Introduction to Java

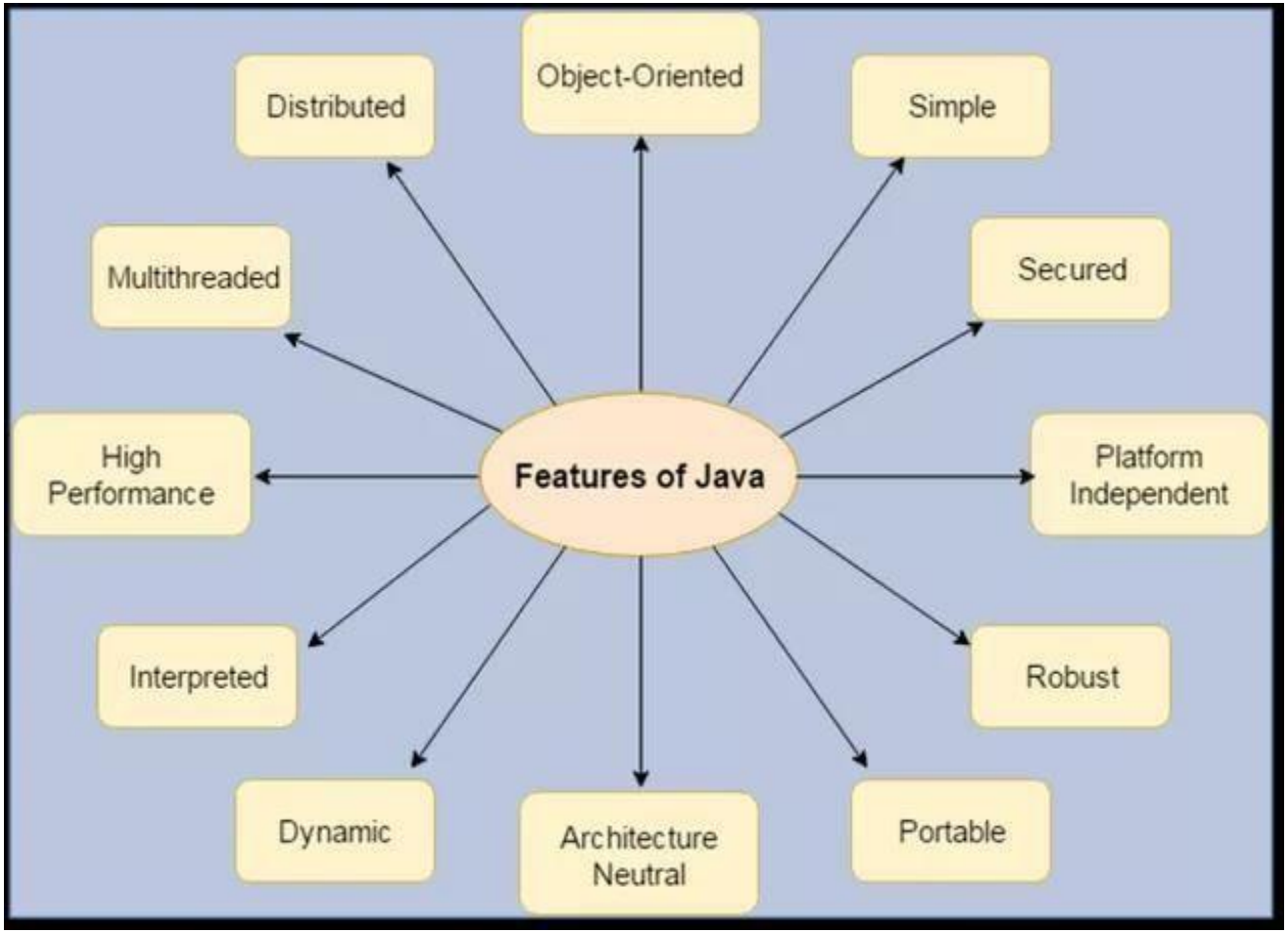
Java History

- There are given the major points that describes the history of java.
- **James Gosling, Mike Sheridan**, and **Patrick Naughton** initiated the Java language project in June 1991. The small team of Sun Microsystem's engineers called **Green Team**.
- Originally designed for small, embedded systems in electronic appliances like set- top boxes.
- Firstly, it was called **"Greentalk"** by James Gosling and file extension was .gt.
- **After that, it was called "Oak" but was renamed as "Java" in 1995.**
- Currently, Java is used in internet programming, mobile devices, games, e-business solutions etc.

