Object Oriented Programming

Dr. D. Sudheer
Assistant Professor Sr. Grade. 1
Email: sudheer.d@vitap.ac.in
Cabin. CB205C

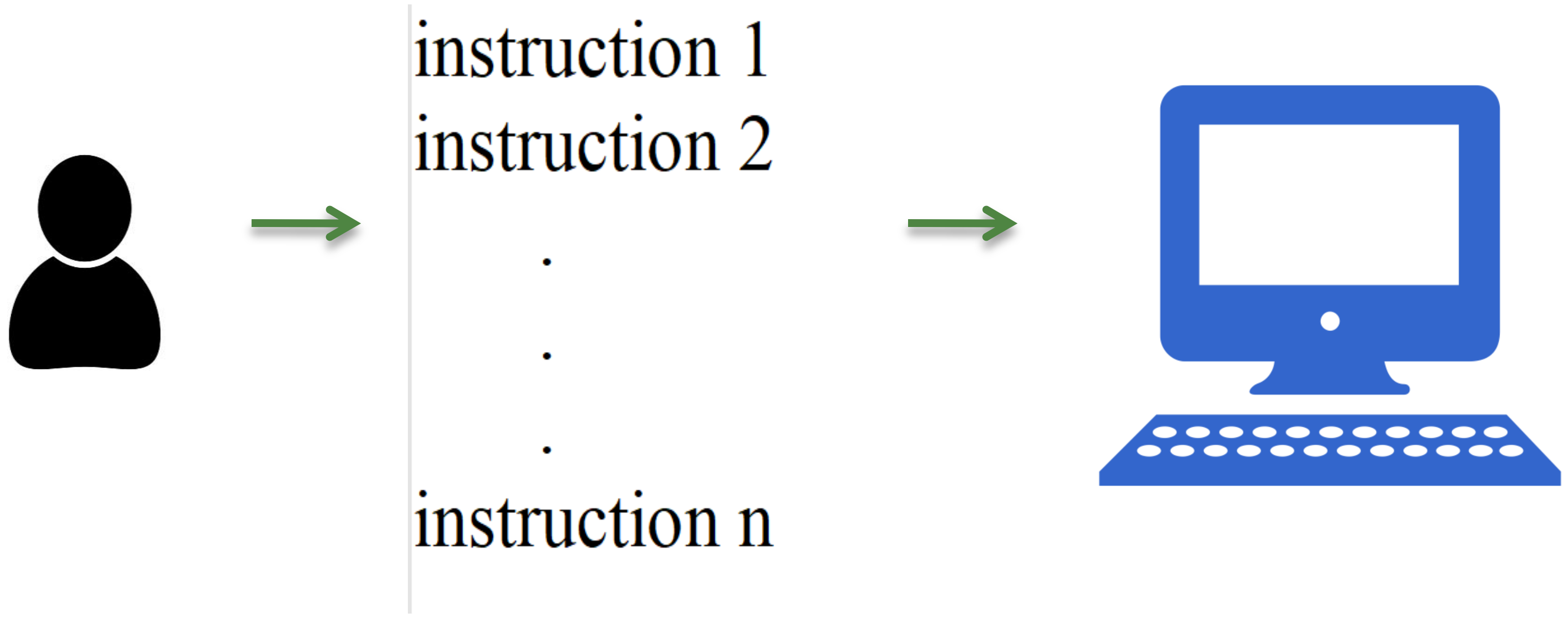
Objectives

- **To design the concepts of object-oriented, event driven, and concurrent programming paradigms and develop skills by using these paradigms in Java.**
- **To analyze, design the principals of inheritance, dynamic polymorphism and interfaces.**
- **To learn writing a computer program to solve specified problems.**
- **To enable using the Java SDK environment to create, debug and run simple applications.**

Expected Outcome

- Design the structure of the Java programming language
- Identify classes, objects, members of a class and relationships among them needed for a specific problem
- Develop applications using packages, interfaces and also database connection.
- Develop Java programs to implement error handling techniques using exception handling
- Develop applications using Object Oriented Programming principals and proper programming structure
- Develop and understand multithreaded applications with synchronization

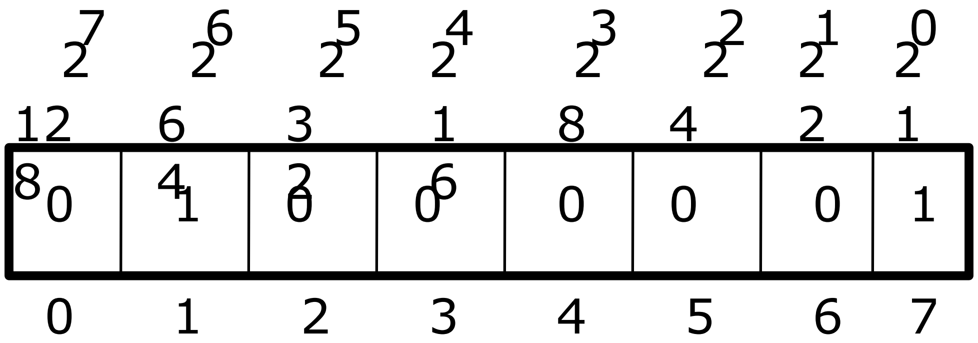
Program ...?



