Java

Shreyata Sugandhi

Corporate Trainer & Consultant

shreyata@goldentouchsolutions.tech

https://www.linkedin.com/in/shreyatasugandhi/

What is Java?

- Java is a programming language and a computing platform for application development.
- It was first released by Sun Microsystem in 1995 and later acquired by Oracle Corporation.
- It is one of the most used programming languages.

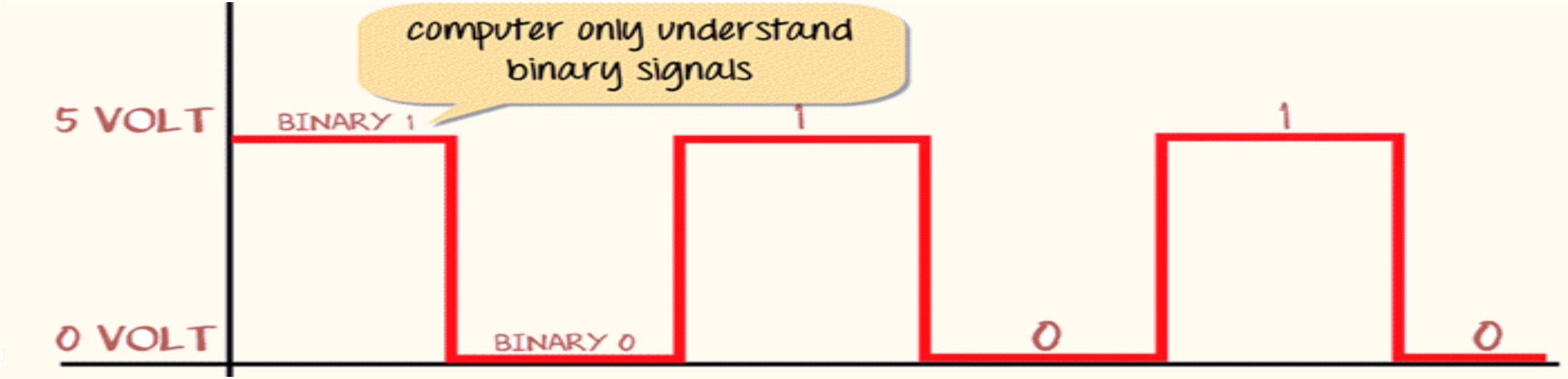
What is Java Platform?

- Java platform is a collection of programs that help to develop and run programs written in the Java programming language.
- Java platform includes an execution engine, a compiler, and a set of libraries.
- JAVA is platform-independent language. It is not specific to any processor or operating system.

What is Assembly Language?

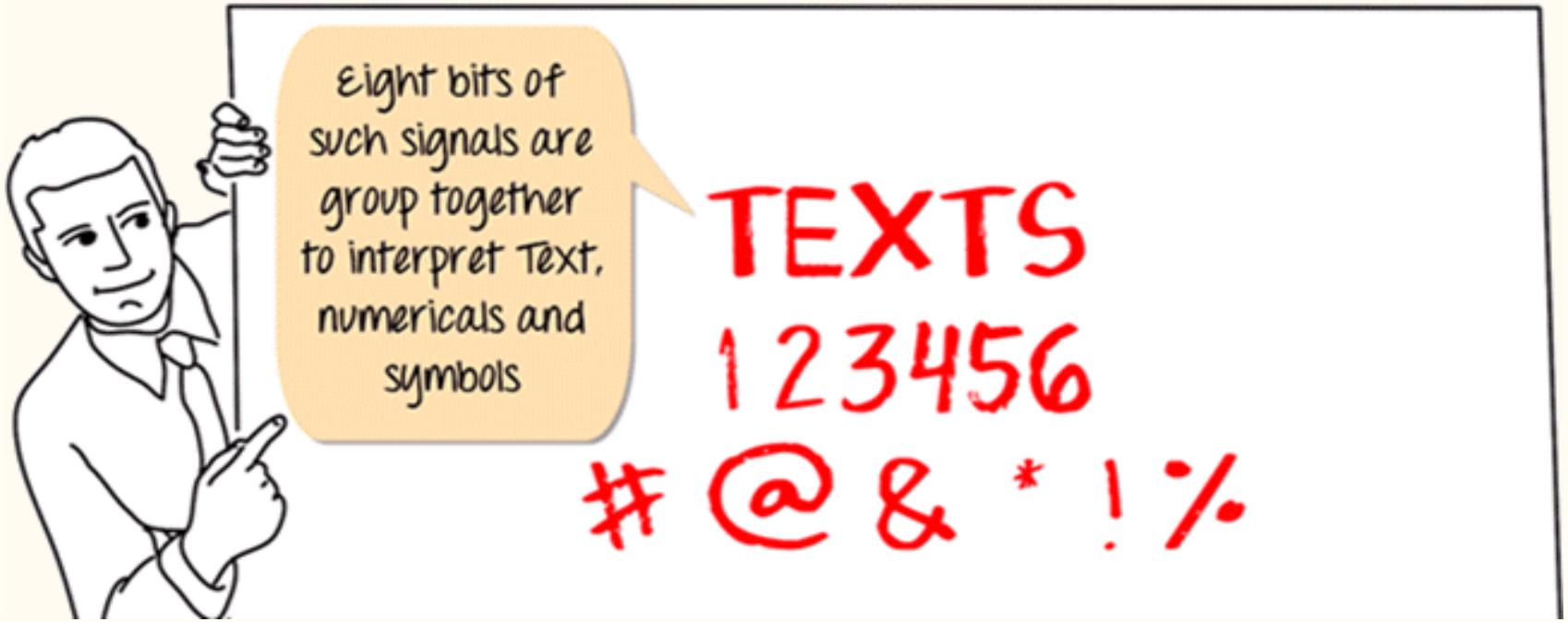
GTS EDUTECH

- Computer is an electronic device, and it can only understand electronic signals or binary signals.
  - Ex - 5-volt electronic signal may represent binary number 1 while 0 volts may represent binary number 0. So your PC is continuously bombarded with these signals.



What is Assembly Language?

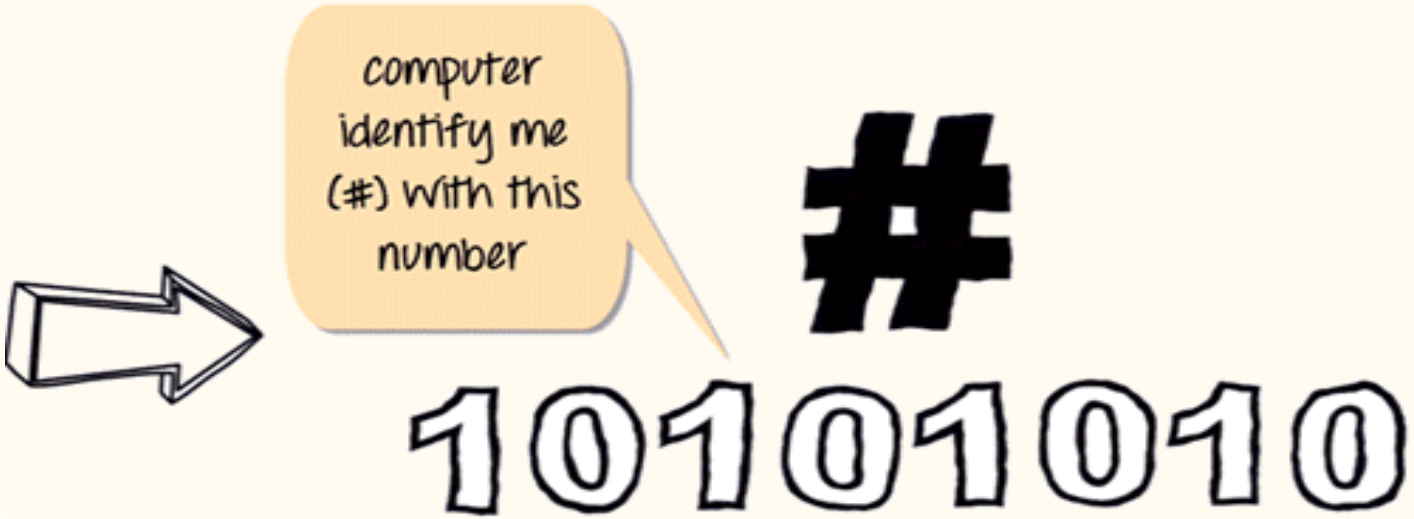
- Eight bits of such signals are group together to interpret Text, numerical and symbols.