Introduction to Java

- Java is one of the most popular and important programming language in the world. Due to its distinct features, it is the most widely used language in every corner of the world and human life. From software and web applications to robotics, Java has become the most wanted language in today's times.

Features of Java

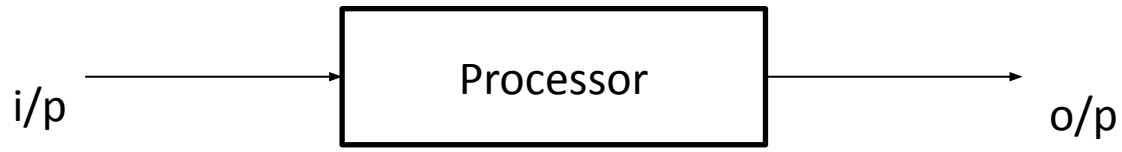


Applications of Java

- Java is not only used in software but is also widely used in designing hardware controlling software components.
- Following are some of the usage of Java :
  - Desktop Applications such as acrobat reader, media player, antivirus etc.
  - Web Applications
  - Mobile Operating System like Android
  - Embedded System
  - Robotics
  - Banking applications
  - Games

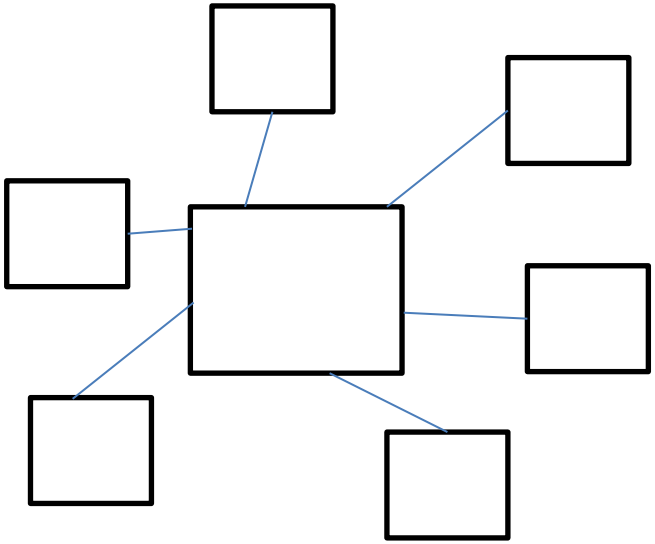
How is java unique?

