Core Java Programming

Presented By:

Corporate Trainer

Mallikarjuna G D

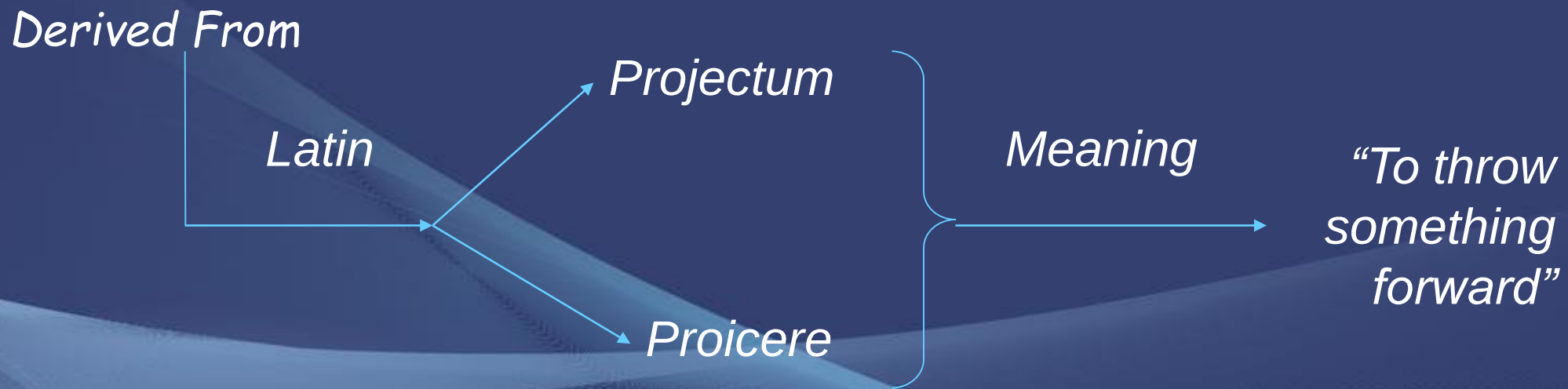
gdmallikarjuna@gmail.com

CONTENTS

- ✓ PROJECT
- ✓ DOMAIN, PLATFORM
- ✓ PROEJCT , PRODUCT
- ✓ PROJECT WORLD
- ✓ TECHNOLOGY
- ✓ ENTERPRISE
- ✓ ARCHTIECTURE
- ✓ JAVA TECH STACK
- ✓ PROCESS
- ✓ JAVA DEVELOPMENT ENVIROMENT
- ✓ JAVA FUNDAMENTALS

What is Project?

- A project is a temporary effort to create a unique product or service. Projects usually include constraints and risks regarding cost, schedule or performance outcome.



Differences

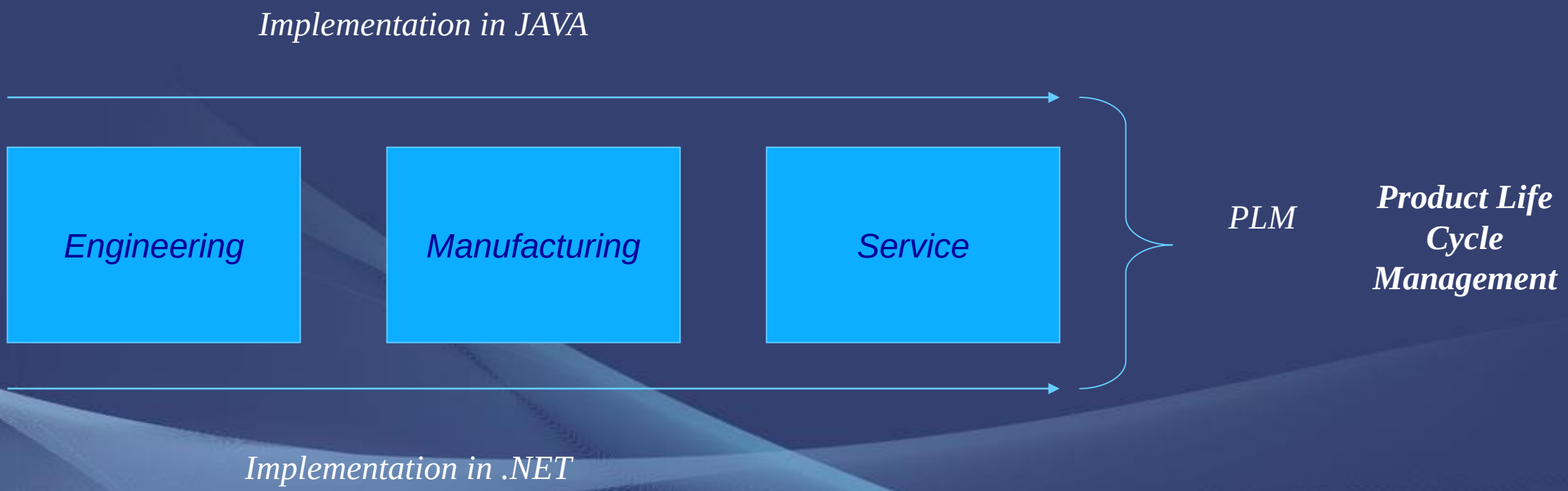
Domain

- It is a field study that defines a set of common requirement Terminology and functionality for any software program constructed to solve a problem in the area of computer programming.
- EX: PLM Domain

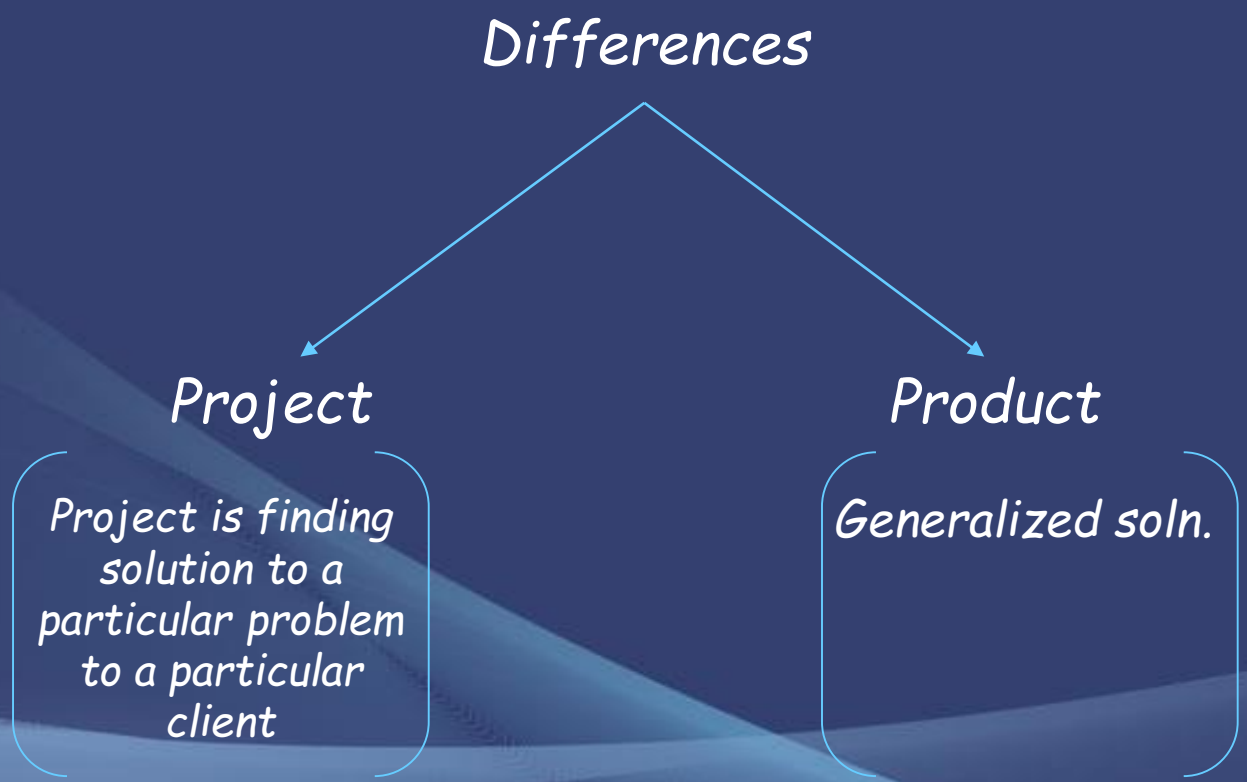
Platform

- A platform is a "Blue print for the evolution of popular software or specification."
- It depends a standard around which a system can be developed.
- EX: Java platform , .Net platform

PDM Domain:


Implementation in JAVA
Engineering
Manufacturing
Service
PLM
Product Life Cycle Management
Implementation in .NET