Binary Language

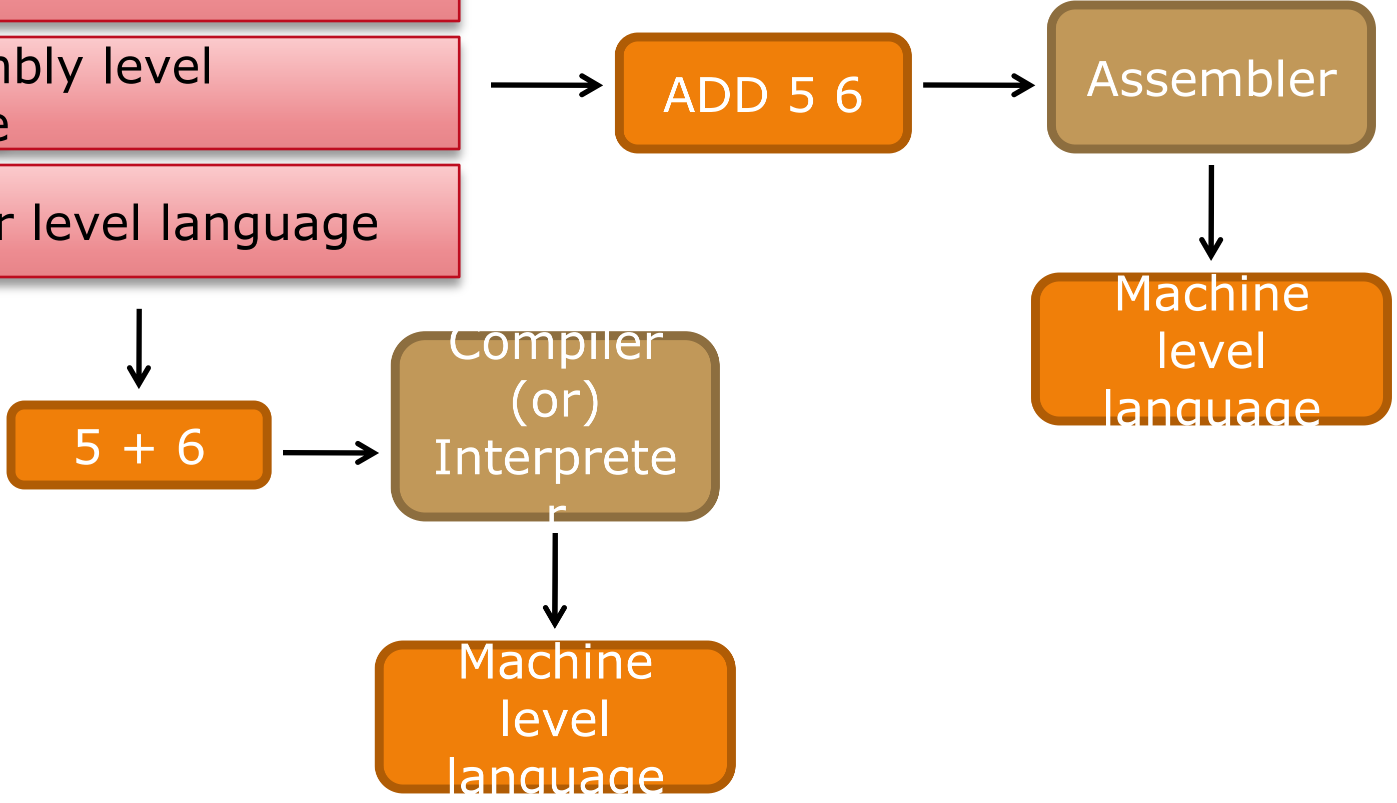
Letter	ASCII	Binary
A	065	01000001
a	097	01100001
b	098	01100010