- For any Programming language

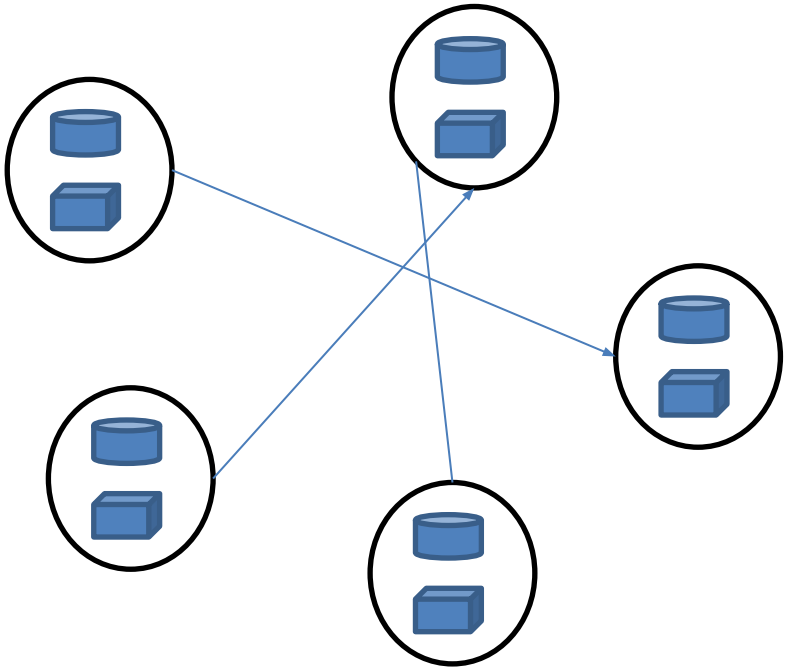


High Level Programming Languages

- Function oriented programming



- Object oriented programming



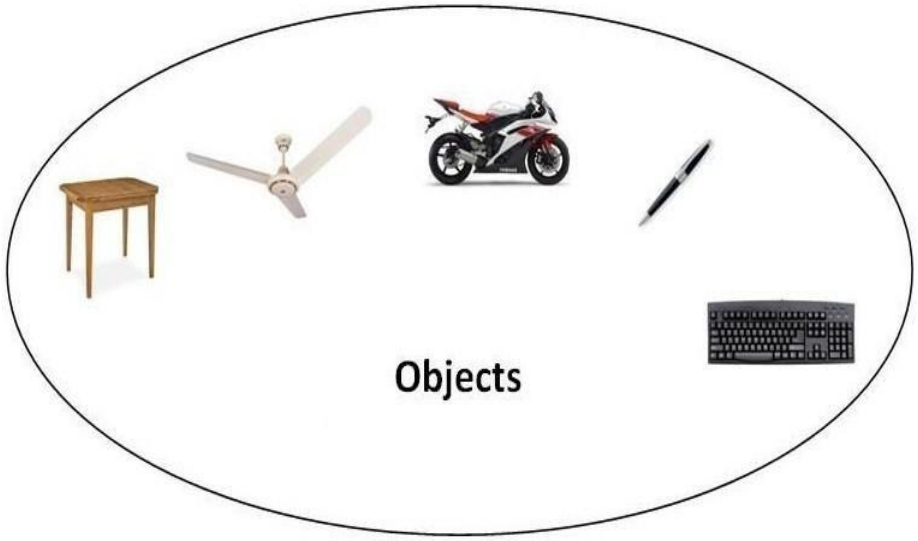
Java Programming Paradigm

- Java is based on the concept of object-oriented Programming.