What is Assembly Language?

GTS EDUTECH

- For example, the # symbol is identified by computer as 10101010. Similarly, the pattern for adding a function is represented by 10000011.



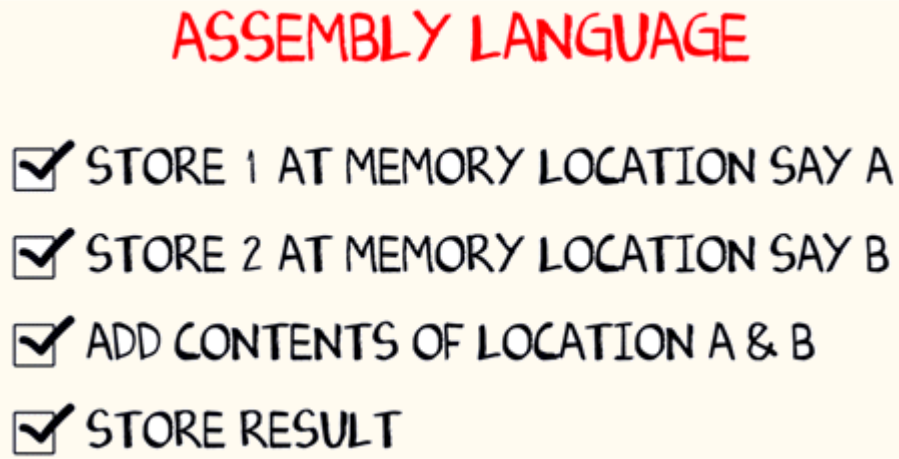
- This is known as 8-bit computing.
- Current day processor is capable of decoding 64-bit time. But what is the relation of this concept with the programming language JAVA? Let’s understand these as an example.
  - Suppose if you want to tell the computer to add two number (1+2) which is represented by some binary numbers (10000011), how are you going to tell the computer?

  Yes, we going to use assembly language to get our code executed.

  "Assembly Language is the most elementary form of software development languages."

GTS EDUTECH

- We are going to give the command to a computer in this format as shown below. Your code to add two numbers in this language would be in this order.



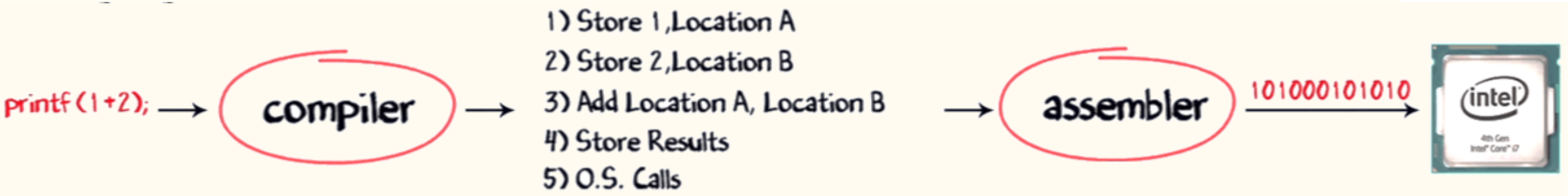
- But how are we going to do this?
  - Back in 1950's when computers were huge and consumed a great deal of power, you would convert your assembly code into corresponding machine code to 1 and 0's using mapping sheets.
  - Later this code will be punched into the machine cards and feed to the computer.
  - The computer will read these code and execute the program. These would be a long process then until ASSEMBLER came to help.

What are Assembler and Compiler?

GTS EDUTECH

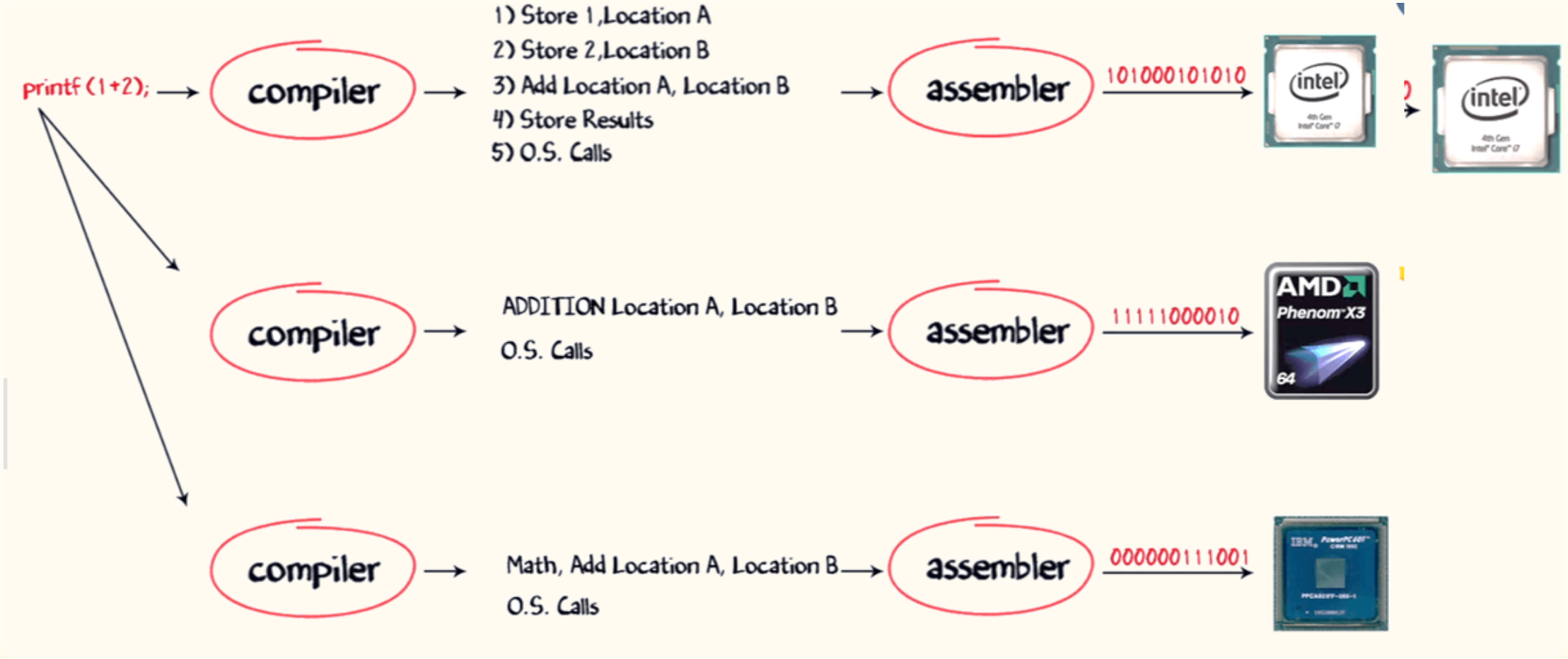
- With the advancement in technology i/o devices were invented, you could directly type your program into the PC using a program called ASSEMBLER.
- It converts it into corresponding machine code (110001..) and feeds to your processor. So coming back to our example addition of (1+2), the assembler will convert this code into machine code and give the output.