Binary Number Representation



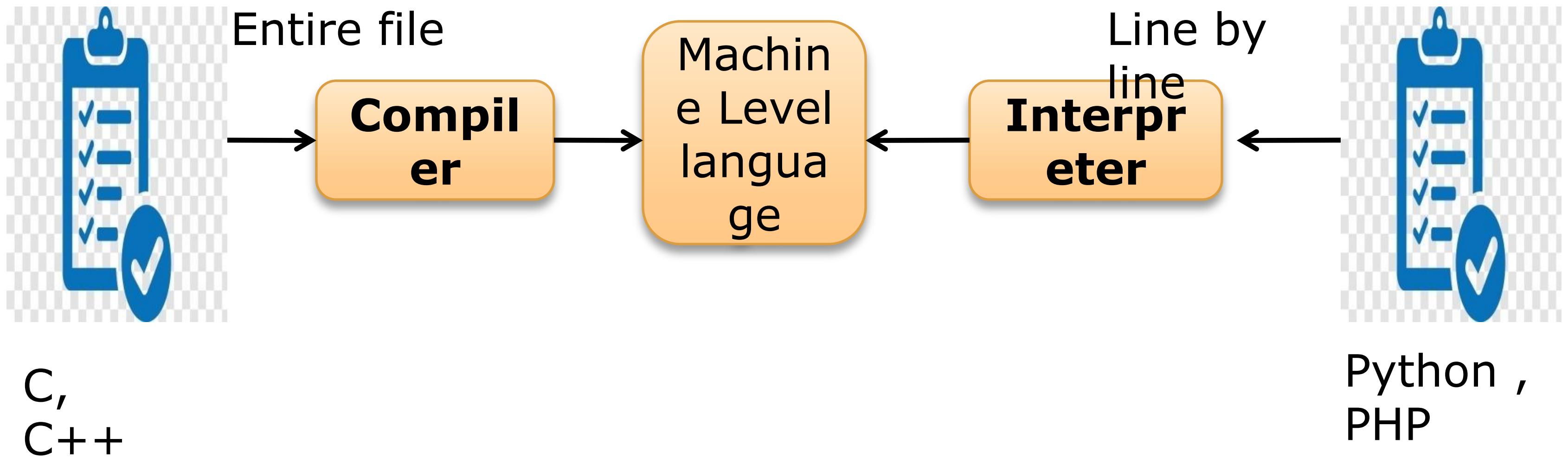
Types of programming language

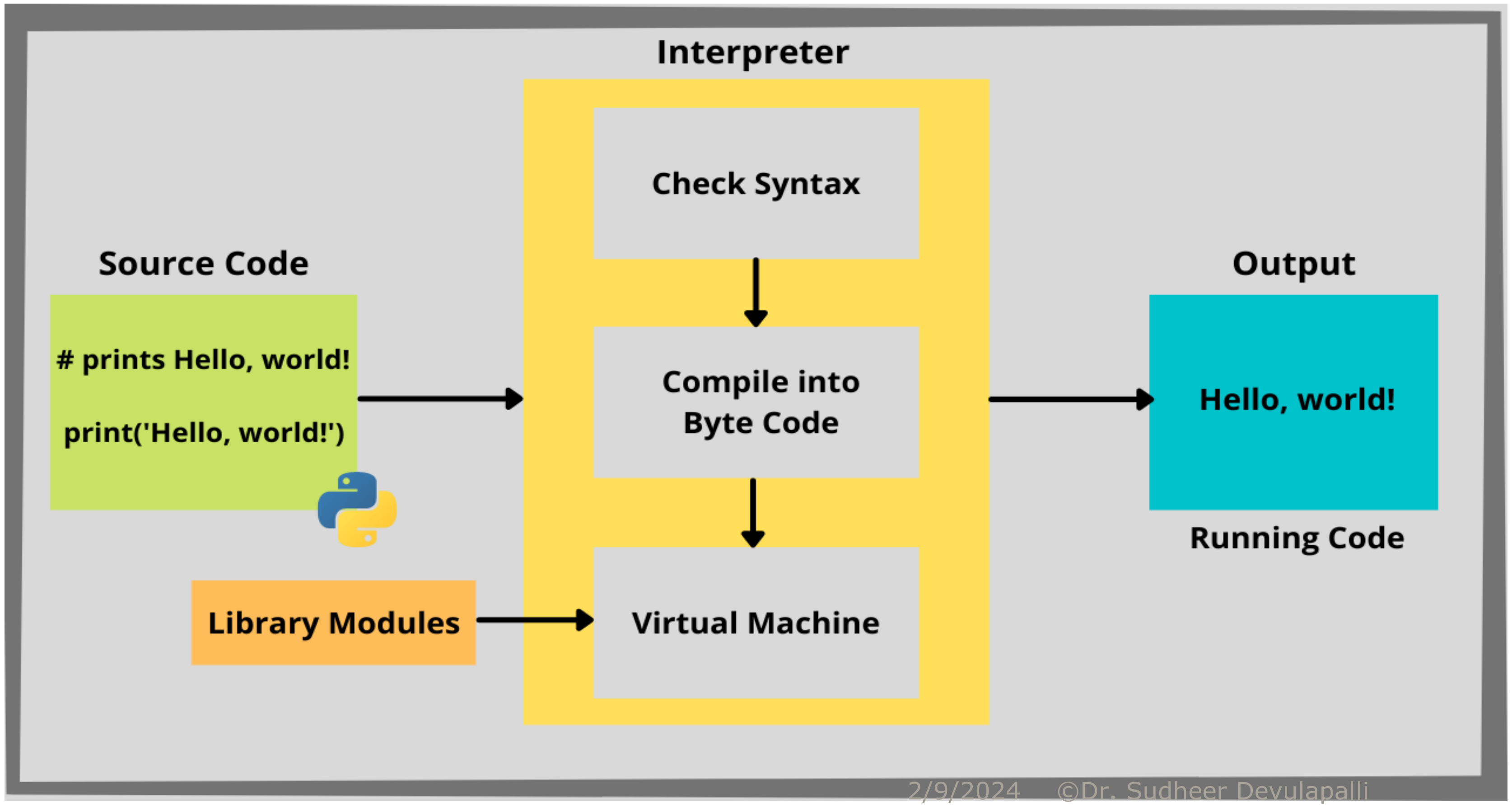
1. Machine level language
2. Assembly level language
3. Higher level language