- Encapsulation
- Inheritance
- Information hiding
- Polymorphism

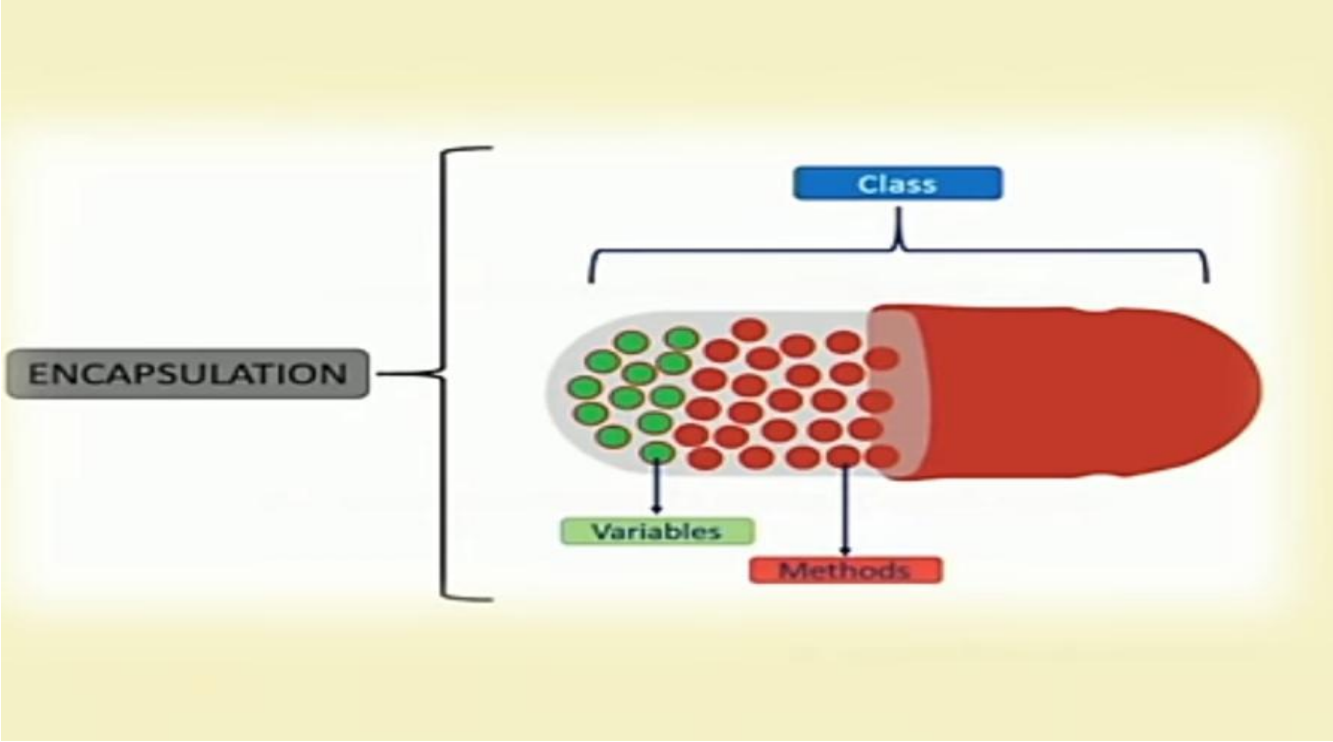
OOPs (Object Oriented Programming System)

- **Object is** a real word entity such as pen, chair, table etc.



Encapsulation

- Encapsulation in java is the process of wrapping code and data into a single unit.



