




# What is Automation Testing?

- Automation testing is the process of testing the software using an automation tools to find the defects. In this process, executing the test scripts and generating the results are performed automatically by automation tools.

## Automation testing tools


- HP QTP(Quick Test Professional)/UFT(Unified Functional Testing)
- **Selenium**
- LoadRunner
- IBM Rational Functional Tester
- SilkTest
- TestComplete
- WinRunner
- WATIR

# When to use Automation Testing?

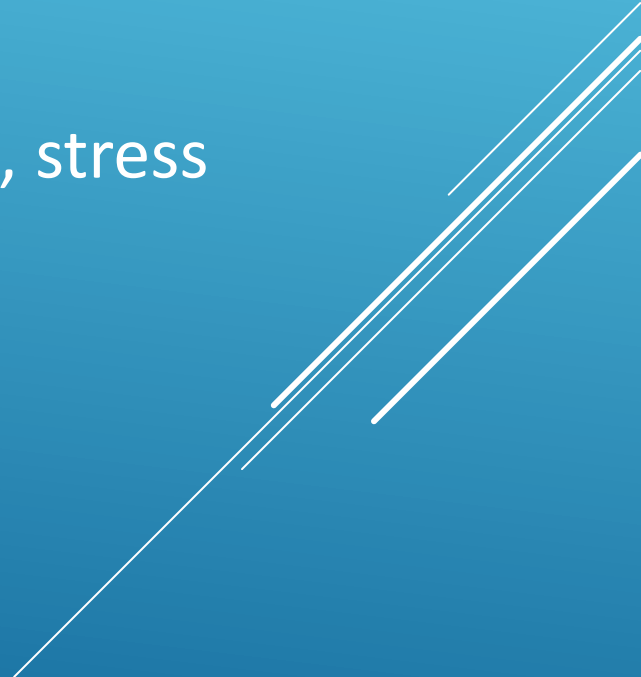
- Regression Testing:
  - Load Testing:
  - Performance Testing:
  - Integration Testing:
  - System Testing:
  - Unit Testing:
  - Acceptance Testing:
- 
- A series of three parallel white diagonal lines extending from the bottom right towards the top right of the slide.

# Which tests cannot be automated?

Tests cannot be automated. Test which take too much effort to automate are:

- Exploratory Testing
  - User interface testing
  - Adhoc Testing
- 
- A series of white diagonal lines of varying lengths and thicknesses, located in the bottom right corner of the slide.

# When do you prefer Automation Testing over Manual Testing?

- To handle repetitive and time-consuming tasks
  - When there is a need of parallel testing
  - To do non-functional testing like load, performance, stress testing
  - To avoid human errors
- 
- A series of white diagonal lines of varying lengths and thicknesses, located in the bottom right corner of the slide, creating a modern, abstract graphic element.

# Advantages of automated testing

- Automation testing is faster in execution
- It is cheaper compared to manual testing in the long run
- Automated testing is more reliable
- Automated testing is more powerful and versatile
- It is mostly used for regression testing
- It is reusable because the automation process can be recorded
- Does not require human intervention. Test scripts can be run unattended
- It helps to increase the test coverage

# Disadvantages of Automated Testing

- It is recommended only for stable products
- Automation testing is expensive initially
- Most of the automation tools are expensive
- It has some limitations such as handling captcha, getting visual aspects of UI such as fonts, color, sizes etc.,
- Huge maintenance in case of repeated changes in the requirements

Automation Testing	Manual Testing
Automated testing is more reliable. It performs same operation each time. It eliminates the risk of human errors.	Manual testing is less reliable. Due to human error, manual testing is not accurate all the time.
Initial investment of automation testing is higher. Investment is required for testing tools. In the long run it is less expensive than manual.	Initial investment of manual testing is less than automation. Investment is required for human resources.
Automation testing is a practical option when we do regressions testing.	Manual testing is a practical option where the test cases are not run repeatedly and only needs to run once or twice.
Execution is done through software tools, so it is faster than manual testing and needs less human resources compared to manual testing.	Execution of test cases is time consuming and needs more human resources
Exploratory testing is not possible	Exploratory testing is possible
Performance Testing like Load Testing, Stress Testing etc. is a practical option in automation testing.	Performance Testing is not a practical option in manual testing
It can be done in parallel and reduce test execution time.	Its not an easy task to execute test cases in parallel in manual testing. We need more human resources to do this and becomes more expensive.
Programming knowledge is a must in automation testing	Programming knowledge is not required to do manual testing.
Human intervention is not much, so it is not effective to do User Interface testing.	It involves human intervention, so it is highly effective to do User Interface testing.



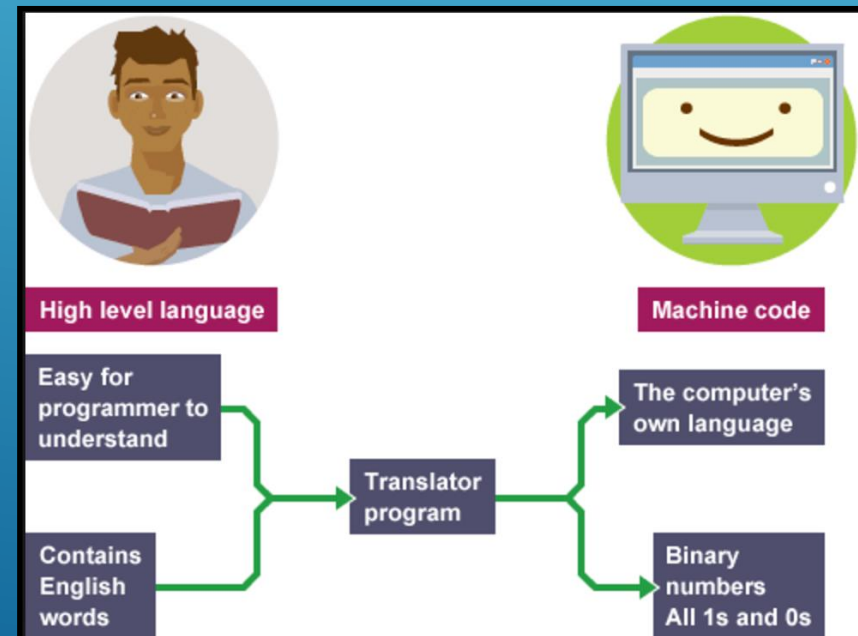
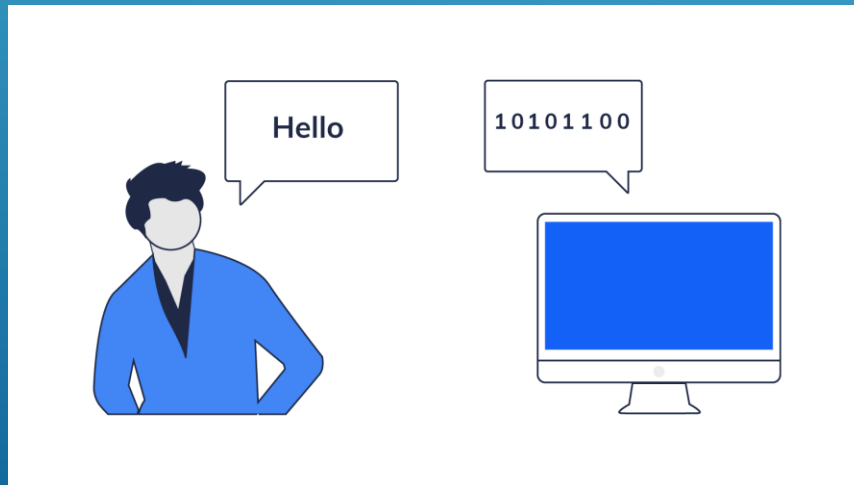
# PROGRAMMING LANGUAGE

## ➤ Program:

- A set of instructions that are to be carried out by a computer.