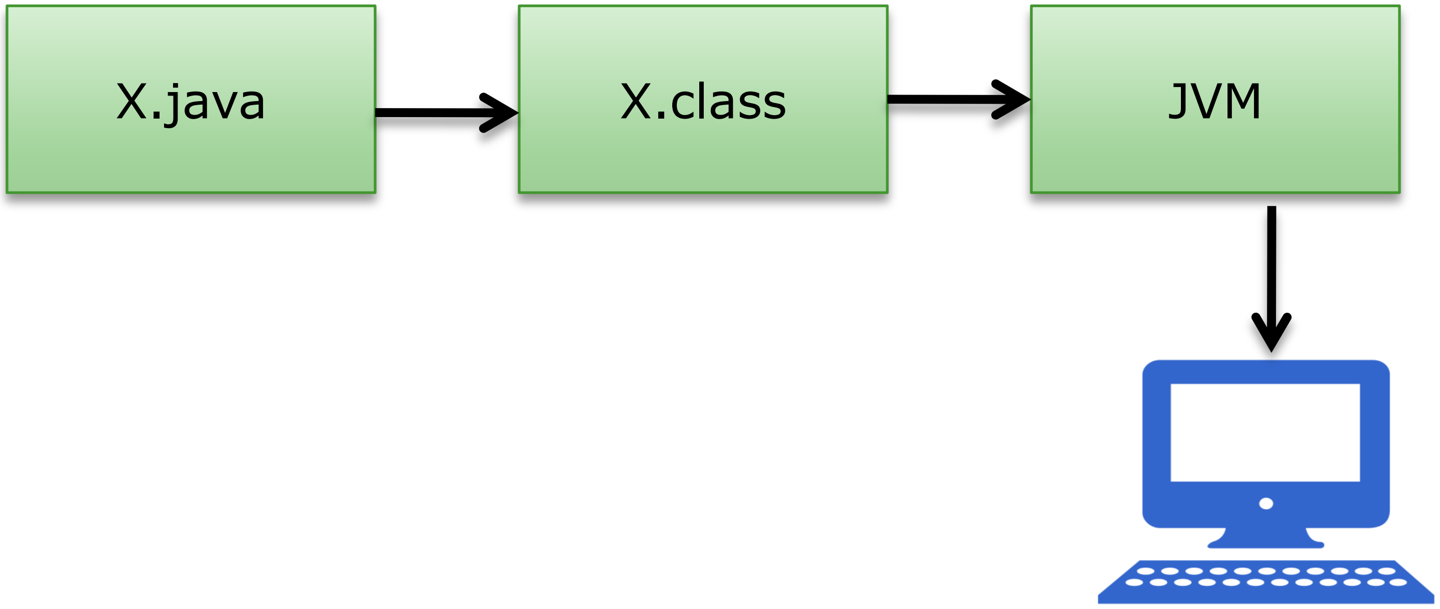
High level to machine level

Machine Level language > Assembly Level language > High Level language




Interpreter
Check Syntax
Compile into Byte Code
Virtual Machine
Source Code
# prints Hello, world!
print('Hello, world!')
Library Modules
Output
Hello, world!
Running Code

Java Execution Process



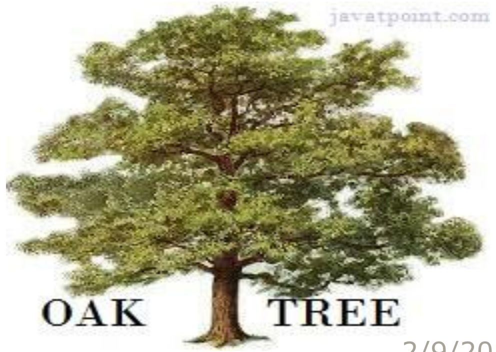
History of Java

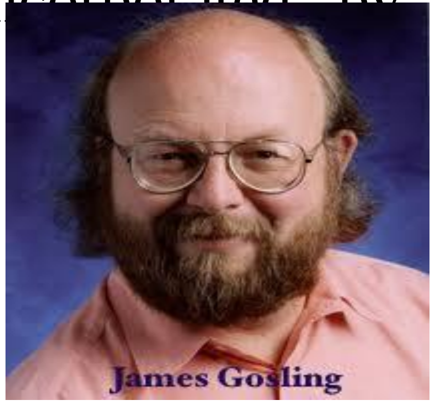
The history of Java starts with the Green Team. Java team members (also known as Green Team), initiated this project to develop a language for digital devices such as set-top boxes, televisions, etc. However, it was best suited for internet programming. Later, Java technology was incorporated by Netscape.