PROJECT V/S PRODUCT

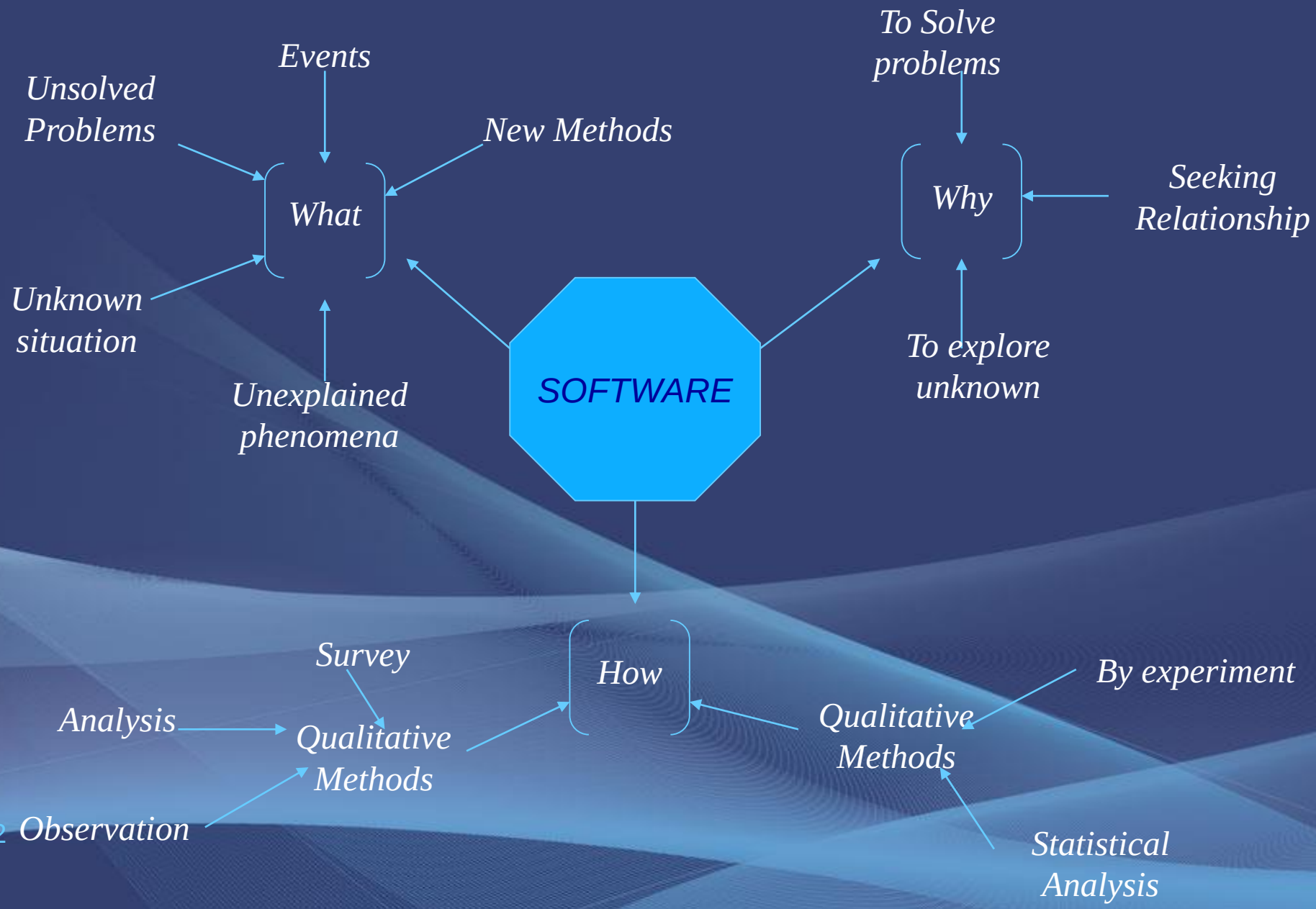


PRODUCT V/S PROJECT

- Here the requirements are gathered from the market survey.
- Develop the application for Global clients
- End users are more than one
- Never ending process
- EX: Mobile

- Here the requirements are gathered from one particular client.
- Develop the application for single clients
- End user is one
- Once developed it will be finished
- EX: Designing one particular game for Nokia mobile

AIM OF SOFTTWARE



- The real-time problems can be addressed through the combinations of products and process
- The aim of technology to accomplish the human needs
- Inotherwords, it is application of science to solve problems
- Java as Technology provides Programming language, development environment, application environment and deployment environment

- System : core software for any system to drive
  - Languages used : C, C++ or Python
- Embedded : combination of specific hardware and software
  - Languages used : C, C++ programming, microcontroller, microprocessors, Linux operating system, RTOS (Realtime operating systems)
- Application: Automation of the manual work for giving convenience to the end users.
  - Domains : CRM, PLM, BANKING,INVESTMENT,FINANCE, RETAIL,EDUCTATION so on
  - Languages used: Java, .NET, Python

- Enterprise Software(EA) : it is computer software supports for the needs of the organization which include any business automation such as logistics, supply chain, transportations, agriculture, education, retail, sales automation and banking system so on.
- These software's are scalable, enhances the efficiency and productivity
- FRONTEND Technologies : HTML4/5, CSS3/4, BOOTSTRAP, JAVASCRIPT, ANGULAR, REACT, NPM, WEBPACK, GRUNT,GULP, JASMINE/KARMA
- BACKEND Technologies : core java, jsp, servlet, hibernate, spring,spring mvc, spring rest, spring boot,mysql, postgressql, Junit
- DEPLOYMENT : JENKINS, DOCKER, AWS, Linode, AZURE
- PROCESS: JIRA, CONFLUENCE
- BUILD MANAGEMENT: MAVEN

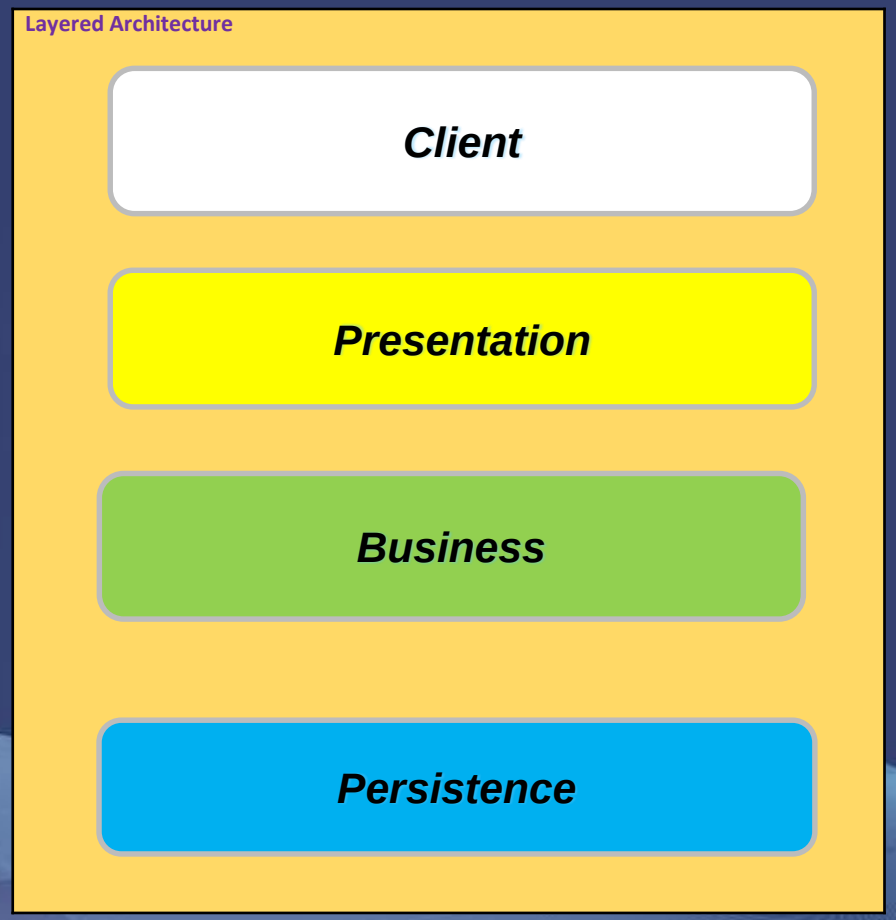
ARCHITECTURE

- It is a blueprint of the system to be developed.
- It is integration of Design, Business needs, quality support workflow, IT infrastructure, Technology stack, Deployment and Humans intervention
- Advantages
  - Overall structure and interfaces
  - Snapshot of Workflow
  - Stakeholders
  - Ownership differentiations
- Limitations
  - No standards and tools
  - No standard design process

ARCHITECTURE

- It is a blueprint of the system to be developed.
- It is integration of Design, Business needs, quality support workflow, IT infrastructure, Technology stack, Deployment and Humans intervention
- Advantages
  - Overall structure and interfaces
  - Snapshot of Workflow
  - Stakeholders
  - Ownership differentiations
- Limitations
  - No standards and tools
  - No standard design process