## ➤ Programming language:

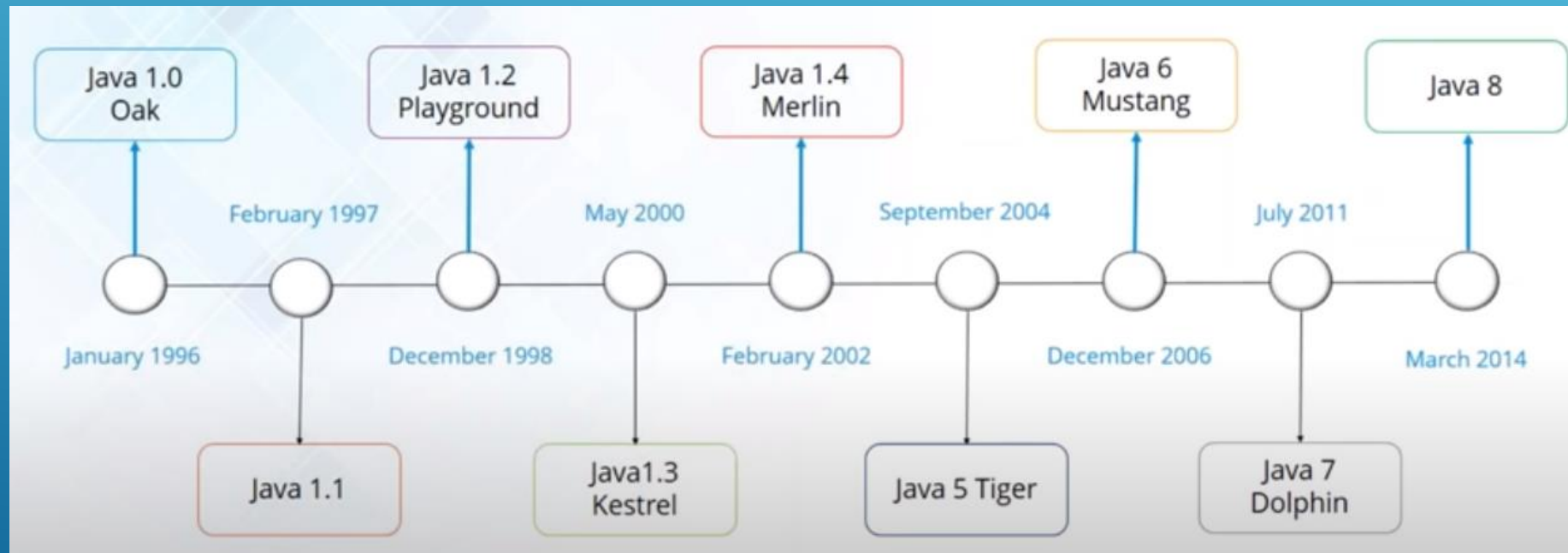
- A systematic set of rules used to describe computations, generally in a format that is editable by humans.
- Forms a bridge between humans and machines (computers).
- Programming languages consist of a set to instructions that we give just like sentences or statements we say to another human being when communicating.



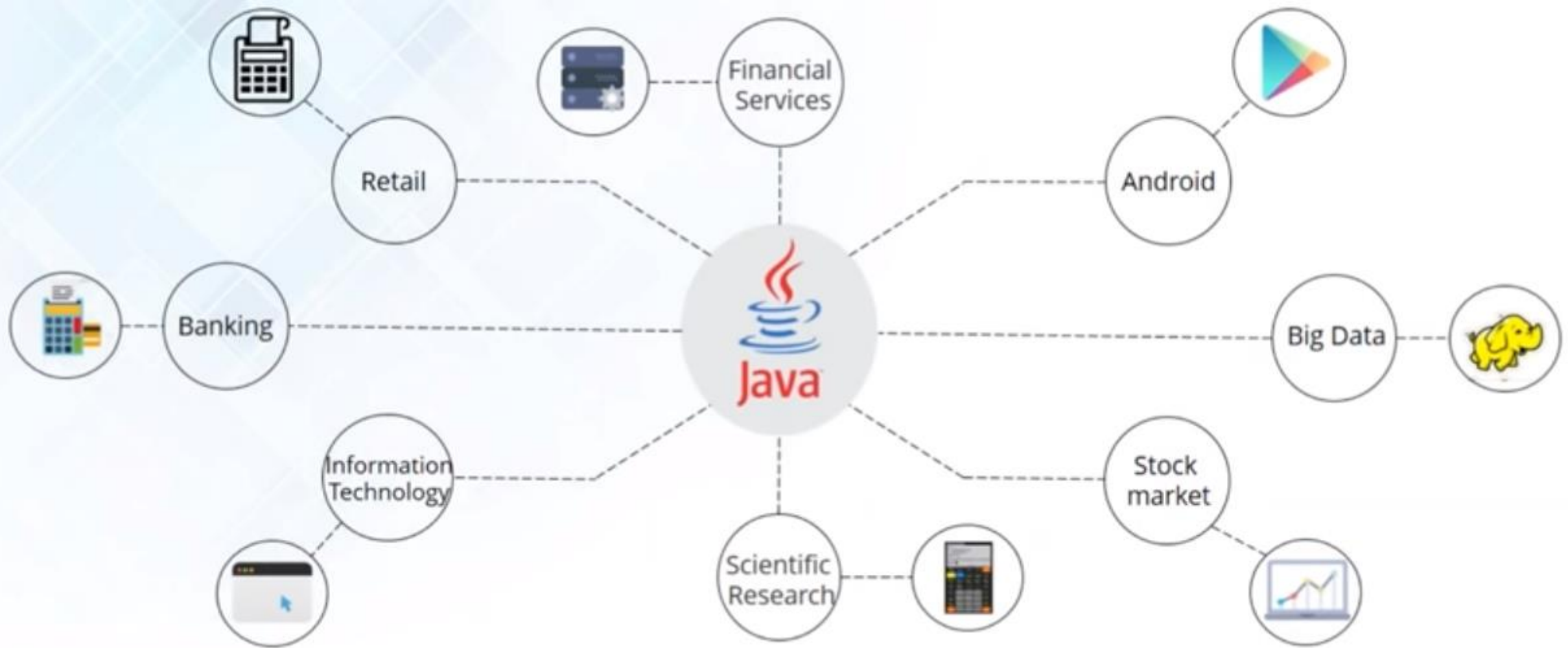
# WHAT IS JAVA?



- Java was created by a team lead by James Gosling in 1995 for Sun Microsystems.
- Java is a platform independent programming language – “Write once, Run anywhere”.