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

Initial Name: Oak

In 1995, Oak was renamed as **"Java"** because it was already a trademark by Oak Technologies.





Why java name?

According to James Gosling, "Java was one of the top choices. It is an island in Indonesia where the first coffee was produced (called Java coffee).

Initially developed by James Gosling at Sun Microsystems (which is now a subsidiary of Oracle Corporation) and released in 1995.

Features of java or Java Buzz Words

- Java is the most popular object-oriented programming language.
- Java has many advanced features, a list of key features is known as Java Buzz Words.

- **Simple**
- **Secure**
- **Portable**
- **Object-oriented**
- **Robust**
- **Architecture-neutral (or) Platform Independent**
- **Multi-threaded**
- **Interpreted**
- **High performance**
- **Distributed**
- **Dynamic**

Simple: Java programming language is very simple and easy to learn, understand, and code. Most of the syntaxes in java follow basic programming language C and object-oriented programming concepts are similar to C++. In a java programming language, many complicated features like pointers, operator overloading, structures, unions, etc. have been removed. One of the most useful features is the garbage collector it makes java more simple.

Secure: Java is said to be more secure programming language because it does not have pointers concept, java provides a feature "applet" which can be embedded into a web application. The applet in java does not allow access to other parts of the computer, which keeps away from harmful programs like viruses and unauthorized access.

Portable: Portability is one of the core features of java which enables the java programs to run on any computer or operating system. For example, an applet developed using java runs on a wide variety of CPUs, operating systems, and browsers connected to the Internet.

Object-oriented: Java is said to be a pure object-oriented programming language. In java, everything is an object. It supports all the features of the object-oriented programming paradigm. The primitive data types java also implemented as objects using wrapper classes, but still, it allows primitive data types to archive high-performance.

Robust: Java is more robust because the java code can be executed on a variety of environments, java has a strong memory management mechanism (garbage collector), java is a strictly typed language, it has a strong set of exception handling mechanism, and many more.

Platform Independent: Java has invented to archive "write once; run anywhere, any time, forever". The java provides JVM (Java Virtual Machine) to to archive architectural-neutral or platform-independent. The JVM allows the java program created using one operating system can be executed on any other operating system.

Multi-threaded: Java supports multi-threading programming, which allows us to write programs that do multiple operations simultaneously.

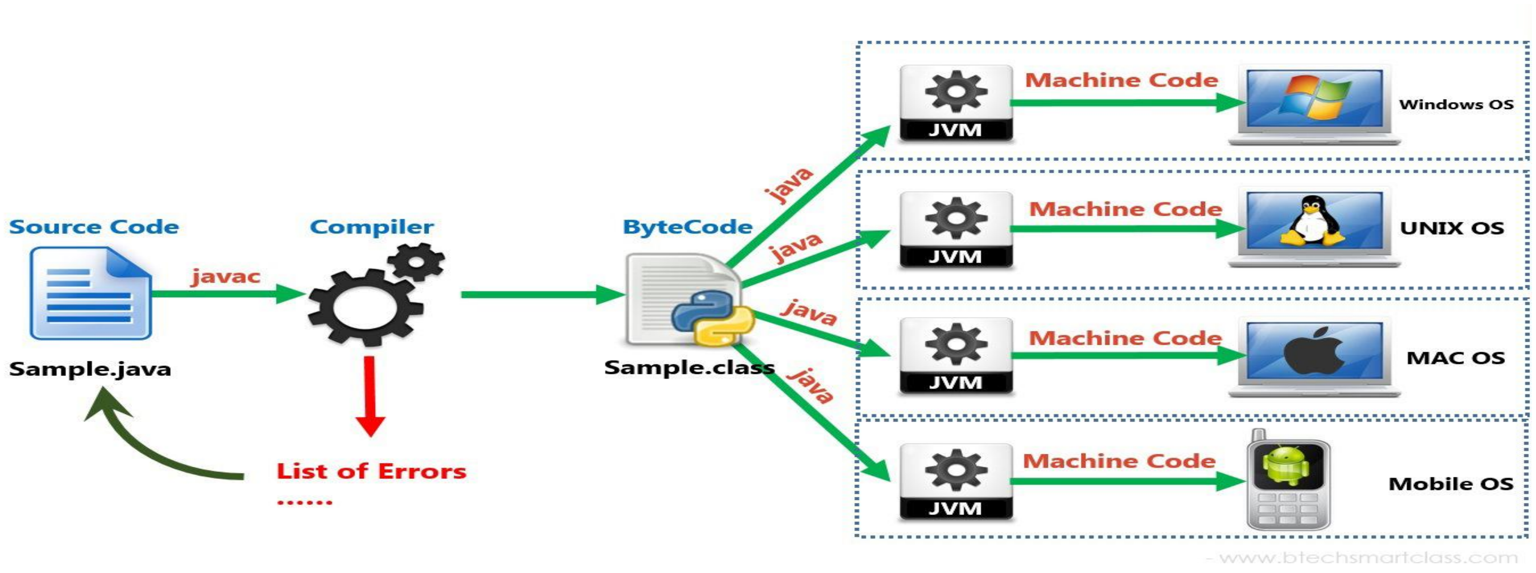
Interpreted: Java enables the creation of cross-platform programs by compiling into an intermediate representation called Java byte code. The byte code is interpreted to any machine code so that it runs on the native machine.