LAYERED ARCHITECTURE



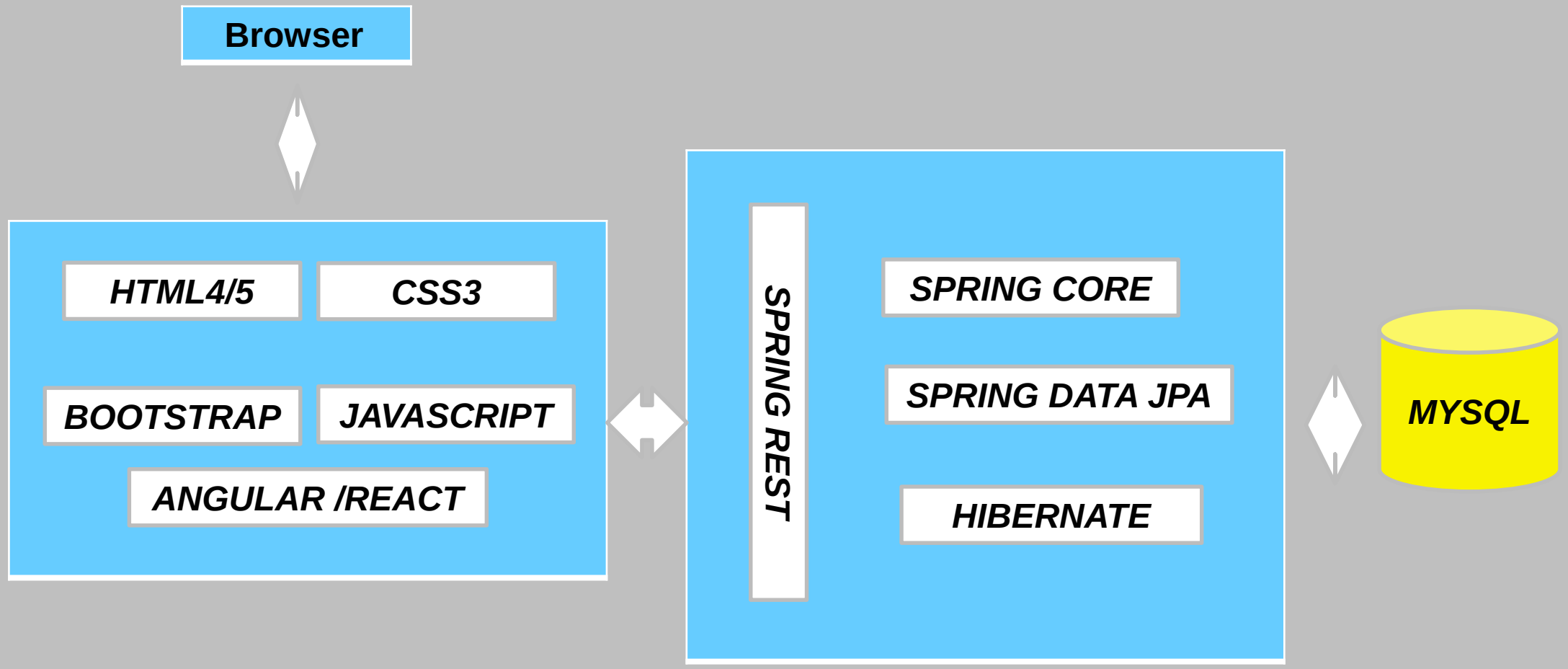
End user submits the request

Renders the what u end user sees on website and mobile

Implementations of the business logic

Save the details on permanent storage

TECHNOLOGY STACK



History:

- Java is a General Purpose, Object Oriented Programming Language developed by James Gosling at Sun Microsystems in 1991 and released in 1995.
- Originally designed for small, embedded systems in electronic appliances like set-top boxes.
- There are many java versions that has been released, current stable release of java is Java SE 9 released On 21 September 2017.

- The Java technology is:
  - A programming language
  - A development environment
  - An application environment
  - A deployment environment
- Programming Language
  - As a programming language, Java can create all kinds of applications that you could create using any conventional programming language
- A Development Environment
  - As a development environment, Java technology provides you with a large suite of tools:
  - A compiler (javac)
  - An interpreter (java)
  - A documentation generator (javadoc)
  - A class file packaging tool and so on...

- Application environment

  Java technology applications are typically general-purpose programs that run on any machine where the Java runtime environment (JRE) is installed.

- Deployment Environment

  There are two main deployment environments:

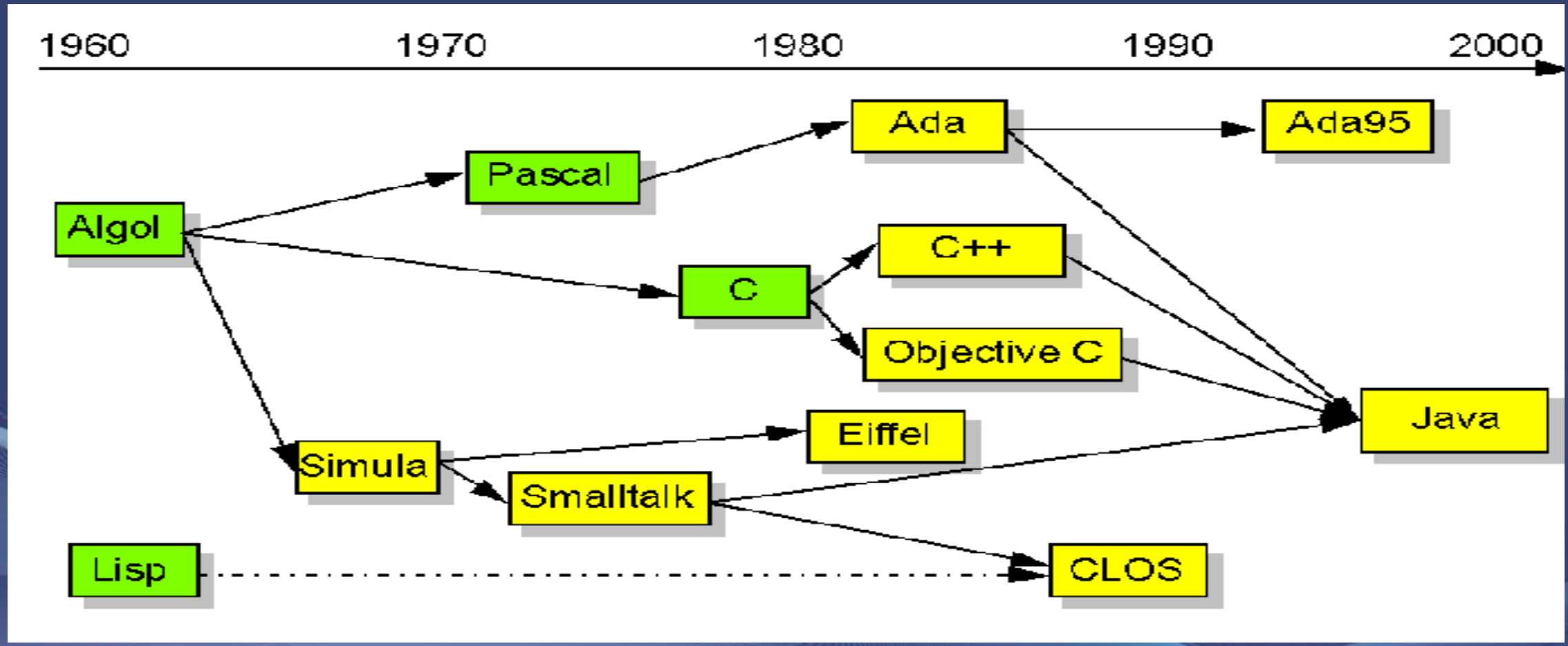
  – The JRE supplied by the Java 2 Software Development Kit (SDK) contains the complete set of class files for all the Java technology packages, which includes basic language classes, GUI component

  – The other main deployment environment is on your web browser.

- Most commercial browsers supply a Java technology interpreter and runtime environment.

HISTORY

JAVA Versons	Version Number	Release Date
JDK 1.0(oak)	1	Jan-96
JDK 1.1	1.1	Feb-97
J2SE 1.2(playground)	1.2	Dec-98
J2SE 1.3(Kestrel)	1.3	May-00
J2SE 1.4(Merlin)	1.4	Feb-02
J2SE 5.0(Tiger)	1.5	Sep-04
Java SE 6(Mustang)	1.6	Dec-06
Java SE 7(Dolphin)	1.7	Jul-11
Java SE 8	1.8	Mar-14
Java SE 9	9	September, 21st 2017
Java SE 10	10	March, 20th 2018
Java SE 11	11	September, 25th 2018
Java SE 12	12	March, 19th 2019
Java SE 13	13	September, 17th 2019
Java SE 14	14	March, 17th 2020
Java SE 15	15	September, 15th 2020
Java SE 16	16	March, 16th 2021
Java SE 17	17	September, 14th 2021
Java SE 18	18	March, 22nd 2022


1960
1970
1980
1990
2000
Algol
Pascal
Ada
Ada95
C
C++
Objective C
Simula
Eiffel
Smalltalk
Java
Lisp
CLOS

Why Java:

- Simple
- Platform Independent
- Object Oriented
- Robust and Secure
- Statically and Strongly typed
- Multithreaded
- Architectural-Neutral
- Interpreted and High Performance
- Distributed

Environment Setup:

Java is one of the most popular programming languages used to create Web applications and platforms.

It was designed for flexibility, allowing developers to write code that would run on any machine, regardless of architecture or platform.

Popular Java Editors:

- Notepad – On Windows machine, you can use any simple text editor like Notepad or TextPad.
- Netbeans – It is a Java IDE that is open-source and free. It can be downloaded from https://netbeans.org/index.html.
- Eclipse – It is also a Java IDE developed by the Eclipse open-source community and can be downloaded from https://www.eclipse.org/.

JDK, JRE and JVM:

Java Environment includes a large number of development tools and hundreds of classes and methods. and the classes and methods are part of the Java Standard Library (JSL), also known as the Application Programming Interface (API).