# WHERE IS JAVA USED?



# JAVA FEATURES



Simple



Portable



Object-oriented



Secure



Distributed



Dynamic



Robust



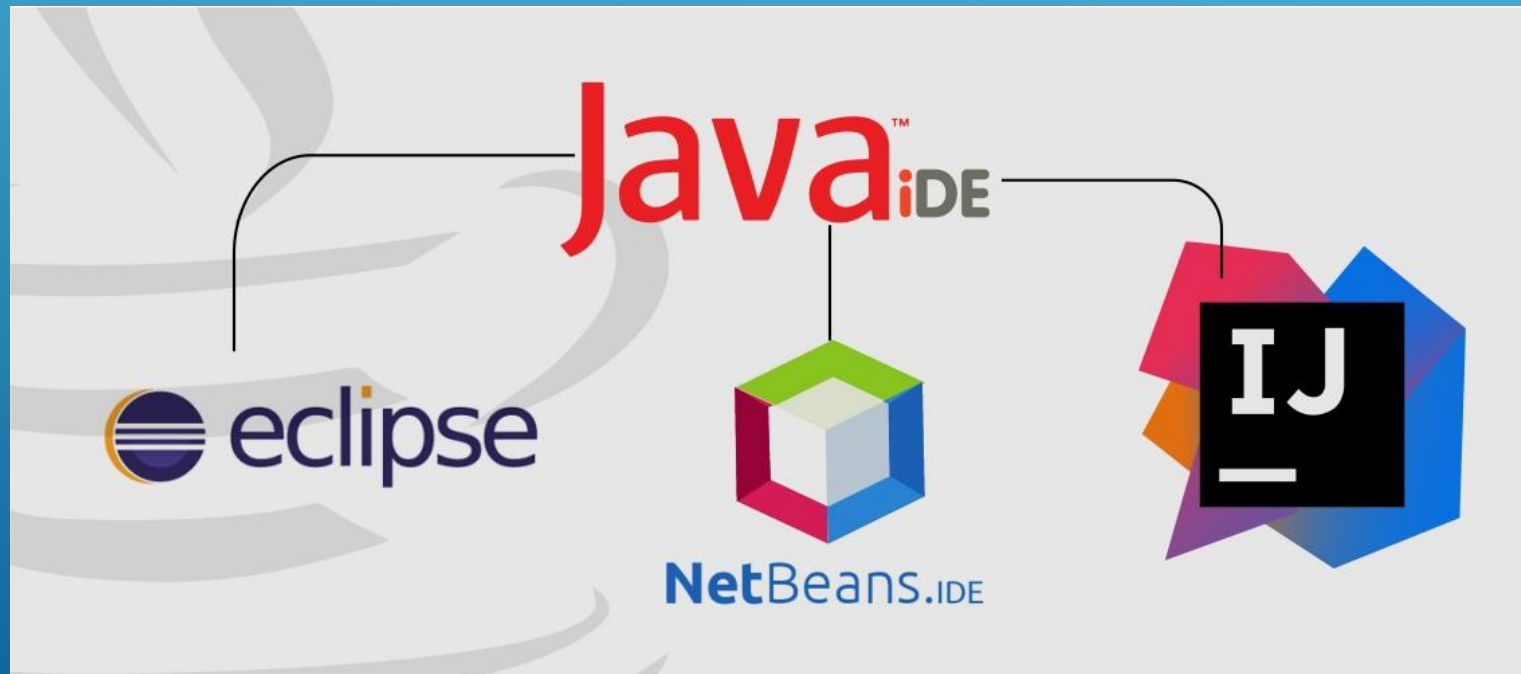
HIGH PERFORMANCE

High Performance

# WHAT IS AN IDE?

(INTEGRATED DEVELOPMENT ENVIRONMENT)

- Software application that provides comprehensive facilities to computer programmers for software development .
- An IDE normally consists of source code editor, build automation tools and a debugger.



# Naming Conventions



- All Java **class** names, **interfaces**, **abstract** classes names and **enum** must be started with Upper case letters and subsequent symbols must also be Upper case letters.
- All Java **variables** names must be started with lower case letters and the subsequent symbols must be upper case letters.
- All Java **method** names must be started with lower case letters but the subsequent symbols must be uppercase letters.
- All Java **constants** must be provided in Upper case letters.
- All Java **package** names must be provided in lower case letters.



# Comment



- Single Line Comment
- Multi-line Comment
- Documentation Comment

```
1 // a single line comment is declared like this
2 /* a multi-line comment is declared like this
3    and can have multiple lines as a comment */
4 /** a documentation comment starts with a delimiter and ends with */
```

# Package



## Package Section:

- package is the collection of related classes and interfaces as a single unit.
- Predefined package (by Java PL and provided along with Java software). `java.io`, `java.util`, `java.sql`
- User defined package (by developer as per application requirement).
- package is a keyword and any keyword in java is written in lowercase letters.



# Import Statement



- To make available classes and interfaces of other packages into the present Java file, so it can be used.
- We can import a specific class or classes in an import statement.
- An import statement is always written after the package statement but it has to be before any class declaration.

```
1 import java.util.Date; //imports the date class
2 import java.applet.*; //imports all the classes from the java applet package
```

# Class

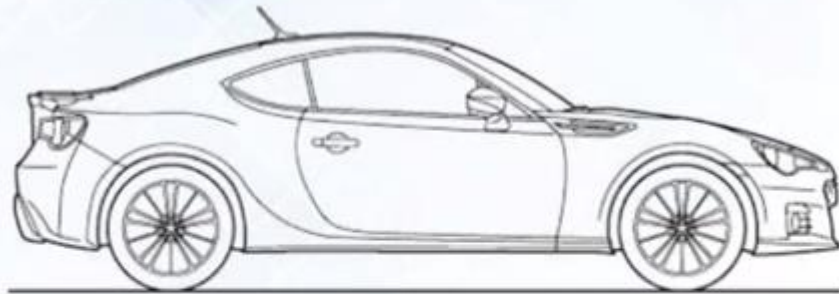


Classes



- ❑ A class in java is a blueprint which includes all the data.
- ❑ It describes the state and behavior of a specific object.

**Example :**



**Syntax :**

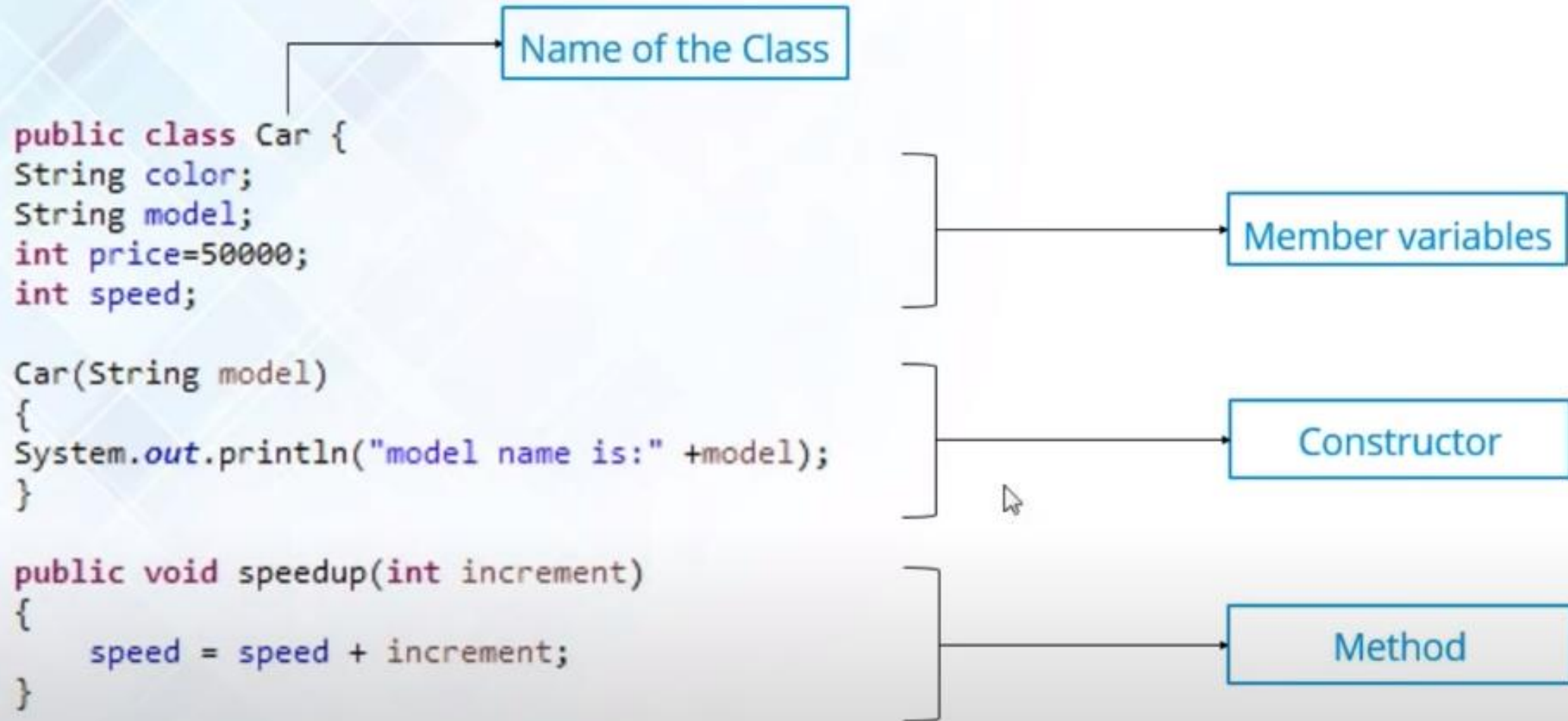
```
Public class Car {  
    Color  
    Model  
    Price  
    speedUp();  
    gearChange();  
}
```

