ASSIGNMENT INTERVIEW QUESTIONS

Q)what is difference between JDK,JRE and JVM

Q)what is JIT compiler

Q)what is class loader

Q)Explain various memory logical partitions

Q)what gives Java its "write once and run anywhere nature"

Q)Explain History of Java?who invented Java?

Q)what was original name of Java?why it was renamed?

Q)List features of Java

Q)List various Datatypes in Java

Q)what is difference between

System.out.print

System.out.println

System.err.print

Q)How is Java Platform independent

Q)what is bytecode?How is it different from machine code

Q)what is difference between Jar file & Runnable jar file

Q)what is difference between Runnable jar file & exe file

Q)How is C platform dependent language

Q)what is differnce between path & classpath

Explain History of Java?who invented Java?

The history of Java is very interesting. Java was originally designed for interactive television, but it was too advanced technology for the digital cable television industry at the time. 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.

JAVA was named in ten best products of 1995 by the **TIME MAGAZINE**.

Java is an Object-Oriented programming language developed by **James Gosling** in the early 1990s

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

JDK 1.0 was released on January 23, 1996.

Q.)what was original name of Java?why it was renamed?

Firstly, it was called **"Greentalk"** by James Gosling and later became to known as "**OAK**".

Why "Oak"?

The name **Oak** was used by **Gosling** after an **oak tree** that remained outside his office. Also, Oak is an image of strength and picked as a national tree of nations like the U.S.A., France, Germany, Romania, etc.

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

WHY **"JAVA"** ?

Gosling and his team did a brainstorm session and after the session, they came up with several names such as **JAVA, DNA, SILK, RUBY, etc.**

Java is an island in Indonesia where the first coffee was produced (called Java coffee). It is a kind of espresso bean. Java name was chosen by James Gosling while having a cup of coffee nearby his office.

What is Byte Code?

Byte Code can be defined as an intermediate code generated by the compiler after the compilation of source code(JAVA Program). This intermediate code makes Java a platform-independent language.

How is Byte Code generated?

Compiler converts the source code or the Java program into the Byte Code(or machine code), and secondly, the Interpreter executes the byte code on the system. The Interpreter can also be called JVM(Java Virtual Machine). The byte code is the common piece between the compiler(which creates it) and the Interpreter (which runs it).

.class contains bytecode

Interpreter runs file line by line

Q.2) what is JIT Compiler?

The Just-In-Time (JIT) compiler is an essential part of the JRE i.e. Java Runtime Environment,

that is responsible for performance optimization of java based applications at run time.

JIT compilers interact with the Java Virtual Machine (JVM) at run time and compile suitable bytecode sequences into native machine code

the JIT compiler determines the most frequently used code and compiles it to machine code and stores in native cache area.

Some examples of JIT Compilers are JVM (Java Virtual Machine) in Java and CLR (Common Language Runtime), in C#.

Q3. what is class loader?

The **Java ClassLoader** is a part of the **Java Runtime Environment** that dynamically loads Java classes into the **Java Virtual Machine** at runtime.

When the JVM requests a class, the class loader tries to locate the class and load the class definition into the runtime using the fully qualified class name.

The ***java.lang.ClassLoader.loadClass()*** **method is responsible for loading the class definition into runtime**

Classloader subsystem is responsiable for the following 3 activities

1.loading

2.linking :- i.verifiction ii.preparation iii.resolution

3.intialization

Thee different types of class loaders, namely Bootstrap, Extensions, and System class loaders.

Bootstrap serves as a parent for all of them, and is responsible for loading the JDK internal classes.

Extensions and system, on the other hand, load classes from the Java extensions directory and classpath, respectively.

Q4) what gives Java its "write once and run anywhere nature"

In Java, the program is not converted to code directly understood by Hardware, rather it is converted to bytecode(.class file), which is interpreted by JVM, so once compiled it

generates bytecode file, which can be run anywhere (any machine) which has JVM(Java Virtual Machine) and hence it gets the nature of Write Once and Run Anywhere.

This means a programmer can develop Java code on one system and can expect it to run on any other Java-enabled system without any adjustment. This is all possible because of JVM.