- That apart, you will also have to make calls to create Operating System provided functions to display the output of the code.
- But alone assembler is not involved in this whole process; it also requires the compiler to compile the long code into a small chunk of codes.
- With advancement in software development languages, this entire assembly code could shrink into just one line print f 1+2 A with the help of software called COMPILER.
- It is used to convert your c language code into assembly code, and the assembler converts it into corresponding machine code, and this machine code will be transmitted to the processor.
- The most common processor used in PC or Computers are Intel processor.



- Though present-day compilers come bundled with assembler can directly convert your higher language code into machine code.
- Now, suppose Windows operating system is running on this Intel processor, a combination of Operating System plus the processor is called the PLATFORM.
- The most common platform in the world is the Windows, and Intel called the Wintel Platform. The other popular platforms are AMD and Linux, Power PC, and Mac OS X.

- Now, with a change in processor, the assembly instructions will also change. For example the
  - Add instruction in Intel may be called ADDITION for AMD OR Math, ADD for Power PC
  - And obviously, with a change in Operating System, the level and nature of O.S level calls will also change.
- As a developer, I want my software program to work on all platforms available, to maximize my revenues. So I would have to buy separate compilers which convert my print f command into the native machine code.


printf (1+2);
compiler
1) Store 1,Location A
2) Store 2,Location B
3) Add Location A, Location B
4) Store Results
5) O.S. Calls
assembler
101000101010
compiler
ADDITION Location A, Location B
O.S. Calls
assembler
11111000010
AMD
Phenom X3
64
compiler
Math, Add Location A, Location B
O.S. Calls
assembler
000000111001

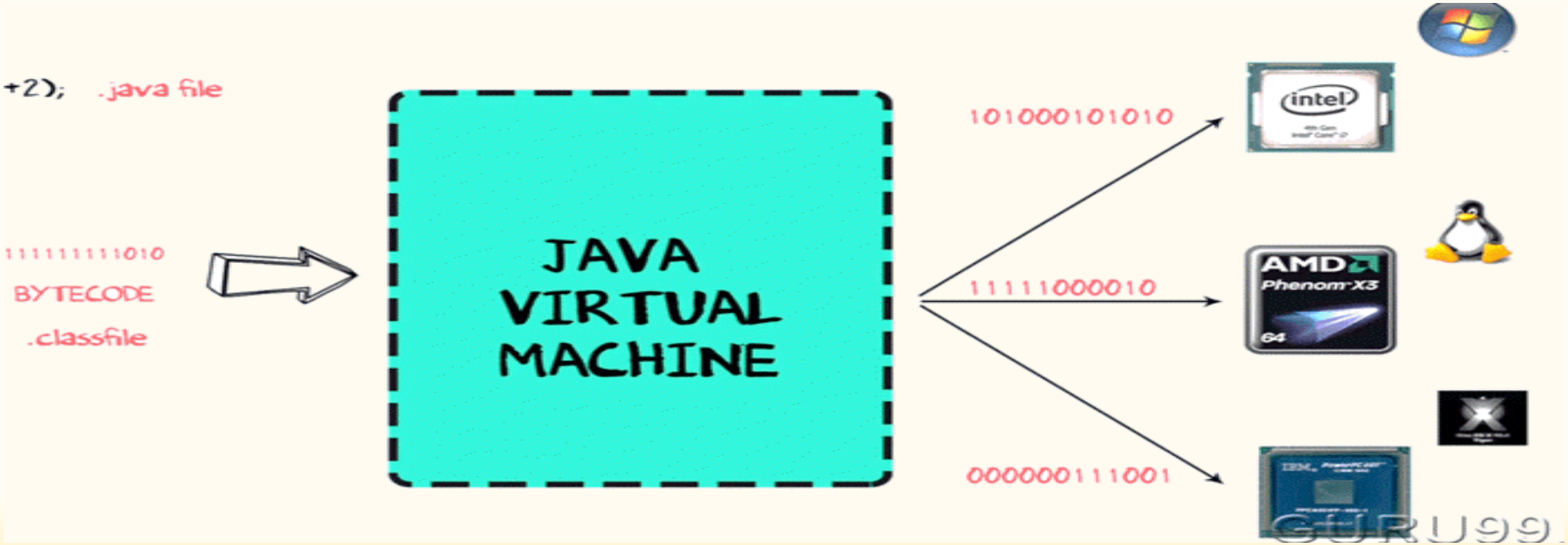
- But compilers come expensive, and there is a chance of compatibility issues.
- So buying and installing a separate compiler for different O.S and processor is not feasible. So, what can be an alternative solution?

  Enter Java language.

How Java Virtual Machine works?

GTS EDUTECH

- y using **Java Virtual Machine**, this problem can be solved. But how it works on different processors and O.S. Let's understand this process step by step.



- **Step 1)** The code to display addition of two numbers is System.out.println(1+2), and saved as .java file.
- **Step 2)** Using the java compiler the code is converted into an intermediate code called the **bytecode.** The output is a **.class file.**
- **Step 3)** This code is not understood by any platform, but only a virtual platform called the **Java Virtual Machine.**
- **Step 4)** This Virtual Machine resides in the RAM of your operating system. When the Virtual Machine is fed with this bytecode, it identifies the platform it is working on and converts the bytecode into the native machine code.
- In fact, while working on your PC or browsing the web whenever you see either of these icons be assured the java virtual machine is loaded into your RAM. But what makes java lucrative is that code once compiled can run not only on all PC platforms but also mobiles or other electronic gadgets supporting java.

- Hence,

"Java is a programming language as well as a Platform"

How is Java Platform Independent?

- Like C compiler, Java compiler does not produce native executable code for a particular machine.
- Instead, Java produces a unique format called bytecode. It executes according to the rules laid out in the virtual machine specification.
- Bytecode is understandable to any JVM installed on any OS.
- In short, the java source code can run on all operating systems.

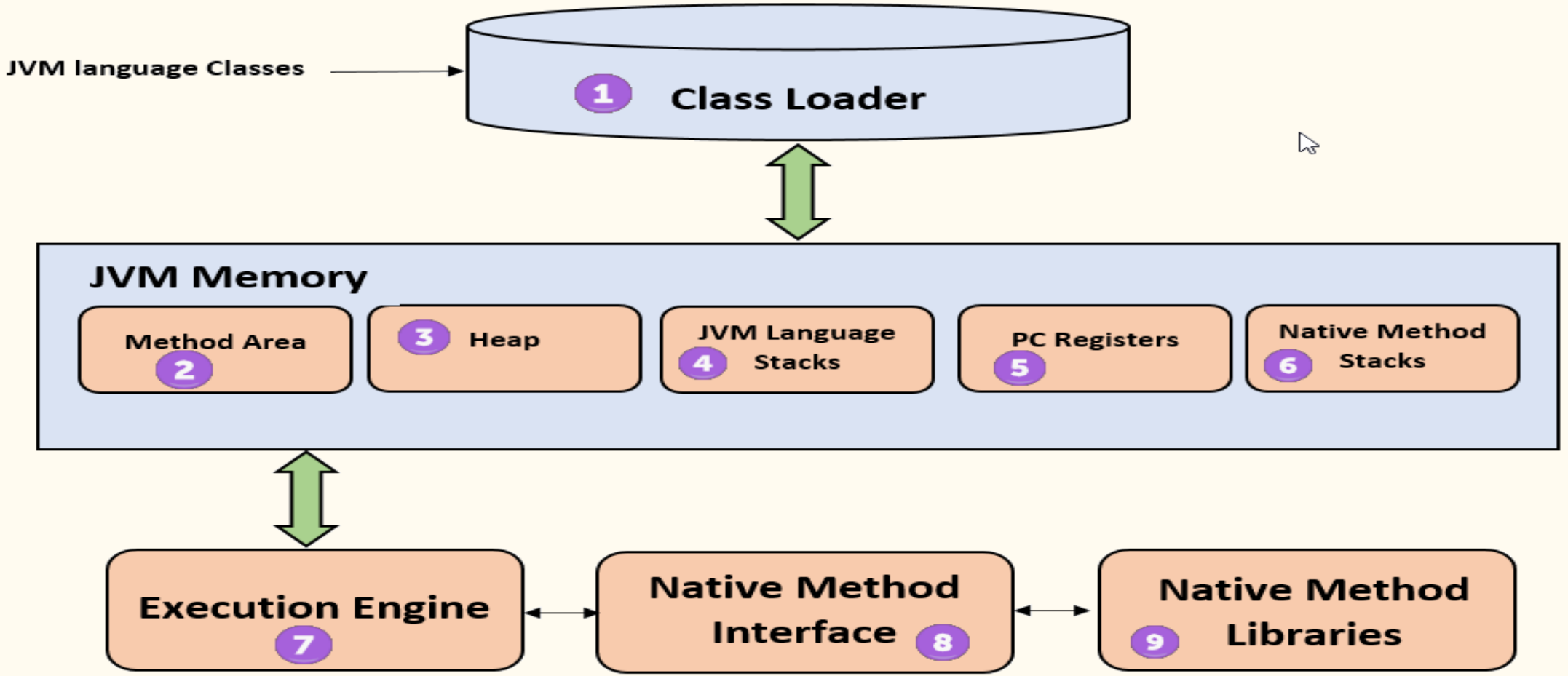
What is JVM?