- JVM: The JVM provides runtime environment in which java bytecode can be Loaded, Verified and executed.
- JRE: It is the implementation of JVM. It physically exists. It contains set of libraries + other files that JVM uses at runtime.

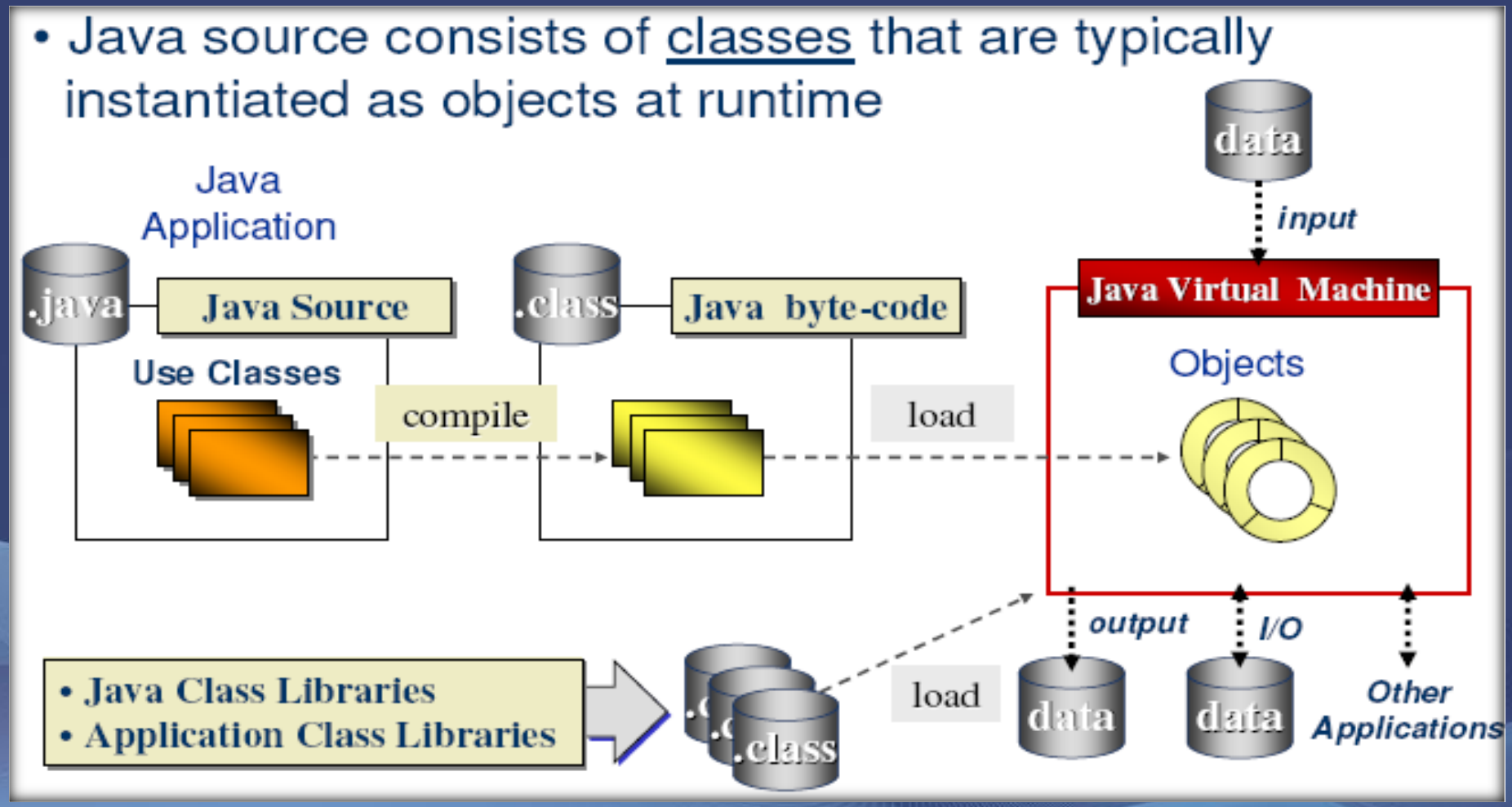
JDK, JRE and JVM (Continue…):

- JDK: Java Development Kit comes with a collection of tools that are used for developing and JRE for running Java Programs. JDK officially named "Java Platform Standard Edition (Java SE)", includes a complete JRE plus tools for developing, debugging, and monitoring Java applications.

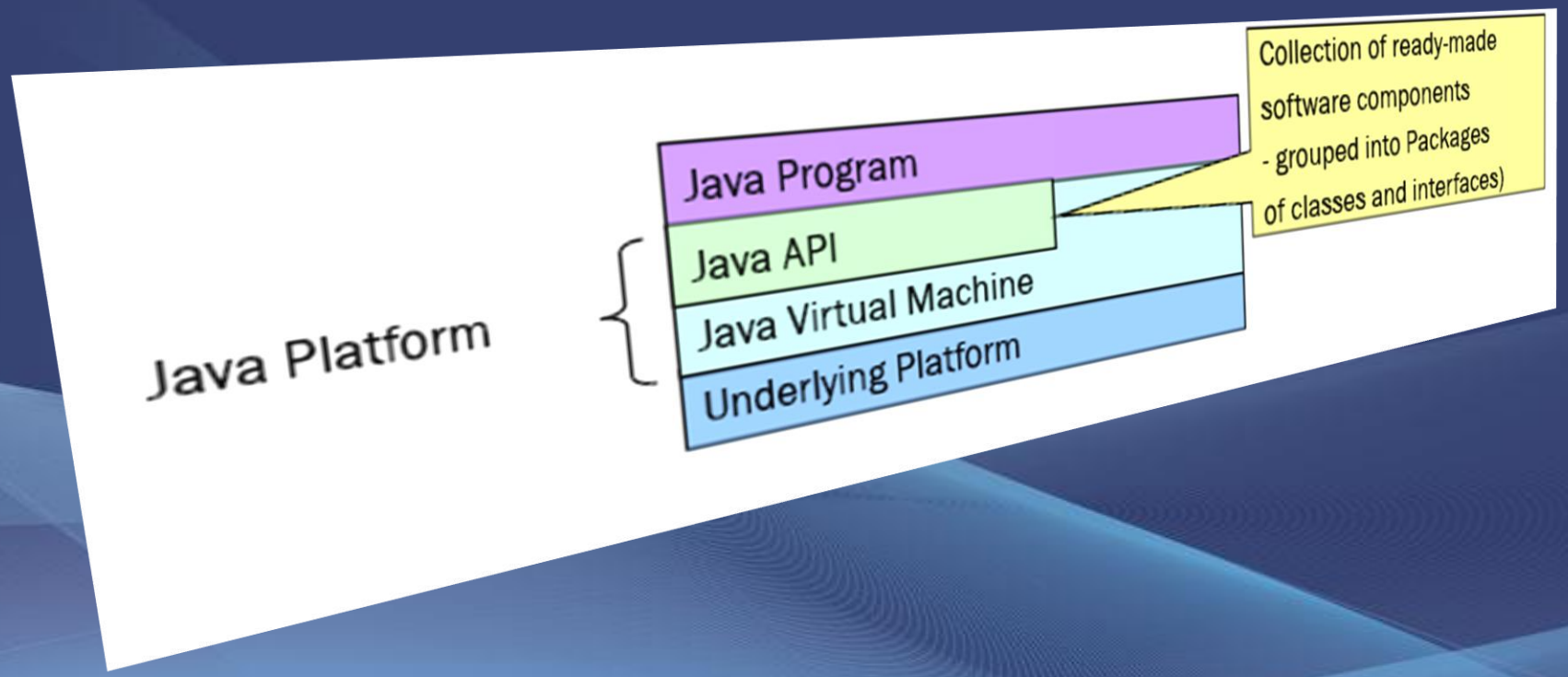
Sample Java Program:

```
class Snipe{
public static void main(string[] args)
{
System.out.println("SNIPE Tech Pvt. Ltd.");
} } // O/P - SNIPE Tech Pvt. Ltd.
```

- class keyword is used to declare a class 'SNIPE' in java.
- public keyword is an access modifier which makes visible to all.
- static keyword is used to declare method is known as static method. The core advantage is that there is no need to create object to invoke the static method. So it saves memory.
- void is the return type of the method, it doesn't return any value.
- main represents startup of the program.
- String[] args is used for command line argument.
- System.out.println() is used print statement 'SNIPE Tech Pvt. Ltd.'.


