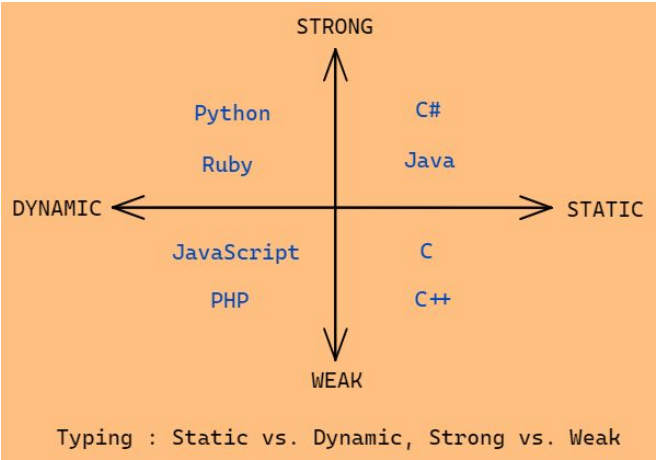
Delete Element from an Index

Size of the List

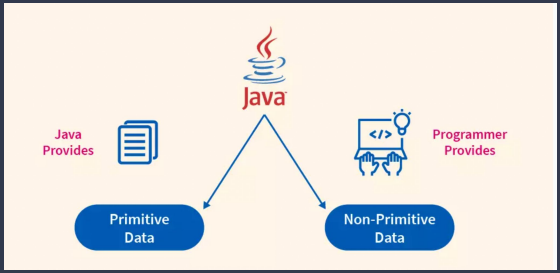
Loop/Iterate on the List

Sort the List

Types of Programming Language



Data Types in Java



Data Types mean to identify the type of the data and associate operations that can be done on the data values. Data types define the value that a variable can take.

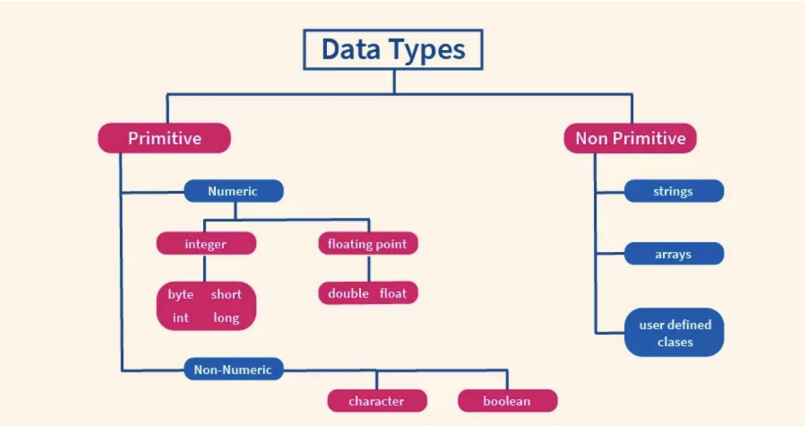
Data types also tell us information about:

The size of the memory location.

The maximum and minimum value of range that can store in the memory location.

Different types of operations can be done on the memory location.

Data types


Data Types
Primitive
Non Primitive
Numeric
strings
integer
floating point
arrays
byte short
int long
double float
user defined
clases
Non-Numeric
character
boolean

Data types sizes

Numeric Data Types and their ranges

SCALER

Data Type	Range	Size
byte	[-128 : 127]	8 bits
short	[-32,768 : 32767]	16 bits
int	[-2,147,483,648 : 2,147,483,647]	32 bits
long	[-9,223,372,036,854,775,808 : 9,223,372,036,854,775,807]	64 bits
float	[1.40239846 x 10^-45 : 3.40282347 x 10^38]	32 bits
double	[4.9406564584124654 x 10^-324 : 1.7976931348623157 x 10^308]	64 bits

Wrapper Classes

Java Wrapper classes provide a mechanism to use primitive data types as objects.

Wrapper Classes

Primitive Type	Wrapper Class
byte	Byte
short	Short
int	Integer
long	Long
float	Float
double	Double
boolean	Boolean
char	Character

Why do we need Wrapper classes?

Collections

The Java collections require objects instead of primitive types.

```java
import java.util.ArrayList;

class Main {
    public static void main(String[] args) {
        ArrayList<int> arr = new ArrayList<int>();  // Error
    }
}
```

```java
import java.util.ArrayList;

class Main {
    public static void main(String[] args) {
        ArrayList<Integer> arr = new ArrayList<Integer>();
    }
}
```

Why do we need Wrapper classes?

2. Synchronization

Java synchronization works with objects in Multithreading.

Autoboxing And Unboxing

Autoboxing is the automatic conversion that the Java compiler makes between the primitive types and their corresponding object wrapper classes.

The automatic conversion of a wrapper object to its corresponding primitive data type is known as Unboxing.

PRIMITIVE WRAPPER CLASSES ARE IMMUTABLE IN JAVA

All primitive wrapper classes (Integer, Byte, Long, Float, Double, Character, Boolean and Short) are immutable in Java, so operations like addition and subtraction create a new object and not modify the old.