Class name

Variables

Methods

# Structure of a Class



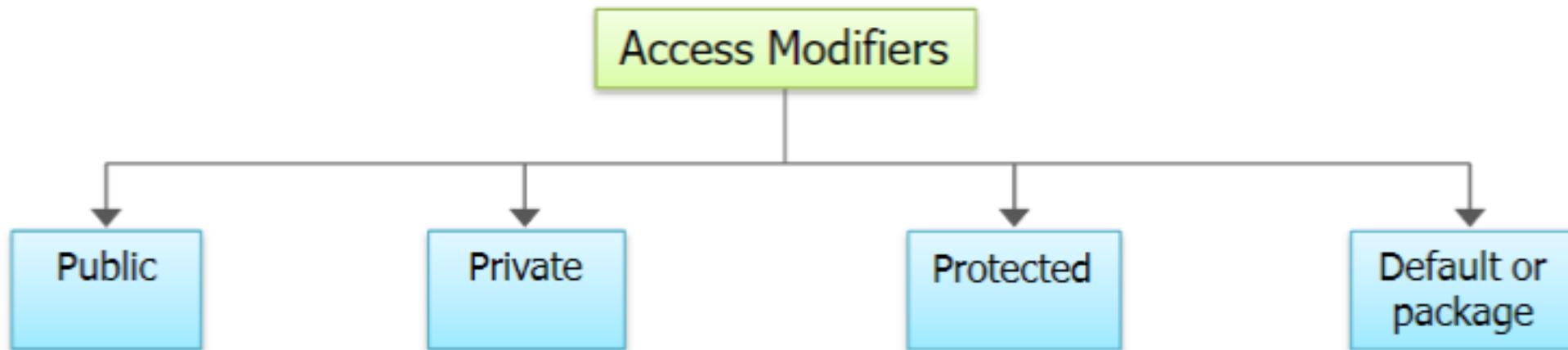
# Main Class



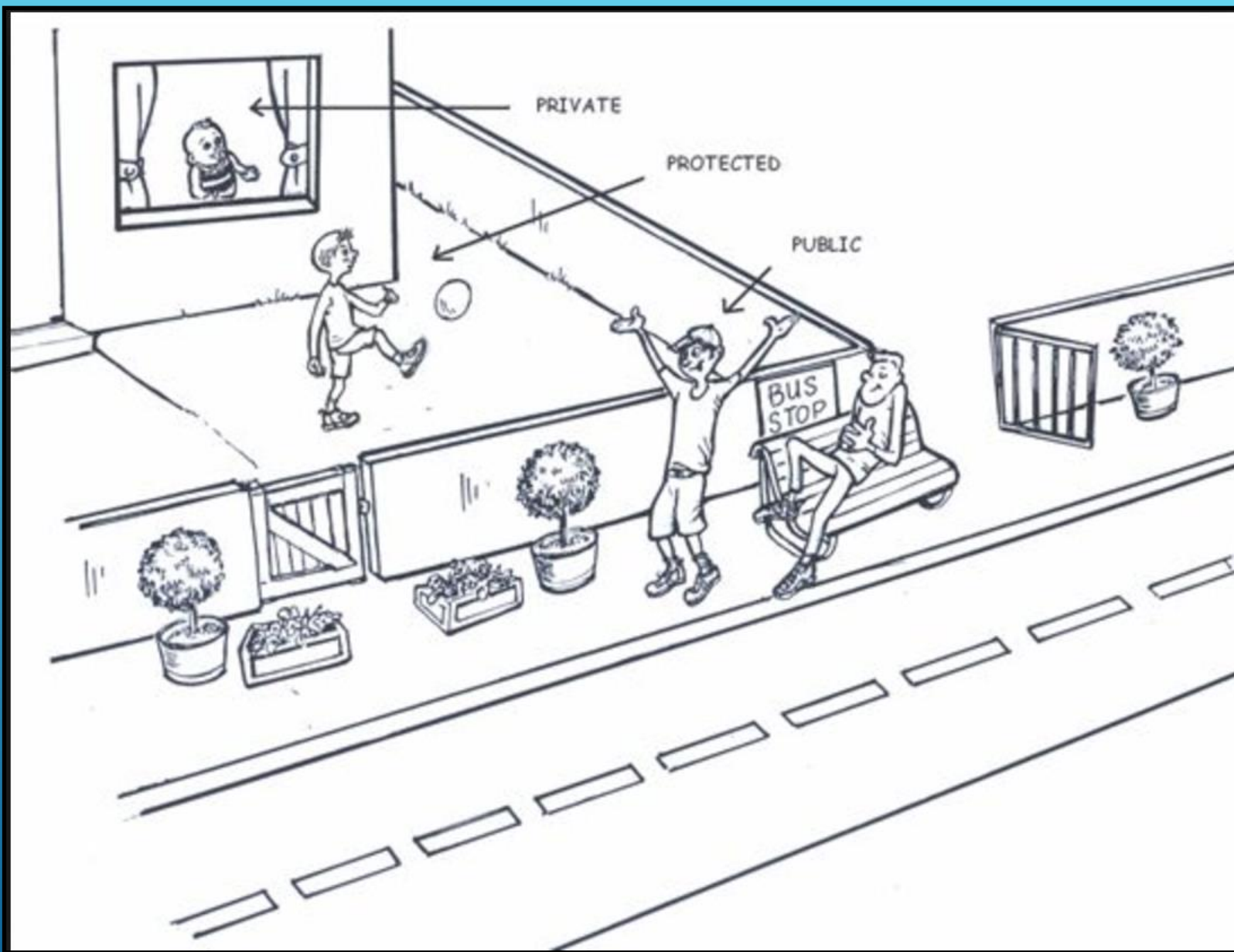
## Main Class Section:

Main Class is a java class, it includes `main()` method.  
The main intention of **`main()` method** is:

- To manage application logic which we want to execute by JVM directly we have to use `main()` method.
- To define starting point and ending point to the application execution we have to use `main()` method.