Book

data

Title
Author
Accession No.
Cost
Borrower DOI

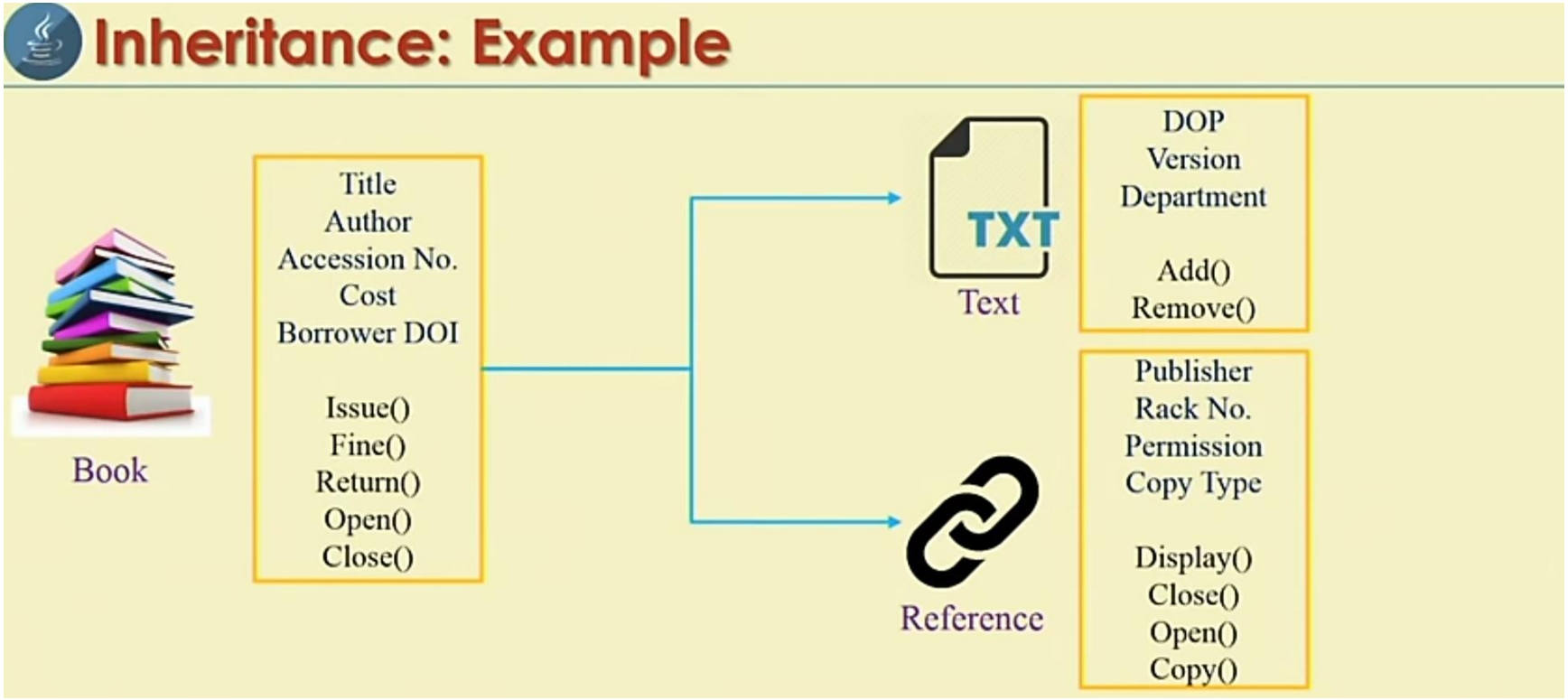
methods

Issue()
Fine()
Return()
Open()
Close()


data
Title
Author
Accession No.
Cost
Borrower DOI
Book
methods
Issue()
Fine()
Return()
Open()
Close()