- VM is a engine that provides runtime environment to drive the Java Code or applications.
- It converts Java bytecode into machines language. JVM is a part of JRE(Java Run Environment).
- It stands for **Java Virtual Machine**

- In other programming languages, the compiler produces machine code for a particular system. However, Java compiler produces code for a Virtual Machine known as Java Virtual Machine.
- First, Java code is complied into bytecode. This bytecode gets interpreted on different machines
- Between host system and Java source, Bytecode is an intermediary language.
- JVM is responsible for allocating memory space.

JVM Architecture

- Architecture of JVM contains classloader, memory area, execution engine etc.



1) ClassLoader

The class loader is a subsystem used for loading class files. It performs three major functions viz. Loading, Linking, and Initialization.

2) Method Area

JVM Method Area stores class structures like metadata, the constant runtime pool, and the code for methods.

3) Heap

All the Objects, their related instance variables, and arrays are stored in the heap. This memory is common and shared across multiple threads.

4) JVM language Stacks

Java language Stacks store local variables, and it's partial results. Each thread has its own JVM stack, created simultaneously as the thread is created. A new frame is created whenever a method is invoked, and it is deleted when method invocation process is complete.

5) PC Registers

PC register store the address of the Java virtual machine instruction which is currently executing. In Java, each thread has its separate PC register.

6) Native Method Stacks

Native method stacks hold the instruction of native code depends on the native library. It is written in another language instead of Java.

7) Execution Engine

It is a type of software used to test hardware, software, or complete systems. The test execution engine never carries any information about the tested product.

8) Native Method interface

The Native Method Interface is a programming framework. It allows Java code which is running in a JVM to call by libraries and native applications.

9) Native Method Libraries

Native Libraries is a collection of the Native Libraries(C, C++) which are needed by the Execution Engine.

Software Code Compilation & Execution process

- In order to write and execute a software program, you need the following

  1) Editor – To type your program into, a notepad could be used for this

  2) Compiler – To convert your high language program into native machine code

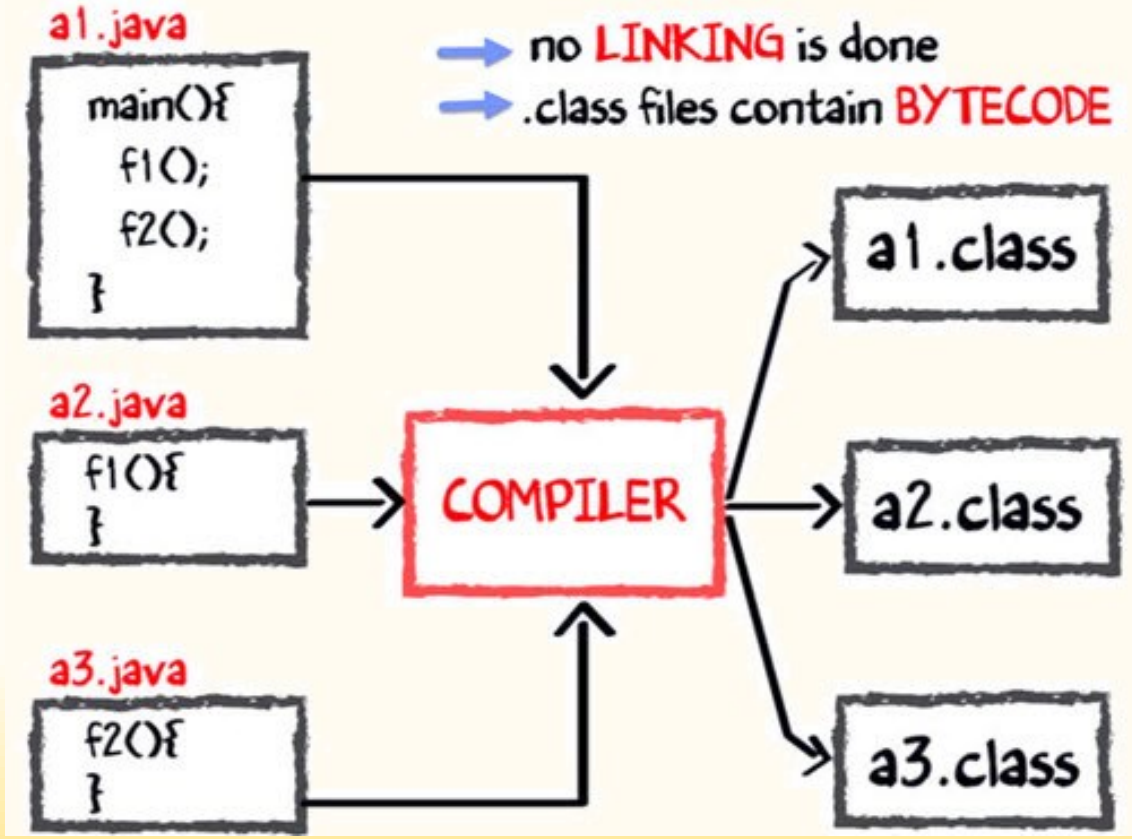
  3) Linker – To combine different program files reference in your main program together.

  4) Loader – To load the files from your secondary storage device like Hard Disk, Flash Drive, CD into RAM for execution. The loading is automatically done when you execute your code.