High performance: Java provides high performance with the help of features like JVM, interpretation, and its simplicity.

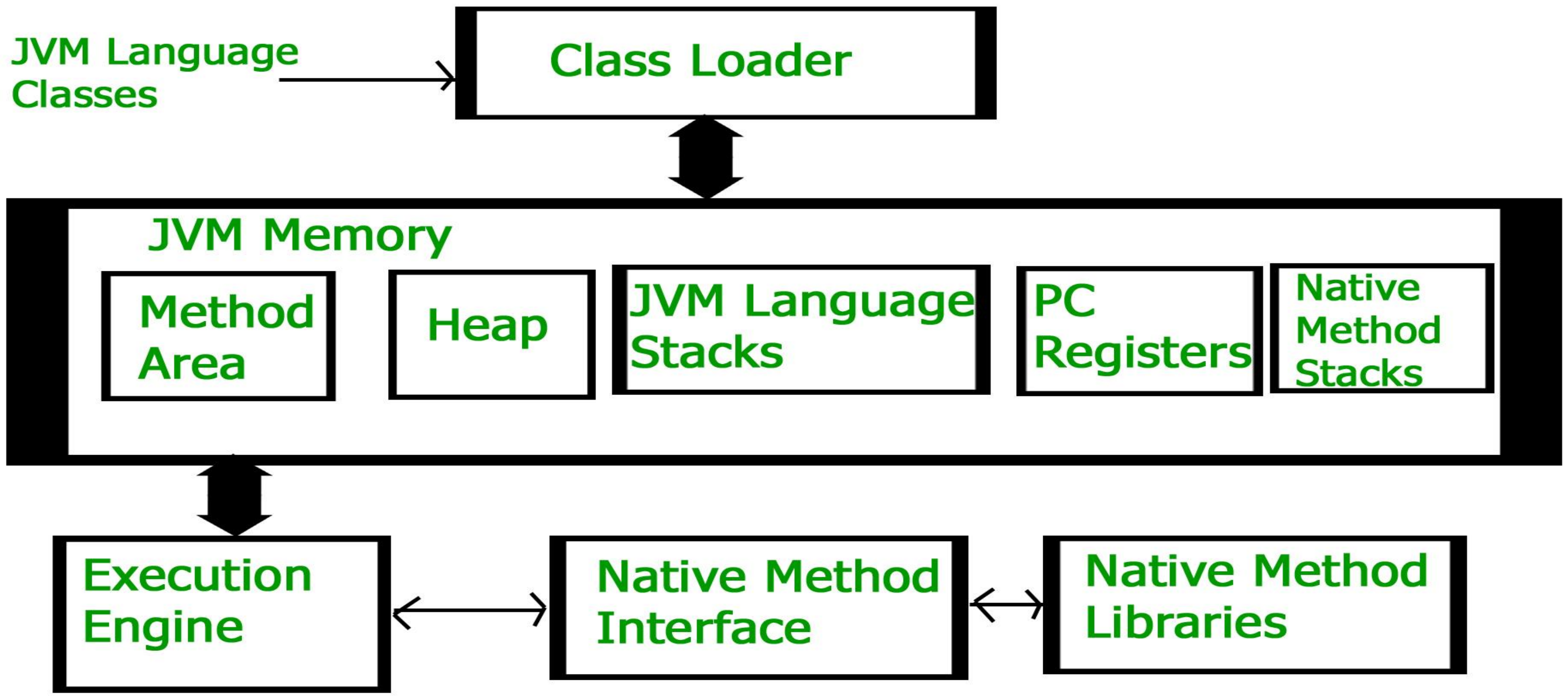
Distributed: Java programming language supports TCP/IP protocols which enable the java to support the distributed environment of the Internet. Java also supports Remote Method Invocation (RMI), this feature enables a program to invoke methods across a network.

Dynamic: Java is said to be dynamic because the java byte code may be dynamically updated on a running system and it has a dynamic memory allocation and de-allocation (objects and garbage collector).

Execution Process of Java Program



- Create a source code (.java file).
- Compile the source code using javac command.
- Run or execute .class file uisng java command.