Books
Borrowers

Inheritance



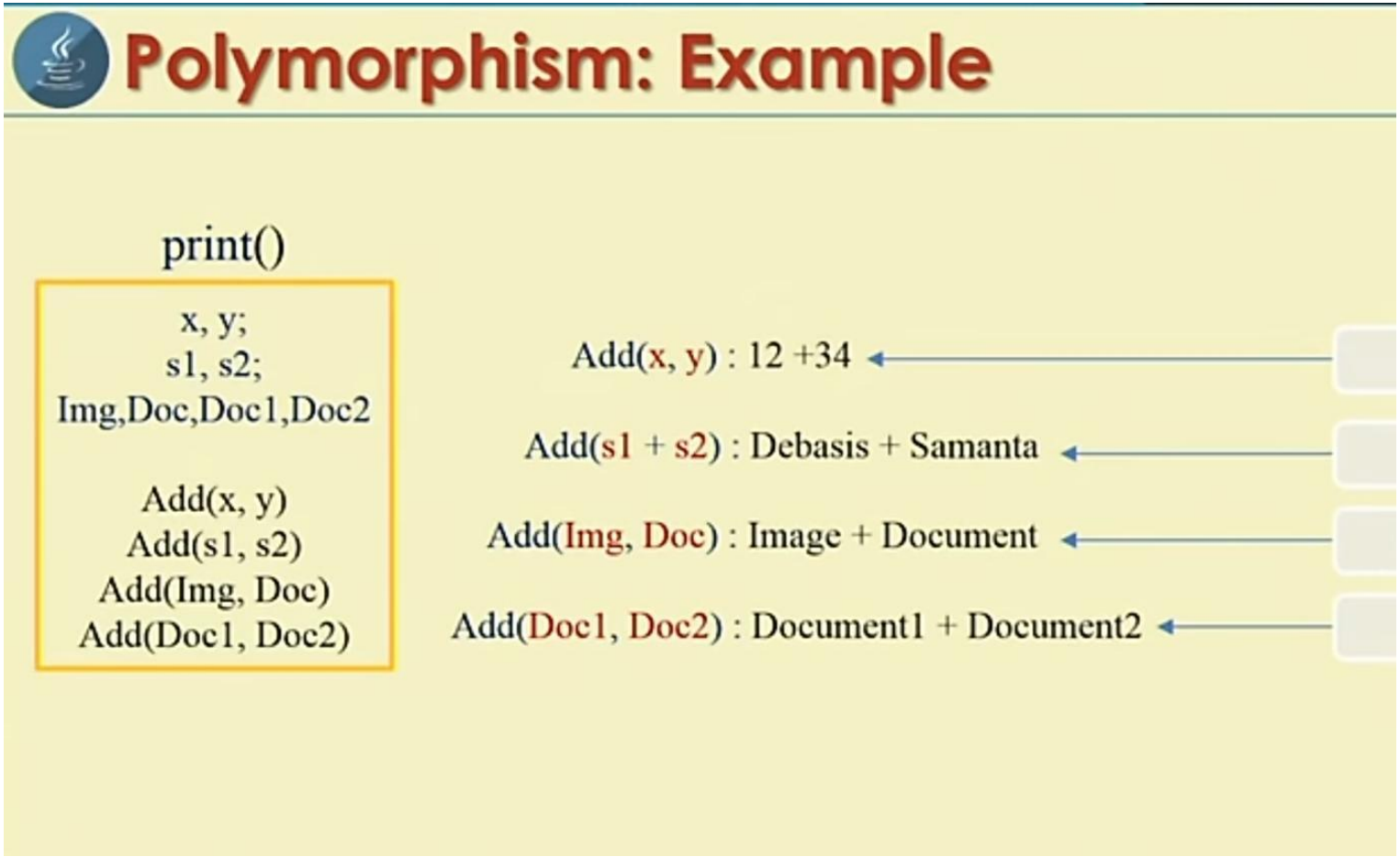
Information hiding

Book

Public
Title
Author
Protected
Account No.
Private
Cost

Public
Issues()
Returns()
Protected
Resave()
Private
Open()
Close()

Polymorphism



Procedural Programming

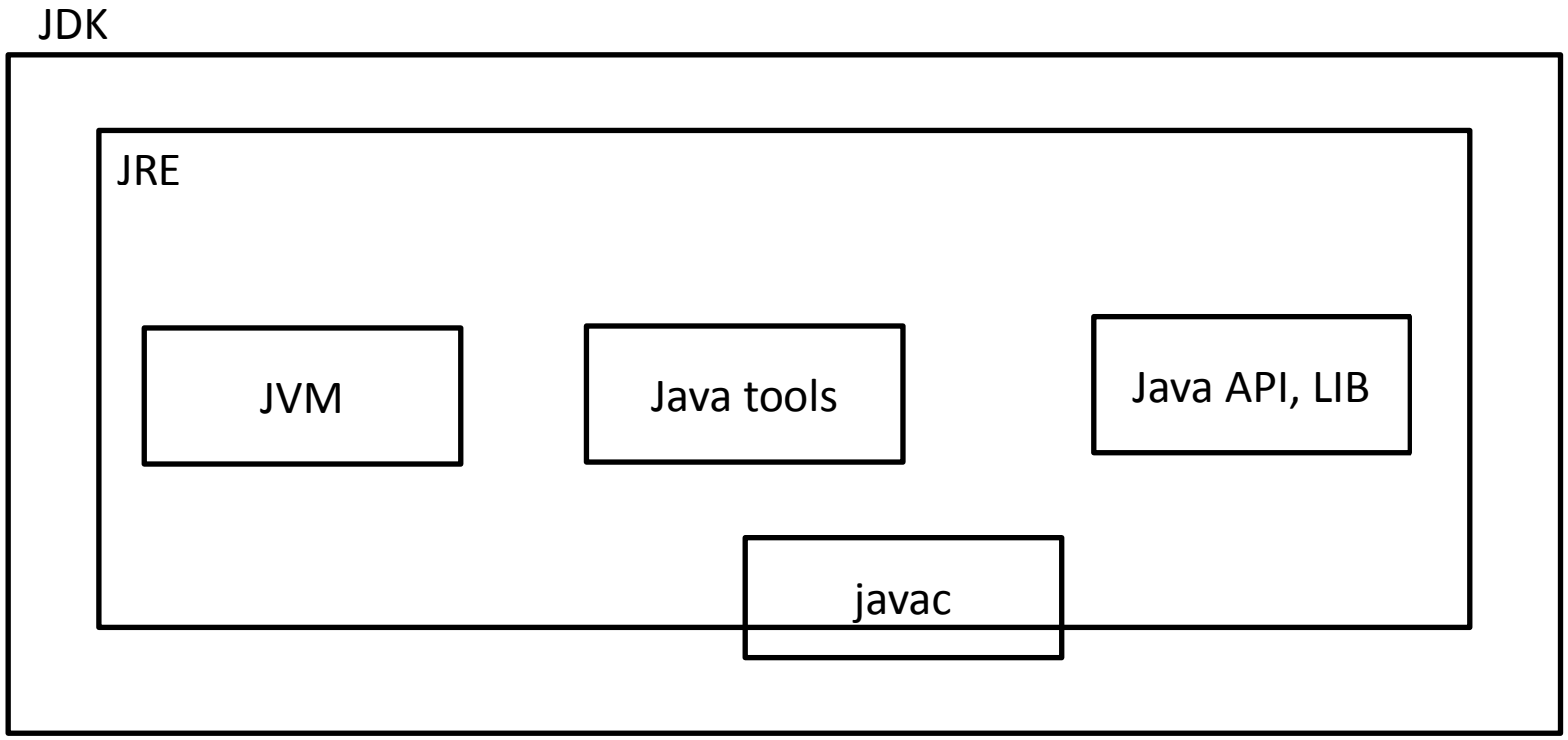
- **Procedural Programming**
  Defined as a programming model which is derived from structured programming, based upon the concept of calling procedure. Procedures, also known as routines, subroutines or functions, simply consist of a series of computational steps to be carried out. During a program's execution, any given procedure might be called at any point, including by other procedures or itself.