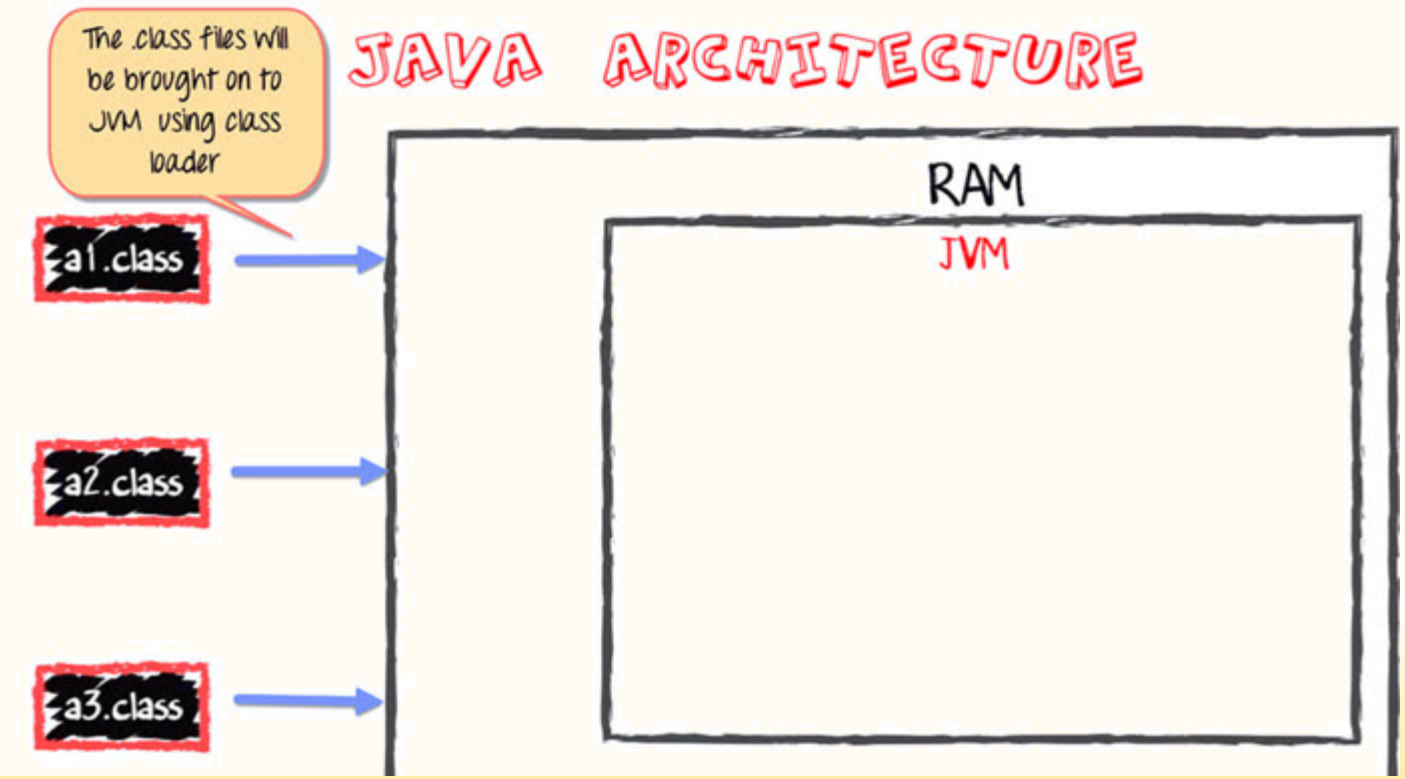
  5) Execution – Actual execution of the code which is handled by your OS & processor.

Java code Compilation and Execution in Java VM

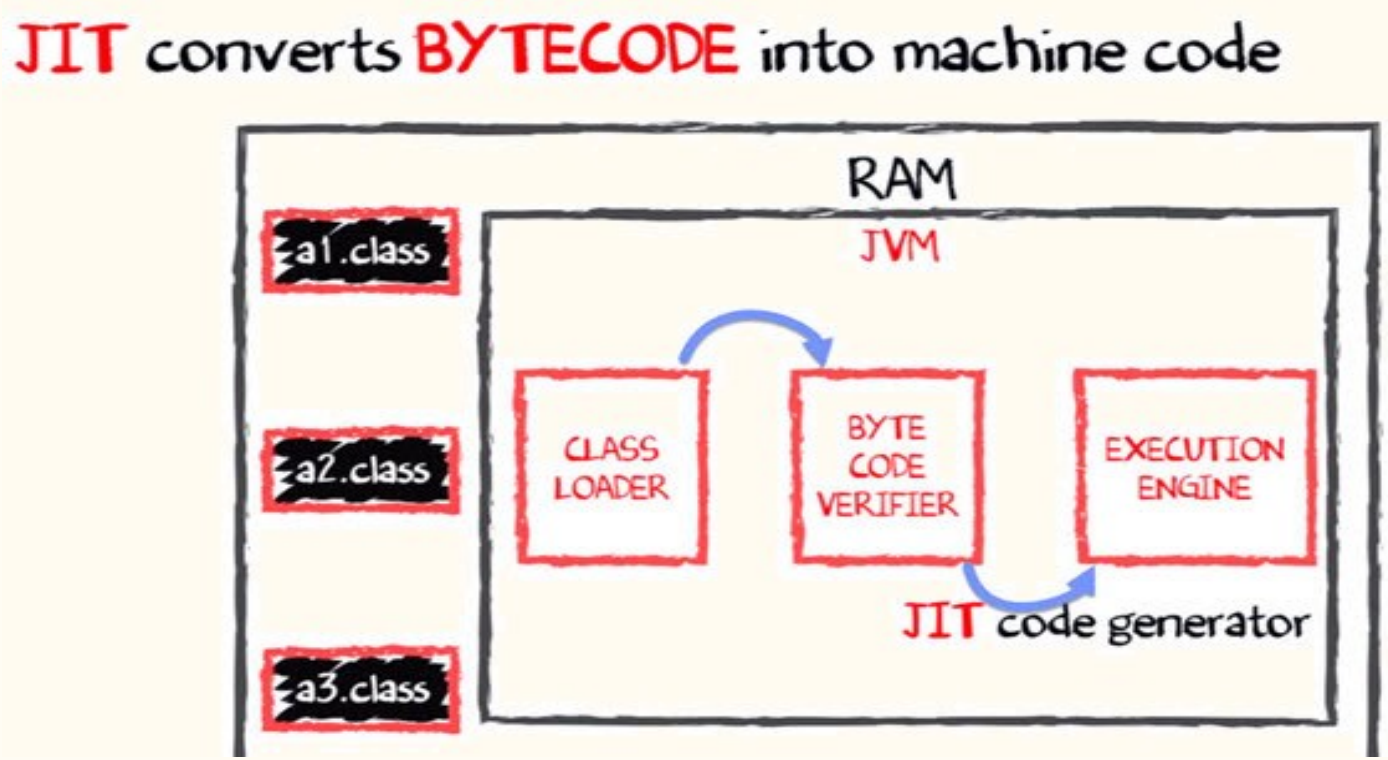
- Let's look at the process for JAVA. In your main, you have two methods f1 and f2.
  - The main method is stored in file a1.java
  - f1 is stored in a file as a2.java
  - f2 is stored in a file as a3.java



- The compiler will compile the three files and produces 3 corresponding .class file which consists of BYTE code. **Unlike C, no linking is done**.
- The Java VM or Java Virtual Machine resides on the RAM. During execution, using the class loader the class files are brought on the RAM. The BYTE code is verified for any security breaches.



- Next, the execution engine will convert the Bytecode into Native machine code. This is just in time compiling. It is one of the main reason why Java is comparatively slow.



NOTE: JIT or Just-in-time compiler is the part of the Java Virtual Machine (JVM). It interprets part of the Byte Code that has similar functionality at the same time.

Why is Java both Interpreted and Compiled Language?

- Programming languages are classified as
  - Higher Level Language Ex. C++, Java
  - Middle-Level Languages Ex. C
  - Low-Level Language Ex Assembly
  - finally the lowest level as the Machine Language.
- A **compiler** is a program which converts a program from one level of language to another. Example conversion of C++ program into machine code.
- The java compiler converts high-level java code into bytecode (which is also a type of machine code).
- An **interpreter** is a program which converts a program at one level to another programming language at the **same level.** Example conversion of Java program into C++
- In Java, the Just In Time Code generator converts the bytecode into the native machine code which are at the same programming levels.
- Hence, Java is both compiled as well as interpreted language.

Why is Java slow?

- The two main reasons behind the slowness of Java are
  - **Dynamic Linking:** Unlike C, linking is done at run-time, every time the program is run in Java.
  - **Run-time Interpreter:** The conversion of byte code into native machine code is done at run-time in Java which furthers slows down the speed

- However, the latest version of Java has addressed the performance bottlenecks to a great extent.