Access Modifiers	Same Class	Same Package	Sub Class	Other Packages
Public	Y	Y	Y	Y
Private	Y	N	N	N
Protected	Y	Y	Y	N
Default	Y	Y	N	N





# My First Program

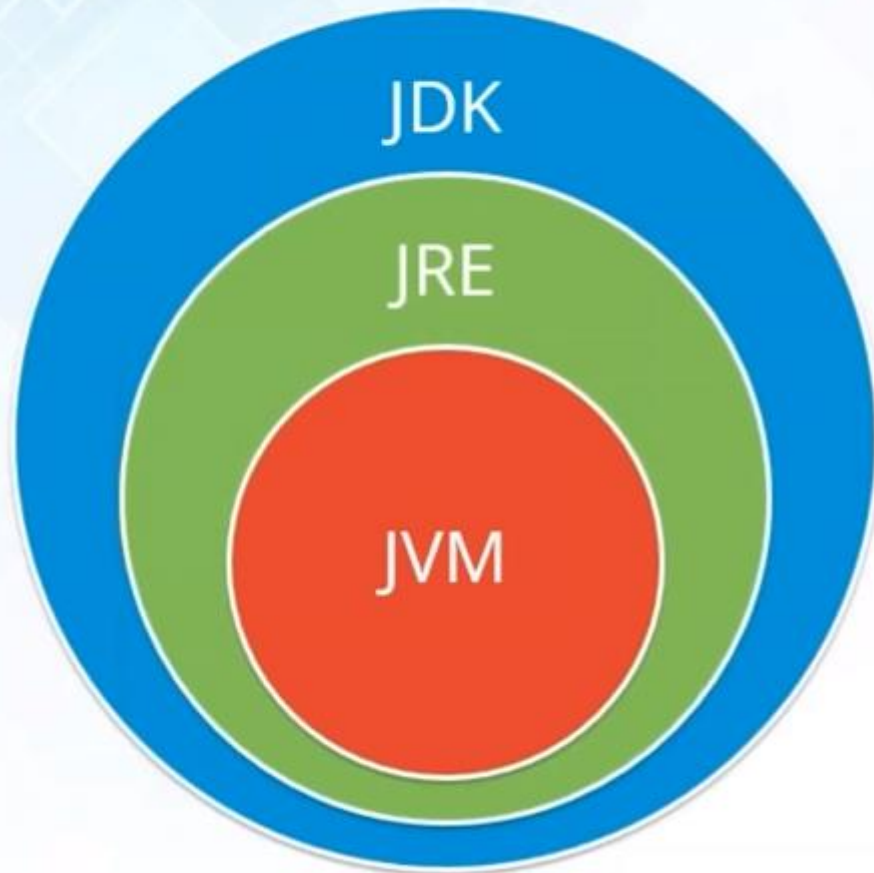


```
class Test
{
    public static void main(String[] args)
    {
        System.out.println("First Java Application");
    }
}
```

Diagram illustrating the components of the Java code snippet:

- Keyword**: `class`
- Class Name**: `Test`
- Access Modifiers**: `public`
- Return type**: `void`
- Method Name**: `main`
- Predefined Class name**: `String`
- User defined variable**: `args`
- Predefined Class Name**: `System`
- Predefined reference variable**: `out`
- Predefined Method**: `println`
- String data**: `"First Java Application"`

# Java Architecture



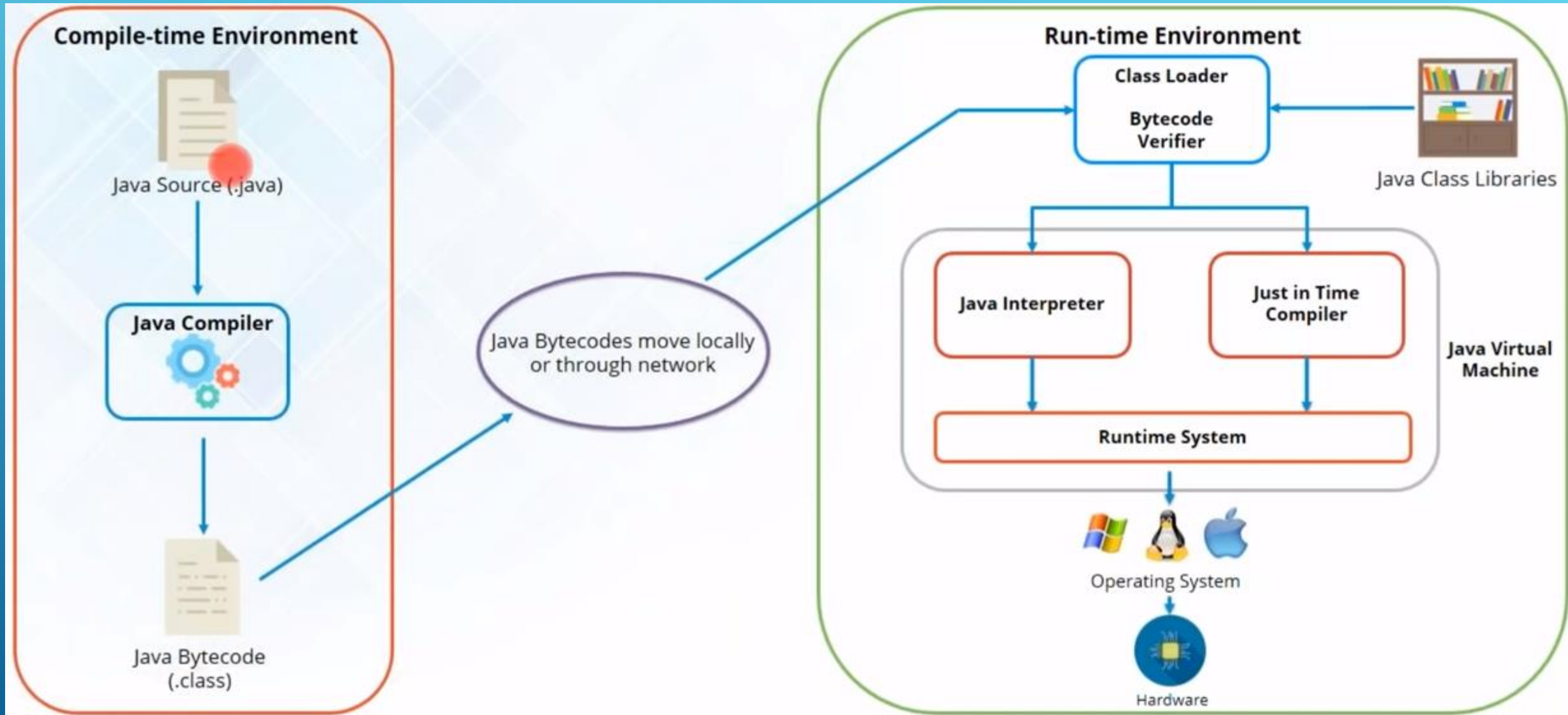
**JVM (Java Virtual Machine)** is an abstract machine. It is a specification that provides runtime environment in which java bytecode can be executed.

**JRE (Java Runtime Environment)** is a runtime environment which implements JVM and provides all class libraries and other files that JVM uses at runtime.