C versus Java

Aspects	C	Java
Paradigms	Procedural	Object-oriented
Platform Dependency	Dependent	Independent
Datatypes : union, structure	Supported	Not supported
Pre-processor directives	Supported (#include, #define)	Not supported
Header files	Supported	Use packages (import)
Storage class	Supported	Not supported

Aspects	C	Java
Inheritance	No inheritance	Supported (Simple inheritance)
Pointers	Supported	No Pointers
Code translation	Compiled	Interpreted
Multi-threading and Interfaces	Not supported	Supported
Exception Handling	No exception handling	Supported
Database Connectivity	Not supported	Supported

JVM,JRE, JDK



JDK, JRE and JVM

- **Java Development Kit (JDK)**

  Java Developer Kit contains tools such as compilers needed to develop the Java programs. A Compiler converts Java code into bytecode.

- **Java Virtual machine (JVM)**

  It is a virtual machine that runs the Java bytecodes. The JVM doesn't understand Java source code (the language in which we write Java programs), so first, the source code is compiled into bytecode which can be understood by the JVM. The JVM provides a platform-independent way of executing code, thus making Java platform independent.

- **Java Runtime Environment (JRE)**

  Java Runtime Environment contains JVM, class libraries, and other supporting files. Actually, JVM runs the program and it uses the class libraries and other supporting files provided in JRE. If you want to run any Java program, you need to have JRE installed in the system.

JRE= JVM + Java Package Classes + runtime libraries

Java Version History

- There are many java versions that has been released. Current stable release of Java is Java SE 8.
  1. JDK Alpha and Beta (1995)
  2. JDK 1.0 (23rd Jan, 1996)
  3. JDK 1.1 (19th Feb, 1997)
  4. J2SE 1.2 (8th Dec, 1998)
  5. J2SE 1.3 (8th May, 2000)
  6. J2SE 1.4 (6th Feb, 2002)
  7. J2SE 5.0 (30th Sep, 2004)
  8. Java SE 6 (11th Dec, 2006)
  9. Java SE 7 (28th July, 2011)
  10. Java SE 8 (18th March, 2014)…….