JVM Language Classes
Class Loader
JVM Memory
Method Area
Heap
JVM Language Stacks
PC Registers
Native Method Stacks
Execution Engine
Native Method Interface
Native Method Libraries

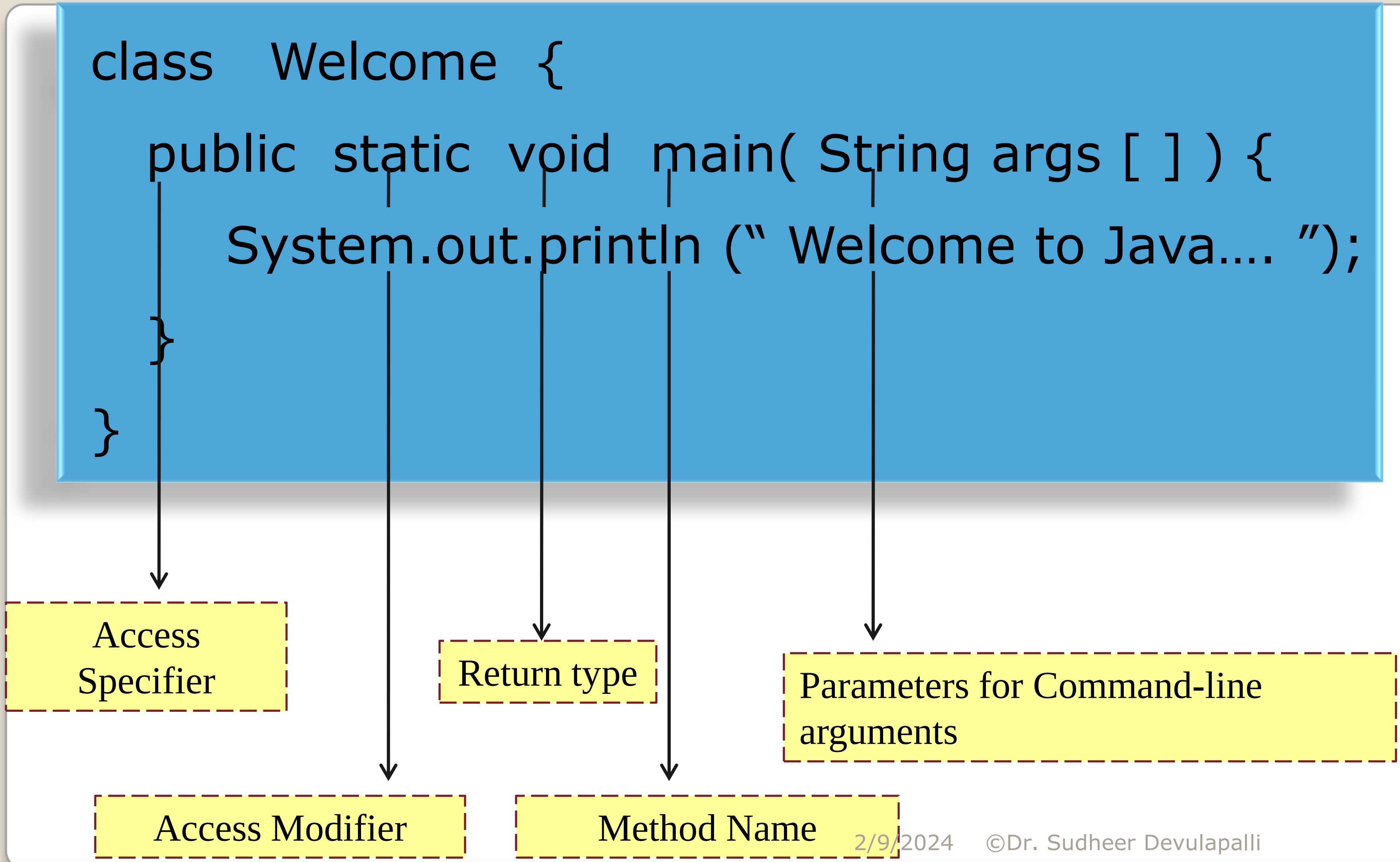
1. It loads the .class file to JVM.
2. JVM verify the byte code instructions, If it finds any instruction suspicious it will reject immediately.
3. If the byte code verification successful then it will allocates necessary memory to execute the program.

Method area: it stores class codecode of variables, code of methods.
Heap: It stores objects, when JVM loads class method and heap area creates immediately.
Java stacks: It is used to execute the methods of java.
PC registers: These are the registers, which contain memory address of the instructions of the methods.
Native method stacks: Java native methods are executed here.

Sample Program

```
class   Welcome  {
   public  static  void  main( String args [ ] ) {
       System.out.println (" Welcome to Java…. ");
   }
}
```

The class Keyword is used to declare a class. Keywords are reserved words that have a special meaning. Here, the class keyword defines the class Welcome. The braces { } known as delimiters, are used to indicate the start and end of a class body



```
class   Welcome  {
   public  static  void  main( String args [ ] ) {
       System.out.println (" Welcome to Java…. ");
   }
}
```

```
class   Welcome  {
    public  static  void  main( String args [ ] ) {
        System.out.println (" Welcome to Java…. ");
    }
}
```

An access specifier defines the scope of a class member. A class member refers to the variables and methods in a class.