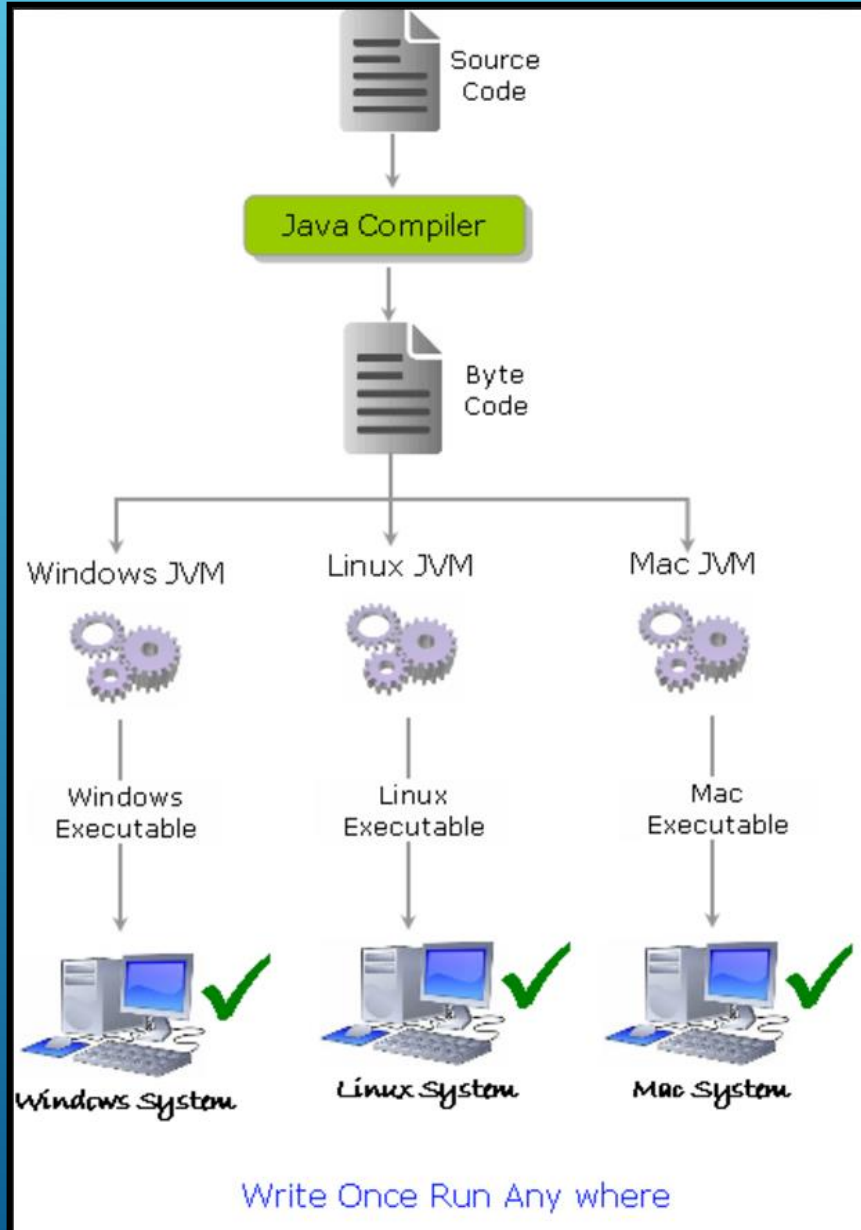
**JDK(Java Development Kit)** is the tool necessary to compile, document and package Java programs. The JDK completely includes JRE.



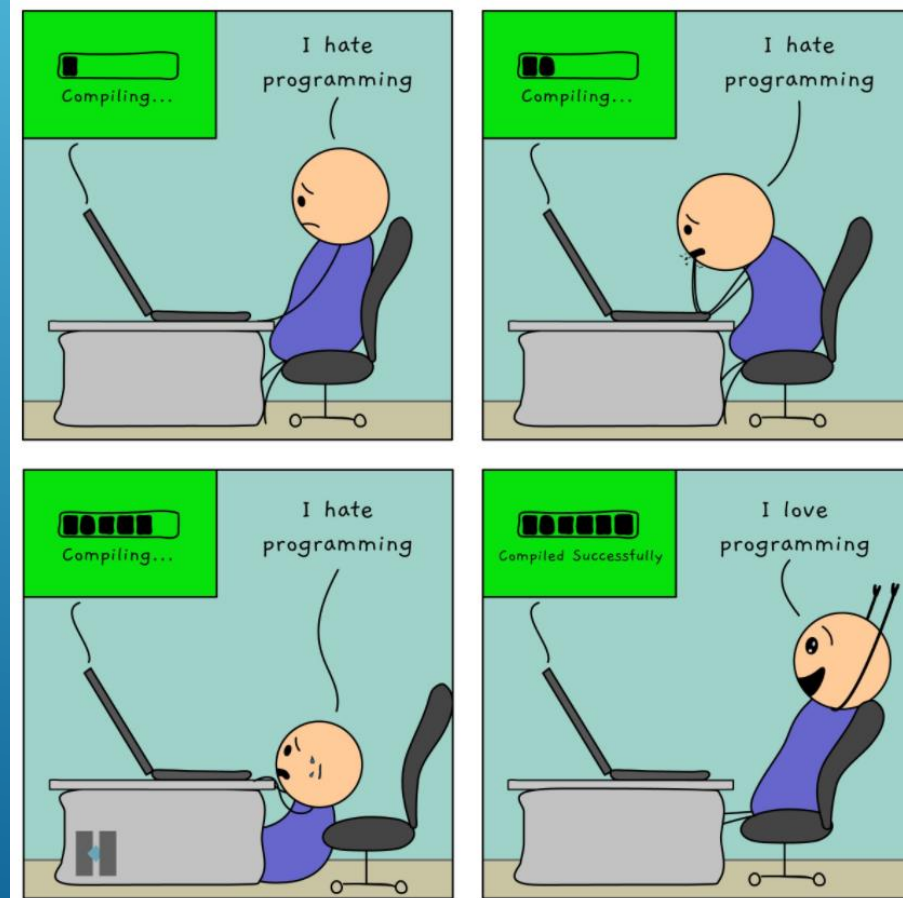
# Java Architecture



# Java Architecture



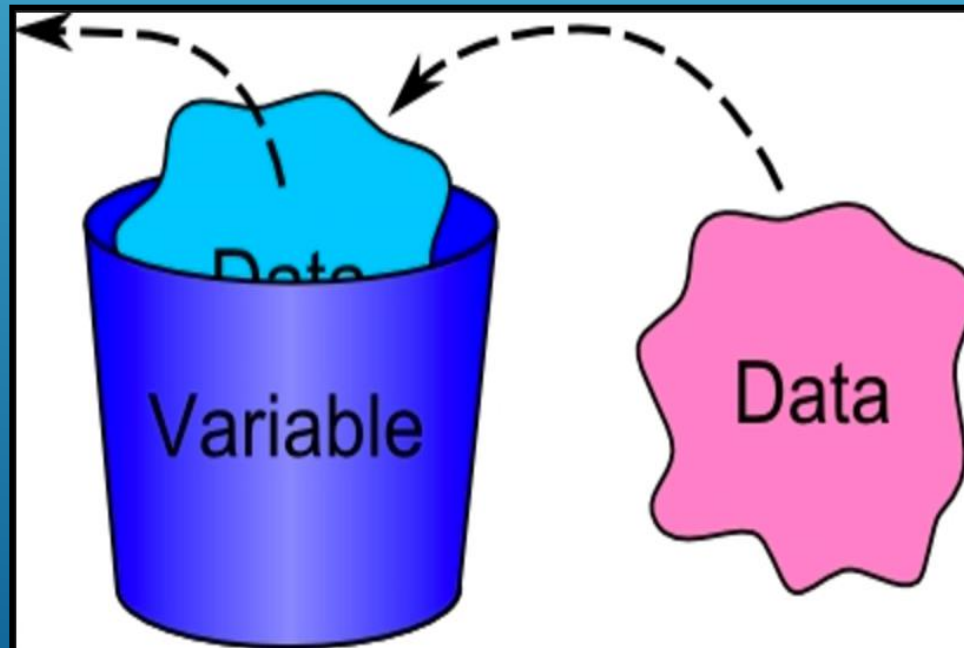
Bytecode is an intermediate code which gets generated when a Java file is compiled using a Javac compiler. After compilation .class file is generated which contains the byte code. This code is platform independent.