• Java source consists of classes that are typically instantiated as objects at runtime
Java Application
.java
Java Source
Use Classes
compile
.class
Java byte-code
load
data
input
Java Virtual Machine
Objects
output
I/O
Other Applications
data
data
• Java Class Libraries
• Application Class Libraries
.class
load

- Software Platform running for Running Java On top of any platform
  - Java virtual machine (JVM)
  - Java Application Programming Interface(JAVA API)



JAVA ARCHITECTURE

- Java architecture arises out of four distinct but interrelated technologies
  - The Java Programming Language
  - The Java Class File Format
  - The Java Application Programming Interface
  - The Java Virtual Machine

- object-orientation
- multi-threading
- structured error-handling
- garbage collection
- dynamic linking
- dynamic extension

Indirectly, It answers Why Java

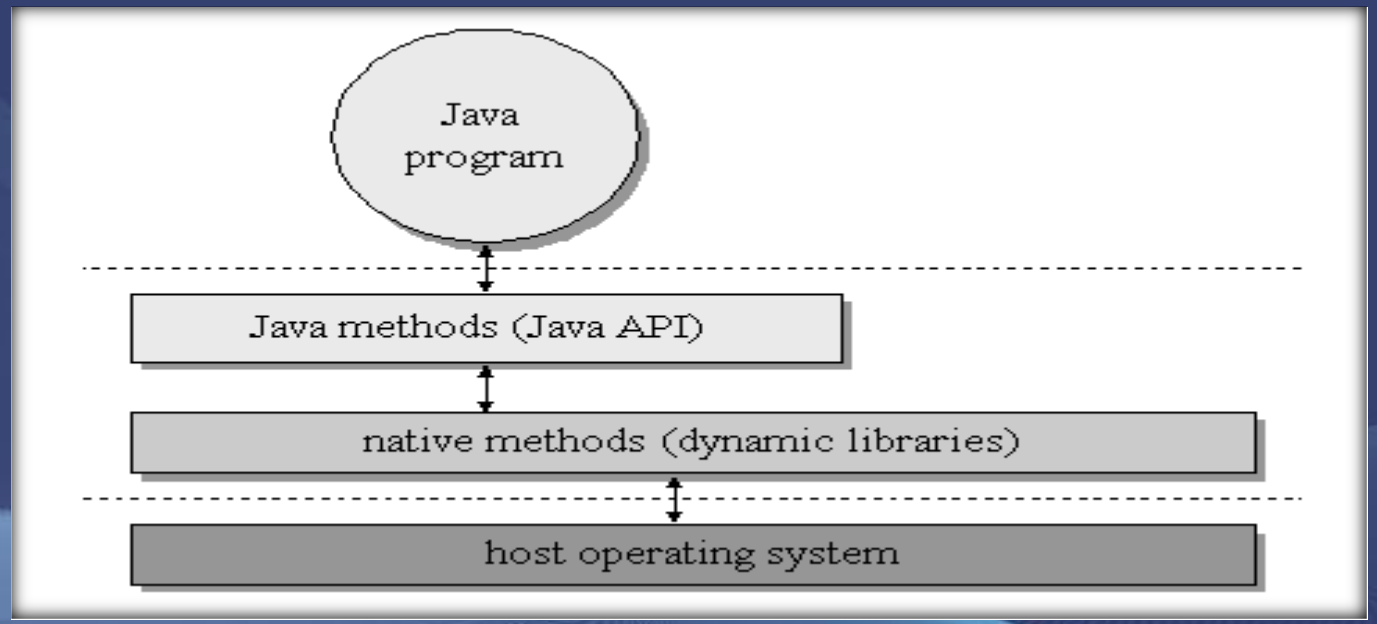
JAVA CLASS FILE

- The Java class file helps make Java suitable for networks mainly in the areas of platform-independence and network-mobility
- Its role in platform independence is serving as a binary form for Java programs that is expected by the Java virtual machine but independent of underlying host platforms
- the Java class file plays a critical role in Java's architectural support for network-mobility. First, class files were designed to be compact, so they can more quickly move across a network. Also, because Java programs are dynamically linked and dynamically extensible, class files can be downloaded as needed
- https://docs.oracle.com/javase/specs/jvms/se7/html/jvms-4.html
- https://docs.oracle.com/javase/specs/jvms/se7/html/index.html

JAVA CLASS FILE

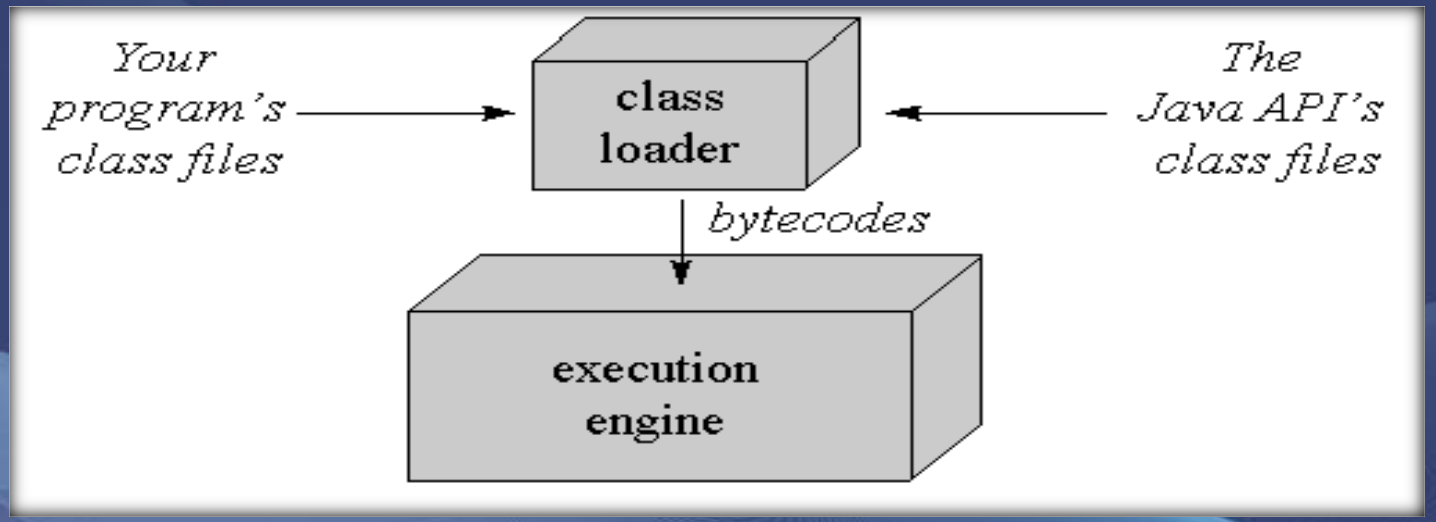
```
ClassFile {
u4 magic;  // Identify class file format
u2 minor version; //
u2 major version;
u2 constant_pool_count;
cp_info constant_pool[constant_pool_count-1];
u2 access_flags;  //permissions
u2 this_class;  //pointer to constant pool
u2 super_class;
u2 interfaces_count;
u2 interfaces[interfaces_count];
u2 fields_count;
field_info fields[fields_count];
u2 methods_count;
method_info methods[methods_count];
u2 attributes_count;
attribute_info attributes[attributes_count];
}
```

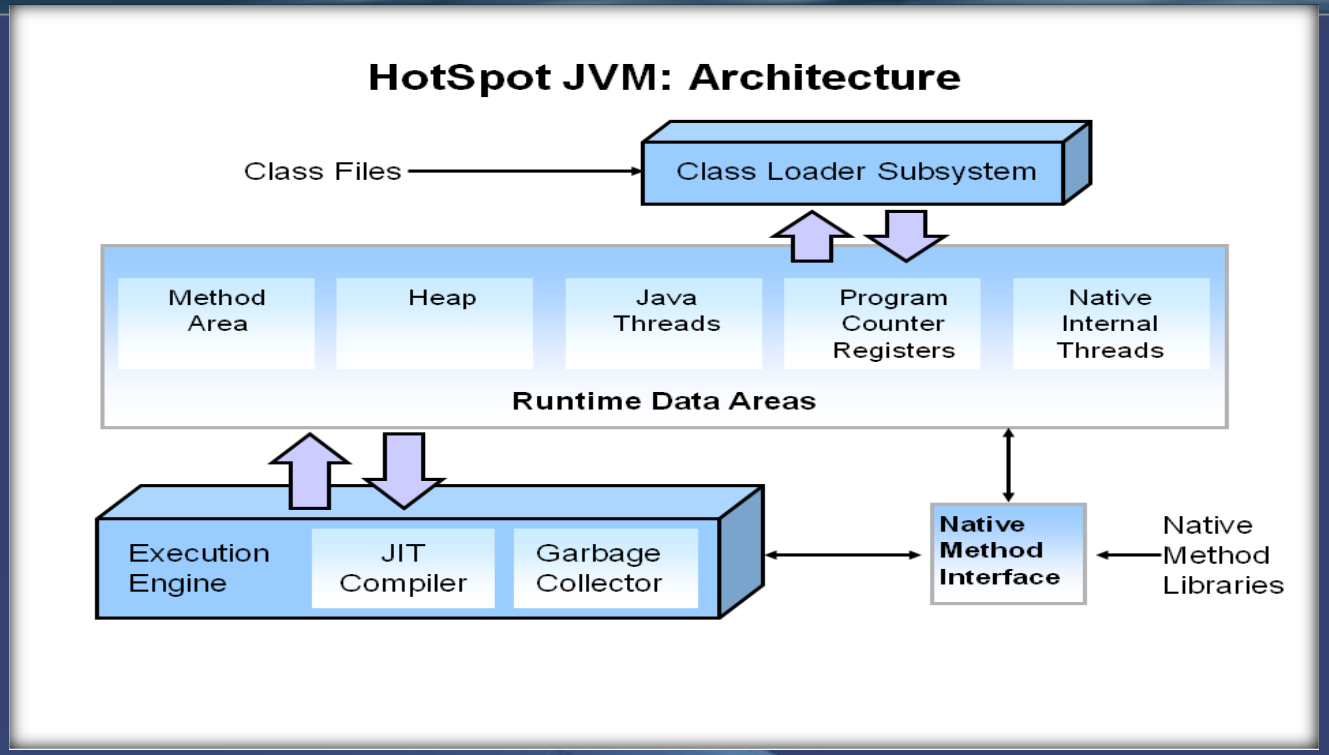
- The Java API is set of runtime libraries that give you a standard way to access the system resources of a host computer.