Q. what is byte code vs machine code

BYTE CODE	MACHINE CODE
It is an intermediate between source code and Machine code.	It is a low-level language.
Its instructions contain binary and hexadecimal characters with macro commands like new, add, swap, etc.	Its instructions are written in a binary form containing only 0s and 1s.
Byte code requires an interpreter to get executed.	The CPU directly executes machine code.
Byte code can be executed using a virtual machine without requiring a CPU.	Machine code entirely relies on the CPU for its execution.
It is platform-independent and not machine-specific. Byte code can be executed on any virtual machine or software.	It is platform-dependent and machine-specific. Machine code is unique to system architecture and the Operating system.
The entire source code need not be converted into bytecode. The high-level language portion of the source code alone undergoes conversion, which is later transformed into object code for the CPU to execute.	The complete source code undergoes transformation to obtain machine code for CPU execution irrespective of being a high-level or low-level language.

Is java interpreted or compiled?

Java is a hybrid language which is compiled as well as interpreted. When we write a java file we have to compile it using javac

Compiler makes .java file as .class file

This .class file contains bytecode which is platform independent

This bytecode can be taken to any platform

A JVM is used to interpret this bytecode

Q) List features of Java/Why java/Advantages of java

Simple

Java is very easy to learn, and its syntax is simple, clean and easy to understand.

Java has removed many complicated and rarely-used features, for example, explicit pointers, operator overloading, etc.

- There is no need to remove unreferenced objects because there is an Automatic Garbage Collection in Java.

Object-oriented

Java is an object-oriented programming language. Everything in Java is an object. Object-oriented means we organize our software as a combination of different types of objects that incorporate both data and behavior.

Platform Independent

The Java platform differs from most other platforms in the sense that it is a software-based platform that runs on top of other hardware-based platforms. It has two components:

1. Runtime Environment
2. API(Application Programming Interface)

Secured

Java is best known for its security. With Java, we can develop virus-free systems. Java is secured because:

- **No explicit pointer**
- **Java Programs run inside a virtual machine sandbox**

Robust

The English mining of Robust is strong. Java is robust because:

- It uses strong memory management.
- There is a lack of pointers that avoids security problems.
- Java provides automatic garbage collection which runs on the Java Virtual Machine to get rid of objects which are not being used by a Java application anymore.
- There are exception handling and the type checking mechanism in Java. All these points make Java robust.

Architecture-neutral

Java is architecture neutral because there are no implementation dependent features, for example, the size of primitive types is fixed.

Independent of hardware architecture

High-performance

Java is faster than other traditional interpreted programming languages because Java bytecode is "close" to native code. It is still a little bit slower than a compiled language (e.g., C++). Java is an interpreted language that is why it is slower than compiled languages, e.g., C, C++, etc

Multi-threaded

A thread is like a separate program, executing concurrently. We can write Java programs that deal with many tasks at once by defining multiple threads. The main advantage of multi-threading is that it doesn't occupy memory for each thread. It shares a common memory area. Threads are important for multi-media, Web applications, etc.

Dynamic

Java is a dynamic language. It supports the dynamic loading of classes. It means classes are loaded on demand.

Application of java

```
Java Applications:
-------------------
-Desktop applications
    -stand-alone applications
    -antivirus, VLc media player, etc...

-Web-based applications
    -client server architecture based application.
    -amazon, railway station,etc...
    -Servlet, JSP(Java server pages), Structs, Spring, Hibernate, JSF, etc...

-Enterprise applications
    -distributed in nature,banking,
    -load bancing, clustering, high level security,etc
    -EJB (Enterprise Java Beans)
    -Scientific applications

-Mobile applications
    -Micro editions of java
    -Gaming applications
    -Android, Java ME

-Cloud applications
```

```
Types of Java application:
--------------------------
1. Stand alone application:
    -Runs on our machine.
    -stand-alone application.
    -Compiler and interpreser will be used.
    -read & write program file

2. Web applet application:
    -Runs on browser.
    -Internet applications.
    -downloaded from server and execute on browser.
    -execute the applet.
```

What are platform editions of java?

There are 3 Java Platform Editions:

1. Java 2 Platform, Standard Edition (J2SE)

• Core Java Platform targeting applications running on workstations

2. Java 2 Platform, Enterprise Edition (J2EE)

• Component-based approach to developing distributed, multi-tier enterprise

applications

3. Java 2 Platform, Micro Edition (J2ME)

• Targeted at small, stand-alone or connectable consumer and embedded devices

Q) List various Datatypes in Java

There are two types of data types in Java:

1. **Primitive data types:** The primitive data types include boolean, char, byte, short, int, long, float and double.
2. **Non-primitive data types:** The non-primitive data types include Classes, Interfaces, and Arrays.