# Java Variable



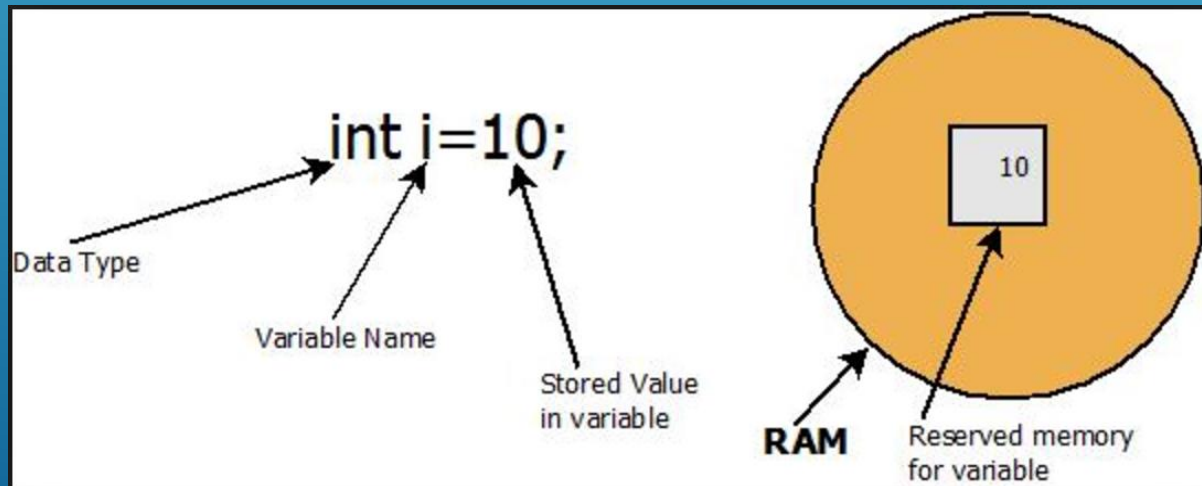
- Variables are nothing but reserved memory locations to store data values, by using this values we are achieving the project requirements. This can be any kind of information ranging from texts, numbers, sentences, etc.
- This means that when you create a variable you reserve some space in the memory. A variable thus has a data type.
- In simple words we can say variables are containers that holds the data.



# Variable Declaration and Initialisation



- A variable must be declared by specifying the variable's name and the type of information that it will hold.
- When we assign a value to variable called - variable initialisation
- Variables are containers for storing data values
- When a variable is referenced in a program (by variable name), its current value is used
- Syntax : type variable = value;



data type

variable name

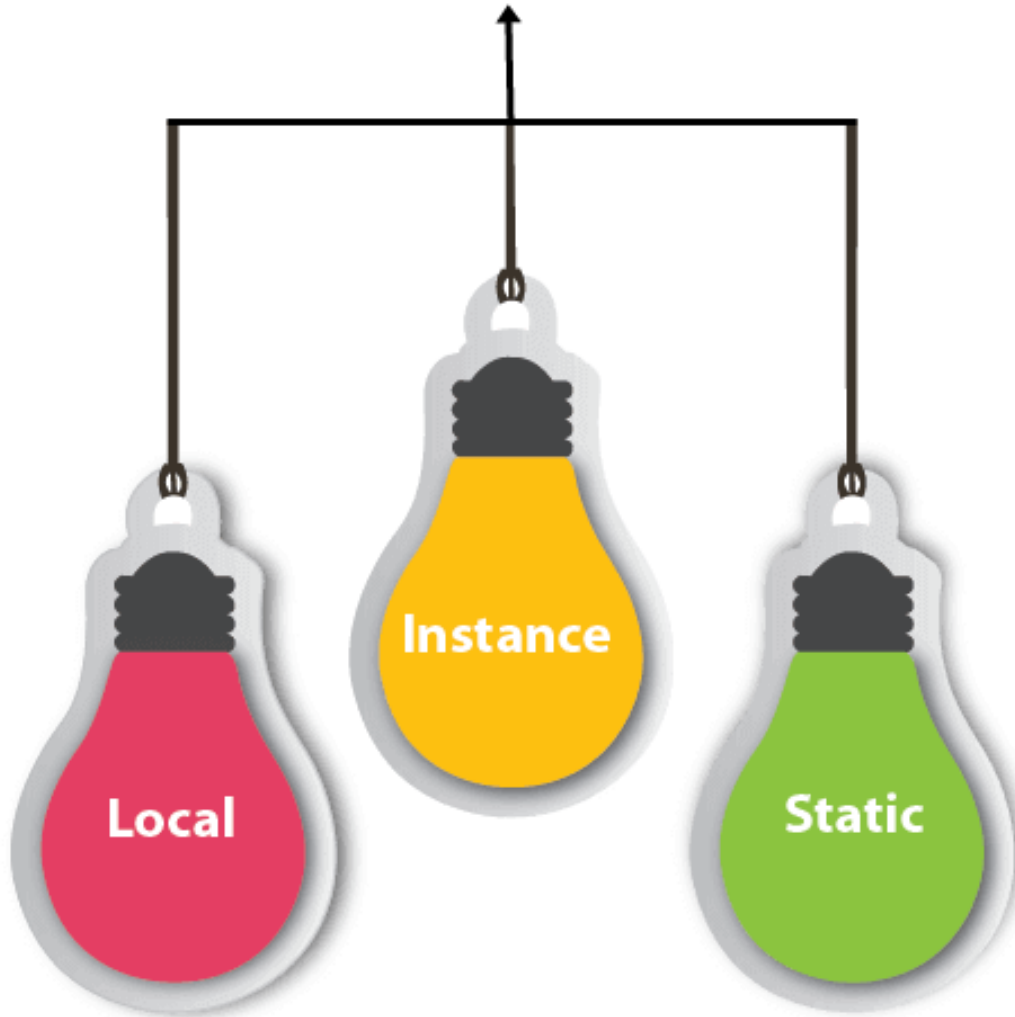
`int total;`

`int count, temp, result;`

Multiple variables can be created in one declaration



# Types of Variables



**Local Variable:** A variable declared inside the body of the method is called local variable. You can use this variable only within that method and the other methods in the class aren't even aware that the variable exists.

A local variable cannot be defined with "static" keyword.

**Instance Variable:** A variable declared inside the class but outside the body of the method, is called instance variable. It is not declared as static.

It is called instance variable because its value is instance specific and is not shared among instances.

**Static variable:** A variable which is declared as static is called static variable. It cannot be local. You can create a single copy of static variable and share among all the instances of the class. Memory allocation for static variable happens only once when the class is loaded in the memory.