JAVA VIRTUAL MACHINE

- Java Virtual Machine is an abstract computer. It is a specification defines certain features every java virtual machine must have, but leaves many choices to designer of each implementation.
- It can be implemented any percentage of hardware or software.





Each JVM has:

- Class loader subsystem: a mechanism for loading types (classes and interfaces) given fully qualified names
- Execution engine: a mechanism responsible for executing the instructions contained in the methods of loaded classes

The JVM when executing a program, maintains information regarding the current state of the executing program. This information is held in various conceptual data spaces. The JVM is divided into four conceptual spaces:

Method area

This is where code and constants are kept

Java stack

Keeps stack frames containing information about method calls, parameters,etc.

Heap area

Contains objects that have been created at runtime

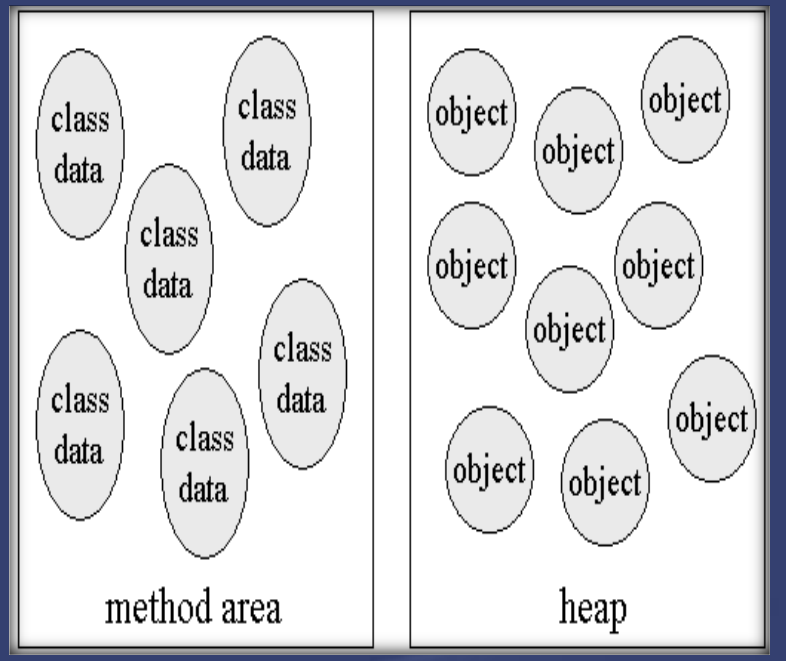
Native stacks

Used for native method implementations via JNI

JVM-METHOD AREA

Method area : This is where code and constants are kept

- Contains a sequence of classes, encapsulating the corresponding field and method information
- Contains the constant pool where constants are kept
- Each class in the class area contains:
  - Class information - class name, modifiers (public, static, etc.)
  - Super class name
  - Field information:
    - sequence of fields with their modifiers
    - static fields also contain their actual data
  - Method information:
    - sequence of methods with the method name, modifiers and method
    - Signature each non-abstract method contains a link to a sequence of bytecode instructions - the method implementation



The main JVM stack is a stack of stack frames

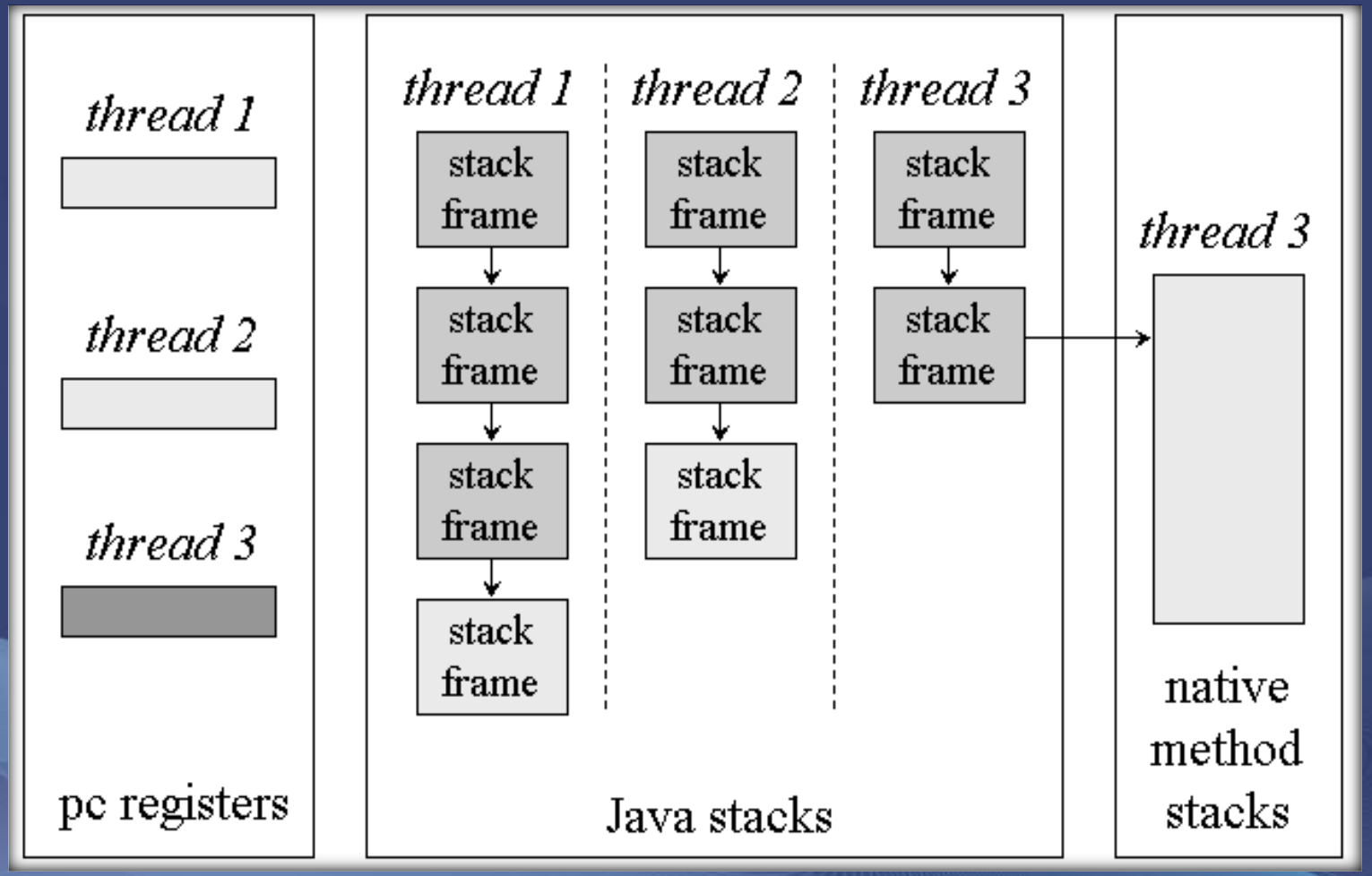
- A stack frame is a data space containing:
  - an operand stack (for executing bytecode operations on)
  - an array of local variable slots (0 to 65,535)
  - a Program Counter (PC), pointing to the currently executing instruction
- The frame on the top of the stack is called the active stack frame

At any given time, only the active stack frame can be used

- When a method is invoked:
  - a new stack frame is created and is pushed onto the JVM stack
  - the current PC is saved as part of the saved stack frame
  - the PC is reset to the start of the called method
- When a method returns:
  - the active stack frame disappears, (is popped) from the stack, and the next
  - stack frame on the stack becomes the current one
  - the program counter is reset to the instruction after the method call and execution continues

The main JVM stack is a stack of stack frames

- Native methods are used like other JVM methods, except they are implemented in some other language (e.g. C)
- Can be used for things that cannot be handled well in Java, e.g. integrating legacy code written in other languages
- Native stacks are used to keep track of state of native methods
- Do not exist on all JVM implementations and different implementations have different standards
- Java Native Interface (JNI) is a common standard


thread 1
thread 2
thread 3
pc registers
thread 1
thread 2
thread 3
stack frame
stack frame
stack frame
stack frame
stack frame
stack frame
stack frame
stack frame
stack frame
Java stacks
thread 3
native method stacks

ARCHITECTUREAL SUPPORT FOR INDEPENDENCE

The Java Platform :-

- It acts as buffer b/n java program and the underlying hardware and operating system.
- The Virtual Machine runs provides JAVA API to program which interacts with hardware of host computer.

The Java Language:-

- The ranges and behavior of primitive types are defined by the language same across different platform.

The Java Platform Version and Edition

- Java 2 platform, Standard Edition and Standard Extension API's will evolved with time and also possible of deprecated due to some reason may affect platform Independence.

Native Methods

- Invoking native method leads platform specific. It may due to
  - For Accessing features of an underlying host platform that are not accessible through the JAVA API.
  - For accessing a legacy system or using an already existing library that is not written in JAVA
  - For Speeding up the performance of a program by implementing time-critical code as native method.