Installation

- Install Java Development Kit(JDK) from
  https://www.oracle.com/technetwork/java/javase/downloads/index.html

- Set JAVA_HOME & JRE_HOME in Windows -
  - set JAVA_HOME = C:\Program Files\Java\jdk1.8.0_131
  - set JRE_HOME = C:\Program Files\Java\jre1.8.0_131

- Set Environment Variables in Java: Path and Classpath in Windows -
  - PATH = %JAVA_HOME%\bin;%JRE_HOME%\bin;%PATH%;%JAVA_HOME%\lib\tools.jar;

- Install Eclipse IDE

What is OOPS?

- OOPS stands for "**Object Oriented Programming** Language and **Systems**"
- It works on a concept that **Objects** are the most important part of your program
- It allows to create object as needed and create methods to handle them.
- Manipulating these objects to get results is the goal of Object Oriented Programming.
- Java uses OOPS for programming

Core OOPS concepts

1. Class
2. Object
3. **Inheritance**
4. **Polymorphism**
5. **Abstraction**
6. **Encapsulation**
7. Association
8. Aggregation
9. Composition

Class

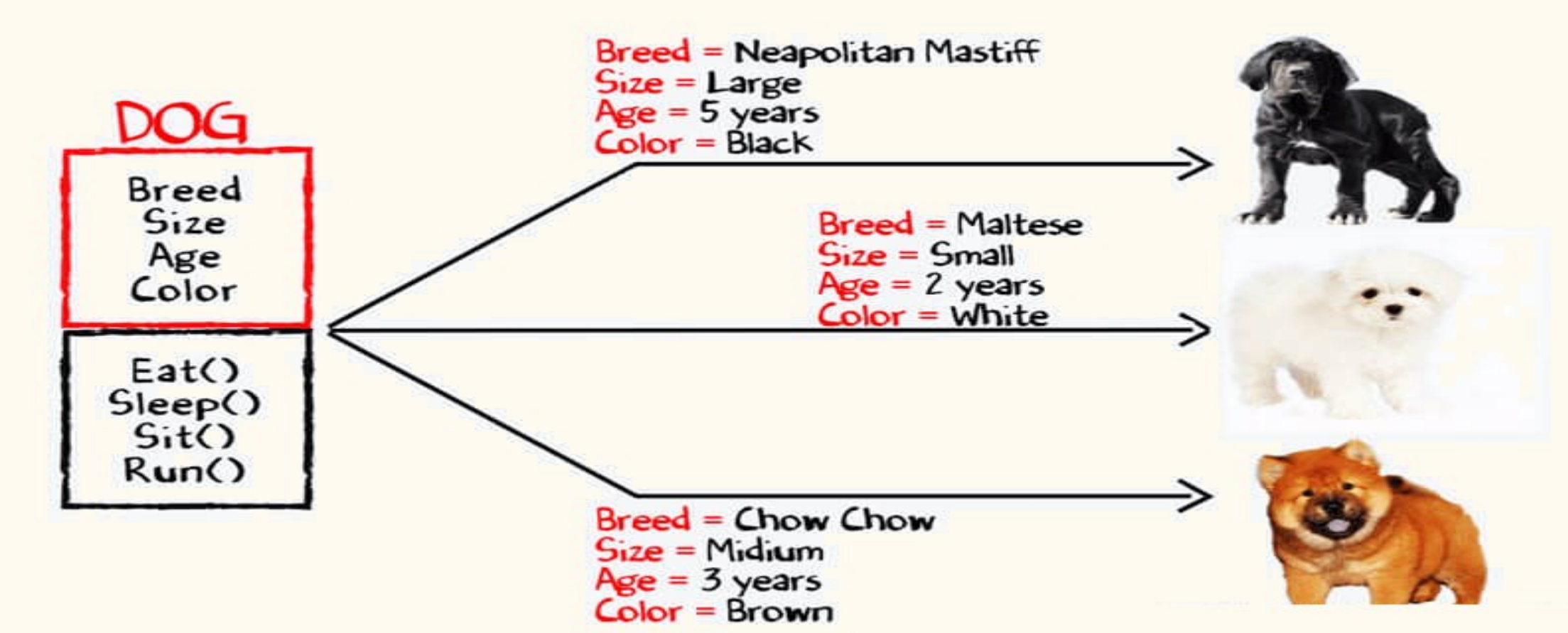
- The class is a group of similar entities
- It is only an logical component and not the physical entity.
- It can have methods

  For example, if you had a class called “Expensive Cars” it could have objects like Mercedes, BMW, Toyota, etc

  Its properties(data) can be price or speed of these cars.

  While the methods may be performed with these cars are driving, reverse, braking etc.

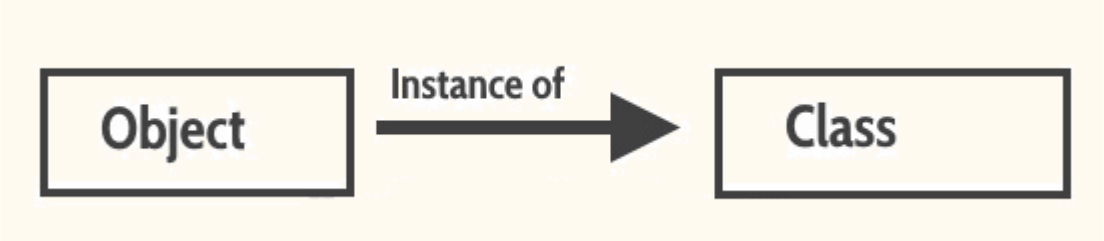
Class, Methods & Data



Object

- An object can be defined as an instance of a class, and there can be multiple instances of a class in a program.
- An Object contains both the data and the function, which operates on the data.
- Objects have two characteristics: They have **states** and **behaviors**.

  For example - chair, bike, marker, pen, table, car, etc.



Inheritance

- Inheritance is an OOPS concept in which one object acquires the properties and behaviors of the parent object.
- It’s creating a parent-child relationship between two classes.
- It offers robust and natural mechanism for organizing and structure of any software.

```
class Super {
   .....
   .....
}
class Sub extends Super {
   .....
   .....
}
```

Polymorphism

- Polymorphism refers to the ability of a variable, object or function to take on multiple forms.
- The most common use of polymorphism in OOP occurs when a parent class reference is used to refer to a child class object.

  For example, in English, the verb “run” has a different meaning if you use it with “a laptop,” “a foot race, and ”business.

  Here, we understand the meaning of “run” based on the other words used along with it.

  The same also applied to Polymorphism.

Abstraction

- An abstraction is an act of representing essential features without including background details.
- It is a technique of creating a new data type that is suited for a specific application.

  For example, while driving a car, you do not have to be concerned with its internal working.

  Here you just need to concern about parts like steering wheel, Gears, accelerator, etc.

Encapsulation

- Encapsulation is an OOP technique of wrapping the data and code.
- In this OOPS concept, the variables of a class are always hidden from other classes.
- They are declared as **‘private’**
- It can only be accessed using the methods of their current class.

  For example - in school, a student cannot exist without a class.

Association

- Association is a relationship between two objects.
- It defines the diversity between objects.
- In this OOP concept, all object have their separate lifecycle, and there is no owner.

  For example, many students can associate with one teacher while one student can also associate with multiple teachers.

Aggregation

- In this technique, all objects have their separate lifecycle.
- However, there is ownership such that child object can’t belong to another parent object.

  For example - consider class/objects department and teacher.

  Here, a single teacher can’t belong to multiple departments

  But even if we delete the department, the teacher object will never be destroyed

Composition

- A composition is a specialized form of Aggregation.
- It is also called "death" relationship.
- Child objects do not have their lifecycle, when parent object is deleted. All child object will also delete automatically along with the Parent.

  For that, let’s take an example of House and rooms.

  Any house can have several rooms.

  One room can’t become part of two different houses.

  So, if you delete the house room will also be deleted.