Java supports the following access specifiers:

- public
- private
- protected
- default

```
class   Welcome  {
   public  static  void  main( String args [ ] ) {
       System.out.println (" Welcome to Java…. ");
   }
}
```

Here, main method is declared `public`. Any member that is declared public can be accessed from outside the class with object.

```
class   Welcome  {
   public  static  void  main( String args [ ] ) {
       System.out.println (" Welcome to Java…. ");
   }
}
```

The keyword **static** is an access modifier. Here, static keyword describes main method does not need an object to get invoked.

```
class   Welcome  {
   public  static  void  main( String args [ ] ) {
       System.out.println (" Welcome to Java…. ");
   }
}
```

The keyword `void` is a return type. It means main method does not return any thing after execution.

```
class   Welcome  {
   public  static  void  main( String args [ ] ) {
        System.out.println (" Welcome to Java…. ");
   }
}
```

The main method

The first line of code that a java compiler looks for in the source file is the main method. This function is the entry point of the application.

The `main method` is ideally used to create objects and invoke methods.

main( ) is the first method which is executed in a java program.

Naming Conventions in Java

The following naming conventions are often followed, although not enforced by Java:

- ❑ Names of ***classes*** and ***interfaces*** begin with an uppercase letter. (Pascal Case)

  Examples : Student , EmployeeData

- ❑ Names of ***methods*** and ***variables*** follows camel Case, i.e., If a name is composed of several words, then each word (except possibly the first one) begins with an uppercase letter.

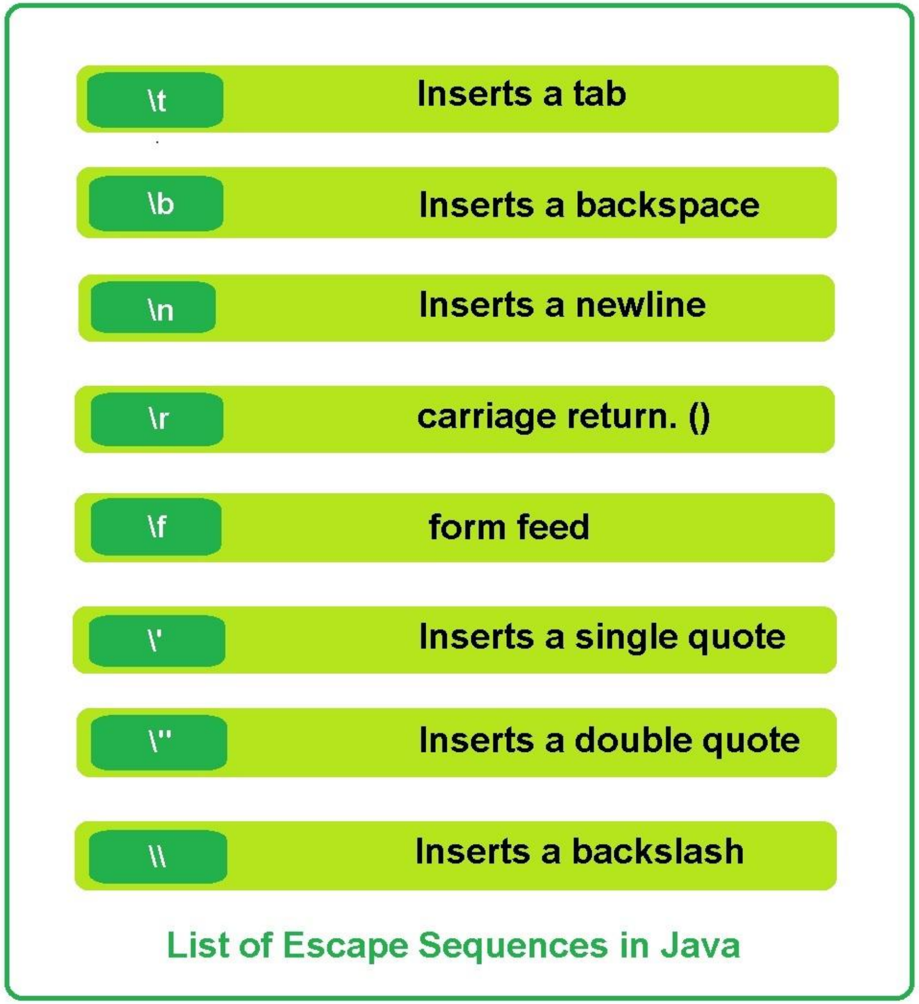
  Examples : setDetails ( ) , showResult ( )

  vehicle , myVehicle

- Named ***constants*** (that is, final variables and fields) are written entirely in uppercase, and the parts of composite names are separated by underscores (_). Examples : CENTER , MAX_VALUE.
- Names of ***packages*** written entirely in lowercase.

  Examples : java.lang , java.awt.event

  For uniqueness, they are often prefixed with reverse domain names, as in com. sun.xml.util.



\t	Inserts a tab
\b	Inserts a backspace
\n	Inserts a newline
\r	carriage return. ()
\f	form feed
\'	Inserts a single quote
\"	Inserts a double quote
\\	Inserts a backslash

List of Escape Sequences in Java