- The Java Class File :-
  - The format of Class File is strictly defined and independent of any platform that hosts a Java Virtual Machine
- Scalability:-
  - It supports for wide range of hosts with varying levels of resources, from embedded devices to mainframe computer.
  - It defines java API such as yielded in three basic API sets, which demonstrate the scalability of the Java Platform
    - Enterprise Edition(J2EE)
    - Standard Edition(J2SE)
    - Micro Edition(J2ME)

FACTORS THAT INFLUENCE PLATFORM INDEPENDENCE

- The Java Platform Version and Edition
  - Java 2 platform, Standard Edition and Standard Extension API's will evolved with time and also possible of deprecated due to some reason may affect platform Independence.
- Native Methods
  - Invoking native method leads platform specific. It may due to
    - For Accessing features of an underlying host platform that are not accessible through the JAVA API.
    - For accessing a legacy system or using an already existing library that is not written in JAVA
    - For Speeding up the performance of a program by implementing time-critical code as native method
- Non –standard Run Time Libraries
  - Since JAVA standard API should be supported by all the vendors but in addition to this some vendors may give extra APIs, these usage leads to platform specific.
- Virtual Machine Dependencies
  - Since JVM implemented differently by different vendors. Following variations may leads to issue.
    - Usage of finalization for program correctness
    - Thread prioritization

- The Java platform Deployment
  - Java program runs only on computers and devices that hosts a JAVA Platform
- Bugs in Java Platform Independence
  - There is a possibility of some bugs in java distribution
- Testing
  - Java Program developed in one platform still need to test in different platforms due to variation in its implementation.
- User Interface Dependencies
  - End users on different platforms are accustomed to different ways of interacting with their computers.
  - Mapping UI libraries may be different in different platform.

GARBAGE COLLECTION

- It is a part of runtime system: it reclaims the heap-allocated records that are no longer used.
- Garbage is an unused or unreachable object in an application or program which are being currently under execution
- Garbage Collection is the process of collecting the garbage automatically from the heap or memory allocated for the particular application or program
- A garbage collector should
  - Reclaim all unused records
  - Spend very little time per record
  - Not cause significant delays
  - Allow all of memory to be used

Advantages

- Programmer is free from memory management
- System cannot crash due to memory management

Disadvantages

- GC could add overhead
- GC can occur in an non-deterministic way

Eligibility for GC Occur?

- An object can become eligible for garbage collection in different ways
  - If the reference variable that refers to the object is set to null, the object becomes eligible for garbage collection, provided that no other reference is referring to it.
  - If the reference variable that refers to the object is made to refer to some other object, the object becomes eligible for garbage collection, provided that no other reference is referring to it.
  - Objects created locally in a method are eligible for garbage collection when the method returns, unless they are exported out of the method (that is, returned or thrown as an exception).
  - Objects that refer to each other can still be eligible for garbage collection if no live thread can access either of them.
- JVM performs GC when it determines the amount of free heap space is below a threshold
  - This threshold can be set when a Java application is run

Eligibility for GC Occur?

- The garbage collector must somehow determine which objects are no longer referenced and make available the heap space occupied by such unreferenced objects
- The simplest and most crude scheme is to keep reference counter to each object.
- There are many different schemes - years of research

Common scheme of GC ?

https://www.oracle.com/webfolder/technetwork/tutorials/obe/java/gc01/index.html

- Reference counting
- Mark and Sweep
- Stop and Copy garbage collection

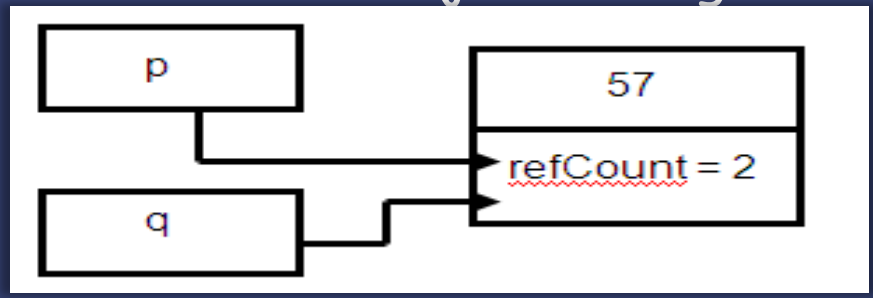
Reference counting?

- Main Idea: Add a reference count field for every object.
- This Field is updated when the number of references to an object changes.

Example

Object p= new Integer(57);

Object q = p;



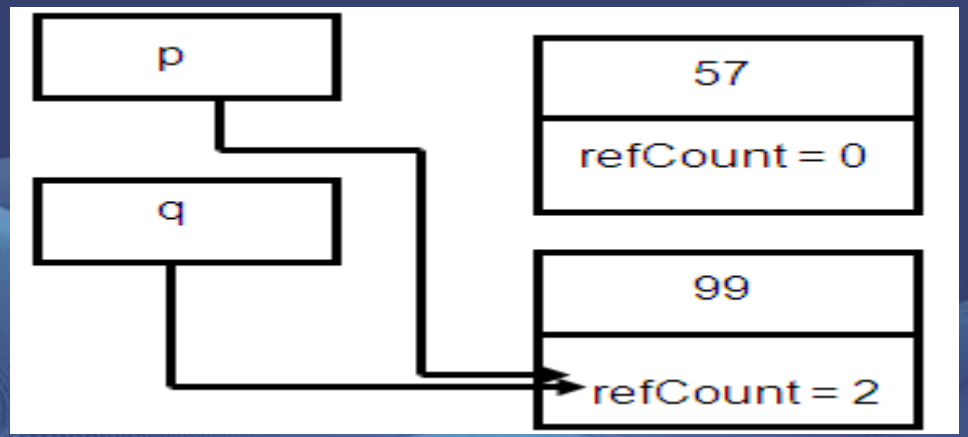
- The update of reference field when we have a reference assignment (i.e p=q) can be implemented as follows

Example:

Object p = new Integer(57);

Object q= new Integer(99);

- p=q



Mark-and-Sweep Garbage Collection?

Obtain locks and suspend threads

Mark phase

Process of identifying all objects reachable from the root set.

All "live" objects are marked by setting a mark bit in the mark bit vector.

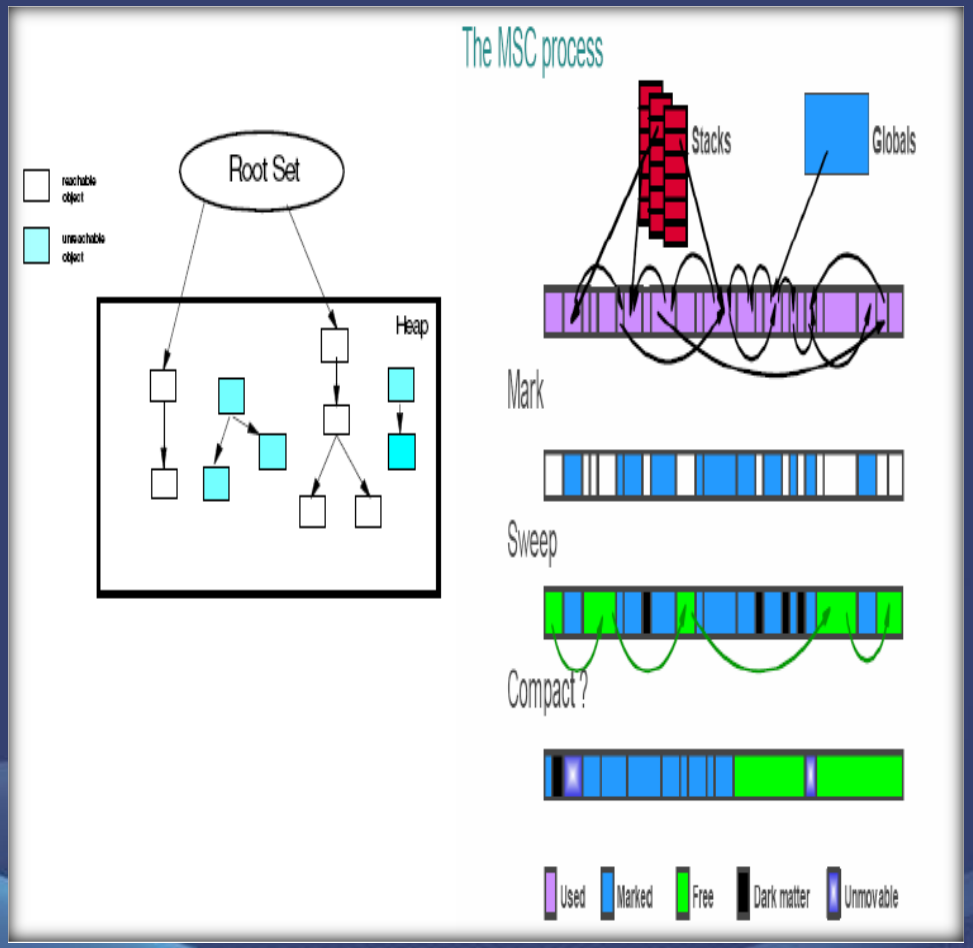
Sweep phase

Sweep phase identifies all the objects that have been allocated,but no longer referenced.