  16. JDK 21

JAVA INSTALLATION

Jdk Installation

1. Open google
2. Type jdk download
3. Java SE 21 -> click on jdk download
4. Goto product -> click the link for Windows x64 Installer
5. Download .exe
6. Open .exe file and install jdk

First Java Program

```
#include<stdio.h>
int main()
{
      printf(“Hello, world”);
      return 0;
}
```

C Program

```
// this is first java program
public class Hello {
    public static void main(String[] args) {
    System.out.println("Hello world");
 }}
```

→ Single line comment

→ Defining the class

→ Defining the main method

Data Types in Java

- Data types specify the different sizes and values that can be stored in the variable. There are two types of data types in Java:
- **Primitive data types:** The primitive data types include boolean, char, byte, short, int, long, float and double.
- **Non-primitive data types:** The non-primitive data types include classes, interfaces and Arrays.

Primitive Data types

Type Name	Description	Size	Range	Sample Declaration & Initialization
boolean	true or false	1 bit	{true, false}	boolean initBool = true;
char	Unicode Character	2 bytes	0-65535	char initChar = 'a';
byte	Signed Integer	1 byte	-128 to 127	byte initInt = 100;
short	Signed Integer	2 bytes	-32768 to 32767	short initShort = 1000;
int	Signed Integer	4 bytes	-2147483648 to 2147483647	int initInt = 100000;
long	Signed Integer	8 bytes	-9223372036854775808 to 9223372036854775807	long initLong = 0;
float	IEEE 754 floating point	4 bytes	±1.4E-45 to ±3.4028235E+38	float initFloat = 10.0f;
double	IEEE 754 floating point	8 bytes	±4.9E-324 to ±1.7976931348623157E+308	double initDouble = 20.0;

https://www.codesdope.com/course/java-data-types/

Java Keywords

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

Identifiers

- **Rules for defining Java Identifiers**
- The only allowed characters for identifiers are all alphanumeric characters([**A-Z**],[**a-z**],[**0-9**]), '**$**'(dollar sign) and '**_**' (underscore).

- Java identifiers are **case-sensitive**.

- There is no limit on the length of the identifier but it is advisable to use an optimum length of 4 – 15 letters only.

- **Reserved Words** can't be used as an identifier.

- These rules are also valid for other languages like C,C++.

Valid and Invalid identifiers

Valid identifiers

MyClass

$amount

totalGrades

TotalGrades

TAX_RATE

one

Invalid identifiers

My Class

numberOf*s

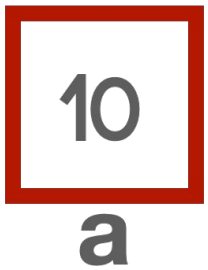
final

1Way

@abc

Variables

- What are Variables?
- A variable is used to store some value. You can think of a variable as a storage which has a name and stores some value.



Variable a storing a value 10

- Declaring variables

type variable_name;

e.g. int x;

String name;

double area;

Initializing Variables

int a=10;

or

int a;

a=10;

char c='x';

Double b=2.33;

```
class Test {
        public static void main(String[] args) {
int n;
                n=5;
                System.out.println(n);
 }}
```

https://www.codesdope.com/course/java-variables/

User input

- We take input with the help of the **Scanner** class.
- Java has a number of predefined classes which we can use.
- Predefined classes are organized in the form of packages.
- This Scanner class is found in the **java.util** package. So to use the Scanner class, we first need to include the **java.util** package in our program.
- Include a package in a program with the help of the **import** keyword.
- A package has many predefined classes.
- We can either import the **java.util.Scanner** class or the entire **java.util** package.
- import java.util.Scanner; // This will import just the Scanner class
- import java.util.*; // This will import the entire java.util package

Accept data from user

```
import java.util.*;
class UserInput {
    public static void main(String[] args)
    {
        Scanner s1 = new Scanner(System.in);
        System.out.println("Enter an integer");
        int a;
        a = s1.nextInt();
        System.out.println("The entered integer is" + a);
    }
}
```

Methods used for taking input of different data types.

Method	Inputs
nextInt()	Integer
nextFloat()	Float
nextDouble()	Double
nextLong()	Long
nextShort()	Short
next()	Single Word
nextLine()	Line of Strings
nextBoolean()	Boolean

https://www.codesdope.com/course/java-input/