Advantages of OOPS

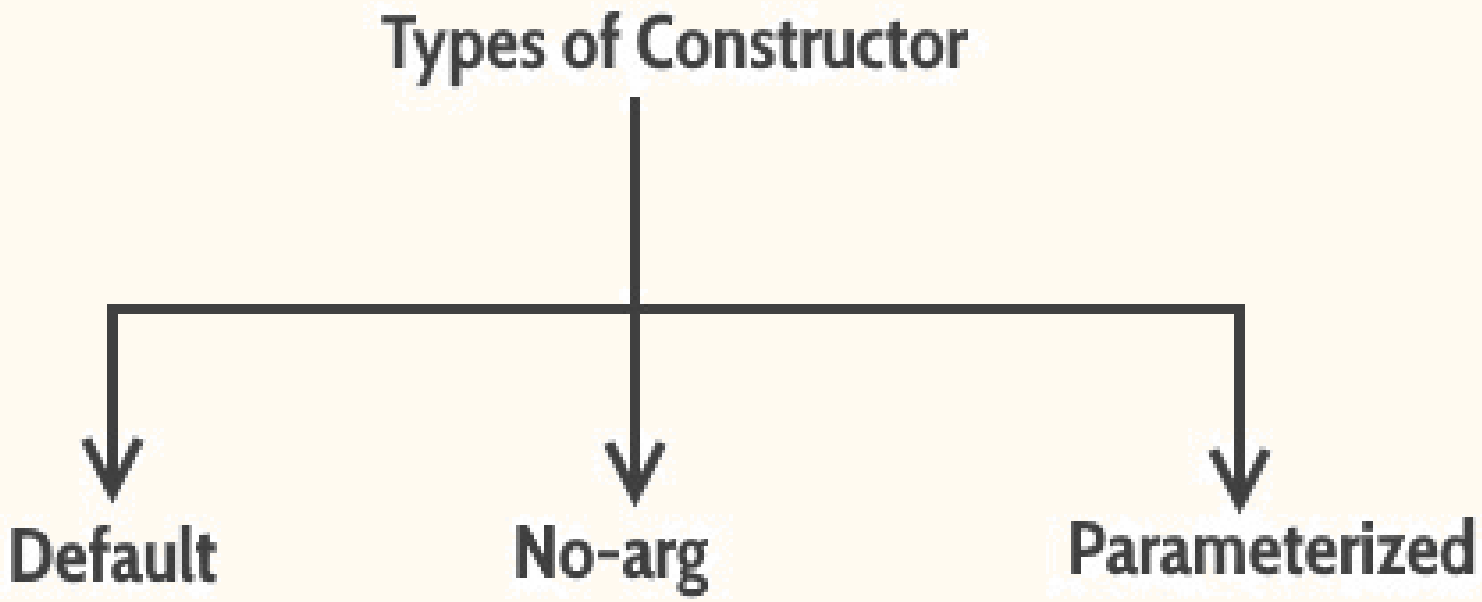
- OOP offers easy to understand and a clear modular structure for programs.
- Objects created for Object-Oriented Programs can be reused in other programs. Thus it saves significant development cost.
- Large programs are difficult to write, but if the development and designing team follow OOPS concept then they can better design with minimum flaws.
- It also enhances program modularity because every object exists independently.

Constructors

- Constructor is a block of code that initializes the newly created object.
- A constructor resembles an instance method in java but it’s not a method as it doesn’t have a return type.
- In short constructor and method are different.
- People often refer constructor as special type of method in Java.
- Constructor has same name as the class and looks like this in a java code.

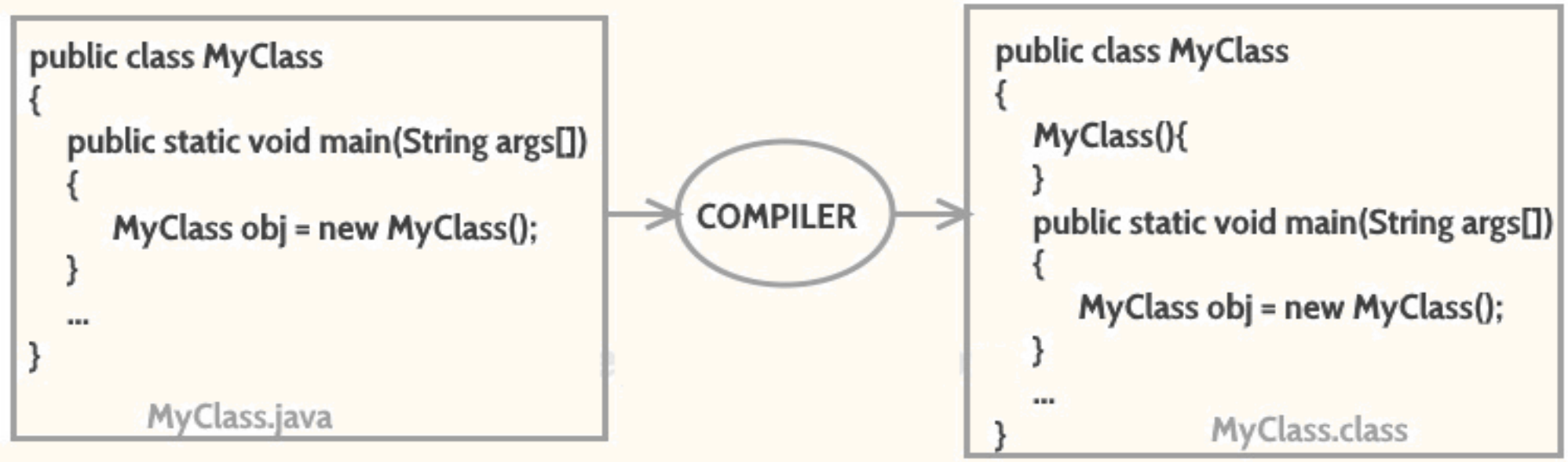
```
public class MyClass{
  //This is the constructor
  MyClass(){
  }
  ..
}
```

Types of Constructor



Default constructor

- If you do not implement any constructor in your class, Java compiler inserts a default constructor into your code on your behalf.
- You would not find it in your source code(the '.java' file) as it would be inserted into the code during compilation and exists in '.class' file.



No argument Constructors

- Constructor with no arguments is known as **no-arg constructor**.
- The signature is same as default constructor, however body can have any code unlike default constructor.

```
class Demo
{
    public Demo()
    {
       System.out.println("This is a no argument constructor");
    }
    public static void main(String args[]) {
                    new Demo();
    }
}
```

Output:

This is a no argument constructor

Parameterized constructor

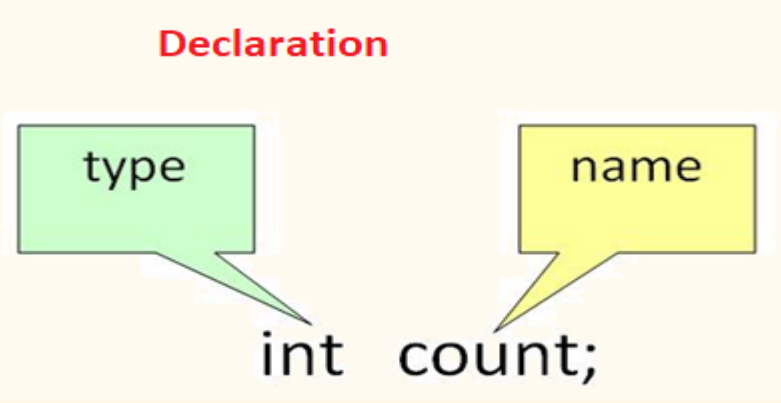
- Constructor with arguments(or you can say parameters) is known as Parameterized constructor.