Compaction (optional)

Once garbage has been removed, we consider compacting the resulting set of objects to remove spaces between them.

Release locks and resume threads



Keywords:

It is a reserved word in programming language which is predefined and cannot be used as name for class, method, function and identifier.

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

Variables:

A variable denotes a storage location used to store a data value.

It is defined by the combination of an identifier , a type, and an optional initializer.

Syntax: `type variable-name=value;`

Example: `int a = 80;`

Types of Variable:

- Local variable - A variable which is declared inside the method.
- Instance variable - A variable which is declared inside the class but outside the method and it is not declared as static.
- Static variable - A variable that is declared as static. It cannot be local.

Example for different type of variables:

```
class A{
int data=50; //instance variable
Static int m=100; //static variable
void method(){
int n=90;//local variable
}
} //end of class
```

Data Types:

It is a keyword, represents a kind of data used to allocate sufficient memory space for the data.

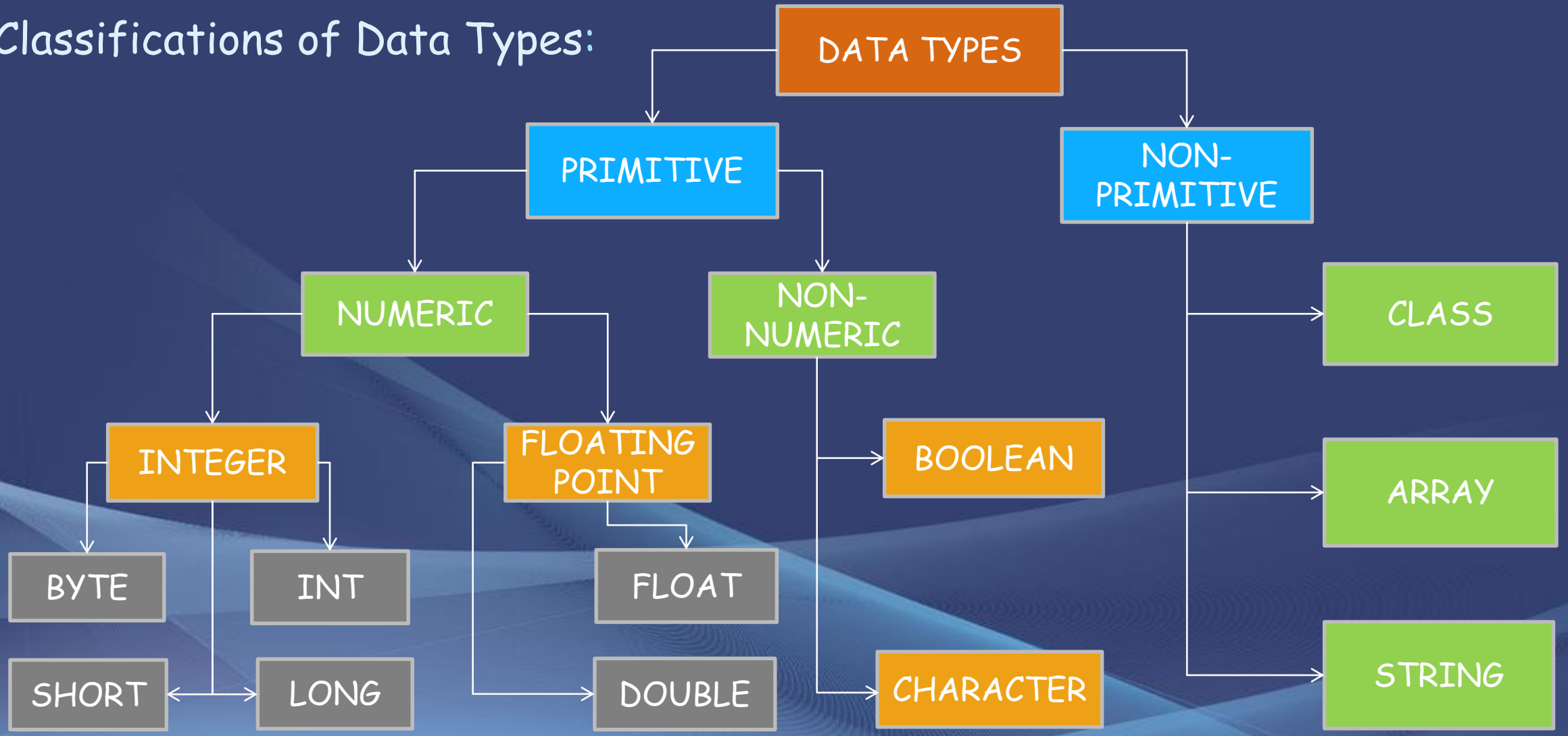
```
int speed;
```

Here, speed is a variable, and the data type of the variable is int.
The int data type determines that the speed variable can only contain integers.

Data types are classified into two types –

- Primitive Data Types
- Non-Primitive Data Types

Classifications of Data Types:



Description of Data Types:

Data Type	Keyword	Kinds of value	Bytes of Memory	Range of values
Byte	byte	Integer	1	-128 to 127
Character	char	1 character - unicode	2	Not applicable
Short integer	short	integer	2	-32,768 to 32,767
Integer	int	integer	4	-2,147,483,648 to 2,147,483,647
Long integer	long	integer	8	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
Float	float	Decimal values to 7 decimal digit precision	4	3.4e-38 to 3.4e38
Double	double	Decimal values to 15 decimal digit precision	8	1.7e-308 to 1.73e308
Boolean	boolean	Boolean values True/false	1	Not applicable

Operators:

Operator in java is a symbol that is used to perform operations.

Types of Operators in Java:

Operator Type	Precedence
Arithmetic Operator	+, -, *, /, %, ++, += , -=, *=, /= ,%= , --
Bitwise Operator	~, &, |, ^, >>, >>>, <<, &=, |=, ^=, >>=, >>>=, <<=
Relational Operator	==, !=, >, <, >=, <=
Logical Operators	||, &&, !
Assignment Operator	=
Ternary Operator	? :

Example for operators:

```
public class ArithmeticDemo {
    public static void main(String[] args) {
        int x = 10;
        int y = 20;
        int result = x + y;
 System.out.println("x + y = " + result); // O/P- x + y =  30
        result = x - y;
 System.out.println("x - y = " + result); // O/P- x - y = -10
        result = x * y;
 System.out.println("x * y = " + result); // O/P- x * y = 200
        result = y / x;
 System.out.println("y / x = " + result); // O/P- y / x = 2
        result = x % 3;
 System.out.println("x % 3 = " + result); // O/P- x + 3 = 1
  }
}
```

Control Statements:

- if statement :
  - 'if' is used to perform conditional checks.
  - It checks boolean condition: true or false.
  - There are various types:
    1. 'if' statement
    2. 'if-else' statement
    3. 'if-else-if' ladder
    4. 'nested if' statement

Syntax:

```
if(condition) {
statement 1;
} else {
statement 2;
}
statement X;
```

Control Statements (Continue..) :

- switch/case : Switch statement is used to execute a particular set of statements among multiple conditions.

Syntax:

```
switch(expression){
case value1:
 //code to be executed;
 break;  //optional
case value2:
 //code to be executed;
 break;  //optional
.....
default:
 code to be executed if all cases are not matched;}
```

- for : for loop is used to execute a set of statements multiple times.
  - There are three types of for loop in java.
    1. Simple for loop
    2. for-each or enhanced for loop
    3. labeled for loop

Syntax:

```
for(initialization; condition; increment/decrement)
{
Statement 1:
}
```

Control Statements (Continue..) :

- while : It is an entry controlled loop. In while, 1st condition will be verified. If the condition is true, loop will be repeated. If the condition is false then control come out of the loop.

- do-while : It is a exit controlled loop. In do-while, 1st all the statements inside the block will be executed and then condition will be verified.

Example:

```
class WhileDemo {
public static void main(String[] args) {
  int i = 1;
  while (i <= 3) {
System.out.println("Num is: " + i);
  i++; } }
} /* O/P -  1
              2
              3   */
```

```
class DoWhileDemo {
public static void main(String[] args) {
  int i = 11;
  do {
System.out.println("Num is: " + i);
  i++;
   } while (i <= 10);
}
}  // O/P - 11
```

Control Statements (Continue..) :

- break – It helps to transfer control to another part of the program.
- continue – It is used when you want to continue running the loop with the next iteration and want to skip the rest of the statements of the body for the current iteration.
- return – It terminates the execution in a method and return the control to the caller method.

Example:

```
public class BreakDemo {
public static void main(String[] args) {
    for(int i=1;i<=10;i++) {
        if(i==4) {
            break; }
        System.out.println(i);
} } }  /* O/P -  1
                  2
                  3   */
```

```
public class BreakDemo {
public static void main(String[] args) {
    for(int i=1;i<=4;i++) {
        if(i==2) {
            continue; }
        System.out.println(i);
} } }  /* O/P -  1
                  3
                  4   */
```

Identifiers :

Identifiers are used for class names, method names, variable names.

An identifier must start with uppercase and lowercase letters, numbers, or the underscore and dollar-sign characters.

Valid Identifiers:	AvgTemp	count	a4	$test	this_is_ok

Invalid Identifiers:	2count	high-temp	Not/ok

Literals:

A constant value in java is created by using a literal representation of it.

100	98.6	'X'	"This is test"

THANK YOU