Boolean Data Type

The Boolean data type is used to store only two possible values: true and false. Default value is false

"size" can't be defined

Byte Data Type

The byte data type is an example of primitive data type. It isan 8-bit signed two's complement integer. Its value-range lies between -128 to 127 (inclusive). Its minimum value is -128 and maximum value is 127. Its default value is 0.

It can also be used in place of "int" data type.

Short Data Type

The short data type is a 16-bit signed two's complement integer. Its value-range lies between -32,768 to 32,767 (inclusive). Its minimum value is -32,768 and maximum value is 32,767. Its default value is 0.

Int Data Type

The int data type is a 32-bit signed two's complement integer. Its value-range lies between -2,147,483,648 (-2^31) to 2,147,483,647 (2^31 -1) (inclusive). Its minimum value is - 2,147,483,648and maximum value is 2,147,483,647. Its default value is 0.

Long Data Type

The long data type is a 64-bit two's complement integer.

Its default value is 0. The long data type is used when you need a range of values more than those provided by int.

Float Data Type

The float data type is a single-precision 32-bit IEEE 754 floating point.Its value range is unlimited. It is recommended to use a float (instead of double) if you need to save memory in large arrays of floating point numbers. The float data type should never be used for precise values, such as currency. Its default value is 0.0F.

Double Data Type

The double data type is a double-precision 64-bit IEEE 754 floating point. Its value range is unlimited. The double data type is generally used for decimal values just like float. The double data type also should never be used for precise values, such as currency. Its default value is 0.0d.

Char Data Type

The char data type is a single 16-bit Unicode character. Its value-range lies between '\u0000' (or 0) to '\uffff' (or 65,535 inclusive).The char data type is used to store characters.

Q) what is difference between

System.out.print - normally outputs the data you write to it to the console and keeps cursor on same line.

System.out.println - outputs the data you write to it to the console and moves the cursor to next line.

System.out.printf – printf is used and then format specifier is used

System.out.**System.err** is also a print stream. It works the same as System.out. It is mostly used to output error texts.

What is JDK?

The Java Development Kit (JDK) is a software development environment which is used to develop java applications and applets.

It contains JRE + development tools.

JDK contains:

- Java Runtime Environment (JRE),
- An interpreter/loader (Java),
- A compiler (javac),
- An archiver (jar) and many more.

What is jar file?

Java archive file

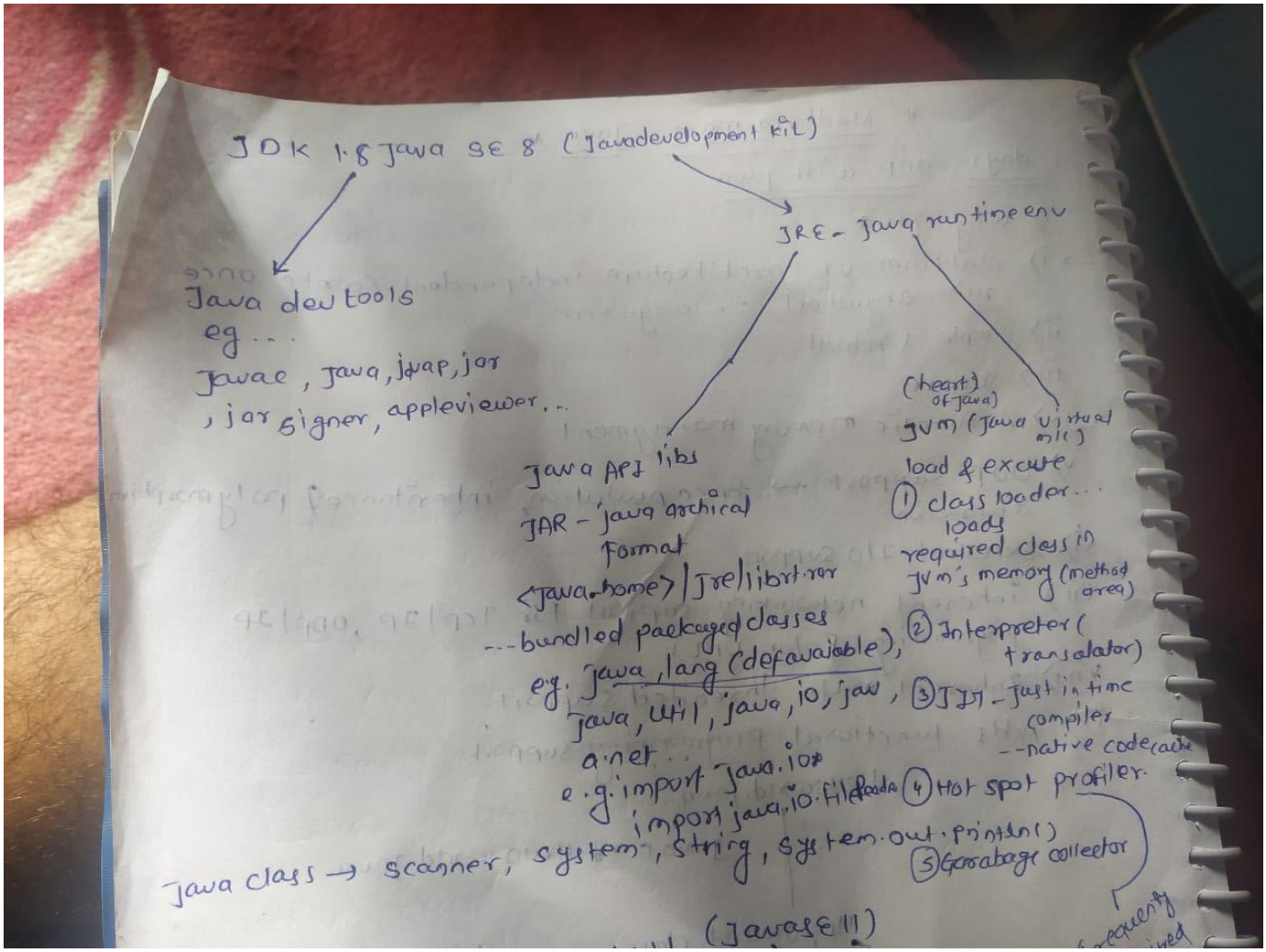
The jar file is nothing but a full pack of Java classes. After creating the .class files, you can put them together in a .jar, which compresses and structures them in a predictable fashion.