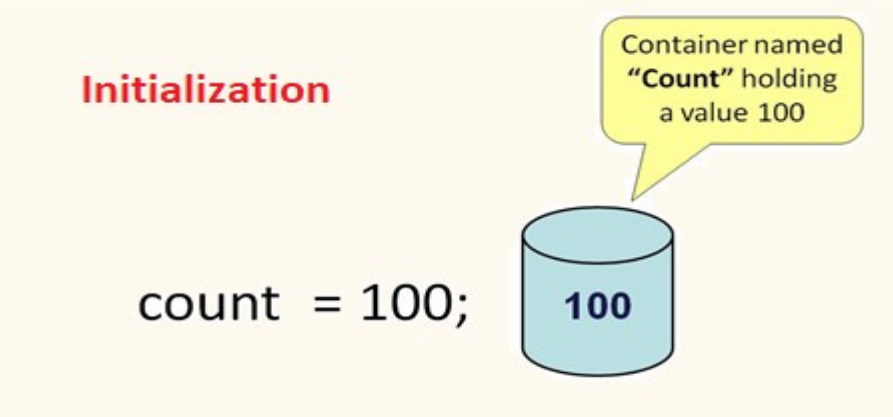
```
// A simple constructor.
class MyClass {
  int x;
  // Following is the constructor
  MyClass(int i ) {
    x = i;
  }
}

public class ConsDemo {
  public static void main(String args[]) {
    MyClass t1 = new MyClass( 10 );
    MyClass t2 = new MyClass( 20 );
    System.out.println(t1.x + " " + t2.x);
  }
}
```

Output --- 10 20

What is a Variable?

- A variable is a container which holds value for you, during the life of a Java program.
- Every variable is assigned a **data type** which designates the type and quantity of value it can hold.
- In order to use a variable in a program you to need to perform 2 steps
  - Variable Declaration
  - Variable Initialization





Types of variables

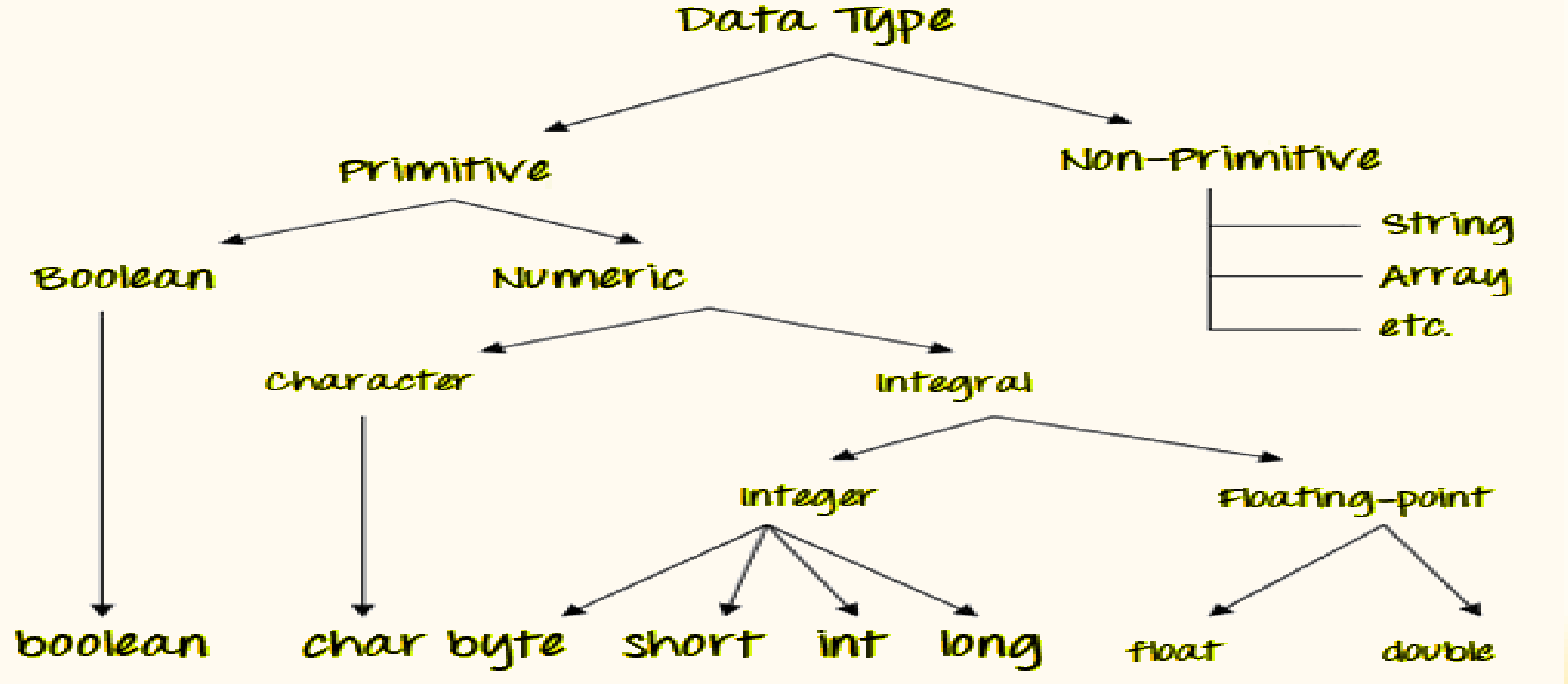
- In Java, there are three types of variables:

1. **Local Variables -** Local Variables are a variable that are declared inside the body of a method.
2. **Instance Variables -** Instance variables are defined without the STATIC keyword. They are defined Outside a method declaration. They are Object specific and are known as instance variables.
3. **Static Variables -** Static variables are initialized only once, at the start of the program execution. These variables should be initialized first, before the initialization of any instance variables.

```
class Variables {
   int data = 99; //instance variable
   static int a = 1; //static variable
   void method() {
      int b = 90; //local variable
   }
}
```

Data Types In Java

GTS EDUTECH



Data Types in Java

GTS EDUTECH

Data Type	Default Value	Default size
byte	0	1 byte
short	0	2 bytes
int	0	4 bytes
long	0L	8 bytes
float	0.0f	4 bytes
double	0.0d	8 bytes
boolean	false	1 bit
char	'\u0000'	2 bytes

Data Types in Java

Points to Remember:

- All numeric data types are signed(+/-).
- The size of data types remain the same on all platforms (standardized)
- char data type in Java is 2 bytes because it uses **UNICODE** character set. By virtue of it, Java supports internationalization. UNICODE is a character set which covers all known scripts and language in the world

Type Casting

- A variable of one type can receive the value of another type. Here there are 2 cases –

  Case 1 - Variable of smaller capacity is be assigned to another variable of bigger capacity.

  ```
  double d;
  int i = 10;
  d = i;
  ```

  This process is Automatic, and non-explicit is known as ***Conversion***

  Case 2 - Variable of larger capacity is be assigned to another variable of smaller capacity

  ```
  double d = 10;
  int i;
  i = (int) d;
  ```

  In such cases, you have to explicitly specify the **type cast operator. This process is known as *Type Casting*.**

Modifiers

- Modifiers are keywords that you add to those definitions to change their meanings.
- Java language has a wide variety of modifiers, including the following
  - Java Access Modifiers
  - Non Access Modifiers

Access Control Modifiers

- Java provides a number of access modifiers to set access levels for classes, variables, methods and constructors.
- The four access levels are −
  - Visible to the package, the default. No modifiers are needed.
  - Visible to the class only (private).
  - Visible to the world (public).
  - Visible to the package and all subclasses (protected).

Non-Access Modifiers

- Java provides a number of non-access modifiers to achieve many other functionality.
  - The ***static*** modifier for creating class methods and variables.
  - The ***final*** modifier for finalizing the implementations of classes, methods, and variables.
  - The ***abstract*** modifier for creating abstract classes and methods.
  - The ***synchronized*** and ***volatile*** modifiers, which are used for threads.

GTS EDUTECH