# Types of Variables



## Local

```
class EmployeeId {  
    public void EmployeeId()  
    {  
        // local variable ID  
        int id = 0;  
        id = id + 6;  
        System.out.println("Employee ID : " +  
id);  
    }  
  
    public static void main(String args[])  
    {  
        EmployeeId obj = new EmployeeId();  
        obj.EmployeeId();  
    }  
}
```

## Instance

```
class Price {  
    // Instance variables that are declared in a class  
    // and not inside any function  
    int guitarPrice;  
    int pianoPrice;  
    int flutePrice;  
}  
  
public class Main {  
    public static void main(String args[])  
    {  
        Price ob1 = new Price();  
        ob1.guitarPrice = 10000;  
        ob1.pianoPrice = 5000;  
        ob1.flutePrice = 1000;  
        System.out.println("Price for second object:");  
        System.out.println(ob1.guitarPrice);  
        System.out.println(ob1.pianoPrice);  
        System.out.println(ob1.flutePrice);  
    }  
}
```

## Static

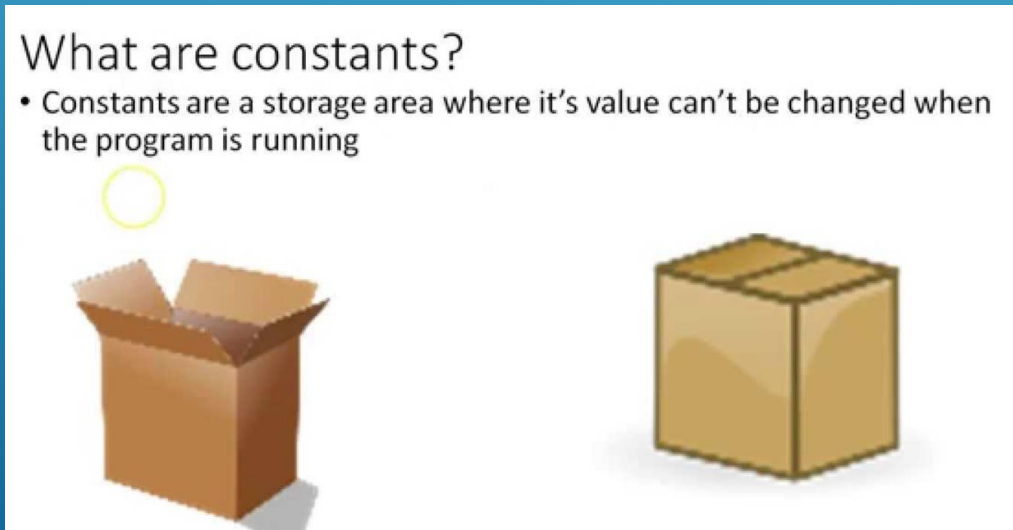
```
class Manager {  
    // static variable salary  
    public static double salary;  
    public static String name = "Jonathan";  
}  
  
public class Main {  
    public static void main(String args[])  
    {  
        // accessing static variable without  
        object  
        Manager.salary = 90000;  
        System.out.println(Manager.name + "'s  
avg salary:"  
+ Manager.salary);  
    }  
}
```

# CONSTANTS



- Constants in java refers to the fixed values, which do not change during execution of the program.
- A static final variable is effectively a constant.
- Java constants are normally declared in ALL CAPS. Words in Java constants are normally separated by underscores.
- An example of constant declaration in Java is written below:

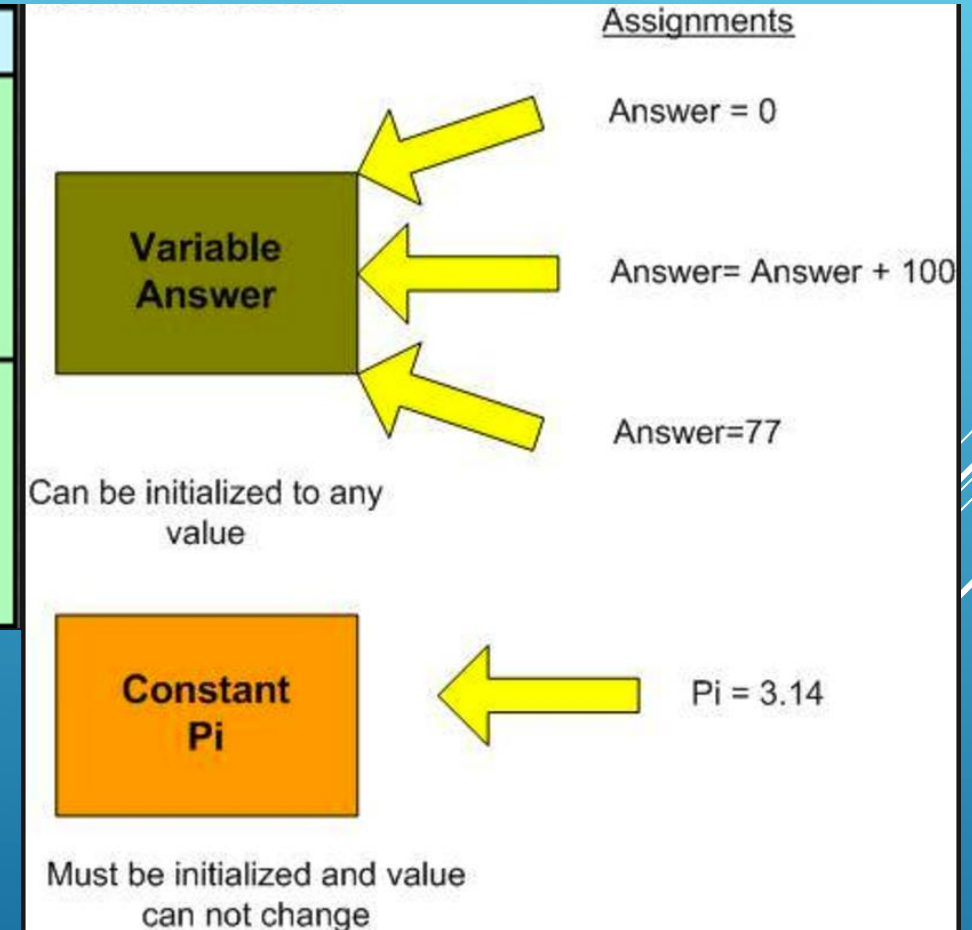
```
public class MaxUnits {  
    public static final int MAX_UNITS = 25;  
    static final double SALES_TAX_RATE = 0.85;  
}
```



# CONSTANTS VS VARIABLES



	Constants	Variables
Characteristics	Value is not changeable during the course of the program.	Value can be changed anytime during the course of the program.
Usage	Use constant when you want to declare something that won't be changed midway in your program execution.	Use variable to store data that may or will change during the running of the program.





# IDENTIFIERS IN JAVA



- An identifier is a name to represent a variable, object or method.
- Identifiers may contain letters (A-Z and a-z), digit (0-9), underscores (\_) and dollar signs(\$).
- Identifiers cannot start with digit.
- Identifiers should be meaningful. (age, sum, totalVolume, bookPrice).
- Example :
  - Reserved words cannot be used.
  - They cannot start with a digit but digits can be used after the first character (e.g., name1, n2ame are valid).
  - They can start with a letter, an underscore (i.e., "\_") or a dollar sign (i.e., "\$").
  - You cannot use other symbols or spaces (e.g., "%", "^", "&", "#").

# RESERVED WORDS (KEYWORDS)



abstract	assert	boolean	break
byte	case	catch	char
class	const	continue	default
do	double	else	enum
extends	final	finally	float
for	goto	if	implements
import	instanceof	int	interface
long	native	new	package
private	protected	public	return
short	static	strictfp	super
switch	synchronized	this	throw
throws	transient	try	void
volatile	while		

- In the Java programming language, a keyword is one of 50 reserved words that have a predefined meaning in the language; because of this, programmers cannot use keywords as names for variables, methods, classes, or as any other identifier.

# HOMEWORK

- Interview preparation for homework
  - Tell me about yourself?
  - What do you know about the company?
  - Revise all manual testing topics so far.

Manual Interview	Company
Anjana	Sainsburys
Kamal	Facebook
Urmi	Jaguar/Land Rover
Kiran	AXA Insurance
Smita	Terraquest Solutions
Priyanka	BT
Malti	Subway
Nilam	Experian
Avni	The Access Group
Neha	Barclays
Ronak	Experian
Karishma	MHR Global
Khyatiben	Bet365
Bhavita	TCS
Naimesh	Tesco



Interview week:  
20<sup>th</sup> December 2021 onwards