JDK VS JRE VS JVM

JDK (Java Development Kit) is a Kit that provides the environment to **develop and execute(run)** the Java program. JDK is a kit(or package) that includes two things

- Development Tools(to provide an environment to develop your java programs)
- JRE (to execute your java program).
- **JRE** (Java Runtime Environment) is an installation package that provides an environment to **only run(not develop)** the java program(or application)onto your machine. JRE is only used by those who only want to run Java programs that are end-users of your system.

3. JVM (Java Virtual Machine) is a very important part of both JDK and JRE because it is contained or inbuilt in both. Whatever Java program you run using JRE or JDK goes into JVM and JVM is responsible for executing the java program line by line, hence it is also known as an ***interpreter*.**



What is JVM?

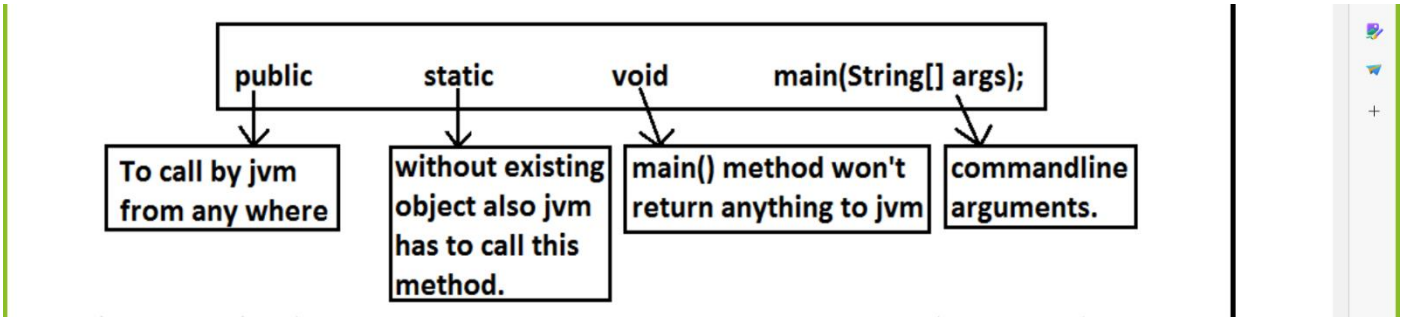
Refer above diagram

Can we run bytecode of one os on another os?

As jvm is platform specific we cannot run it on other system.

We need JVM on system for the bytecode to run(interpreter does the translation line by line)

WHAT IS public static void main(String[] args) ?



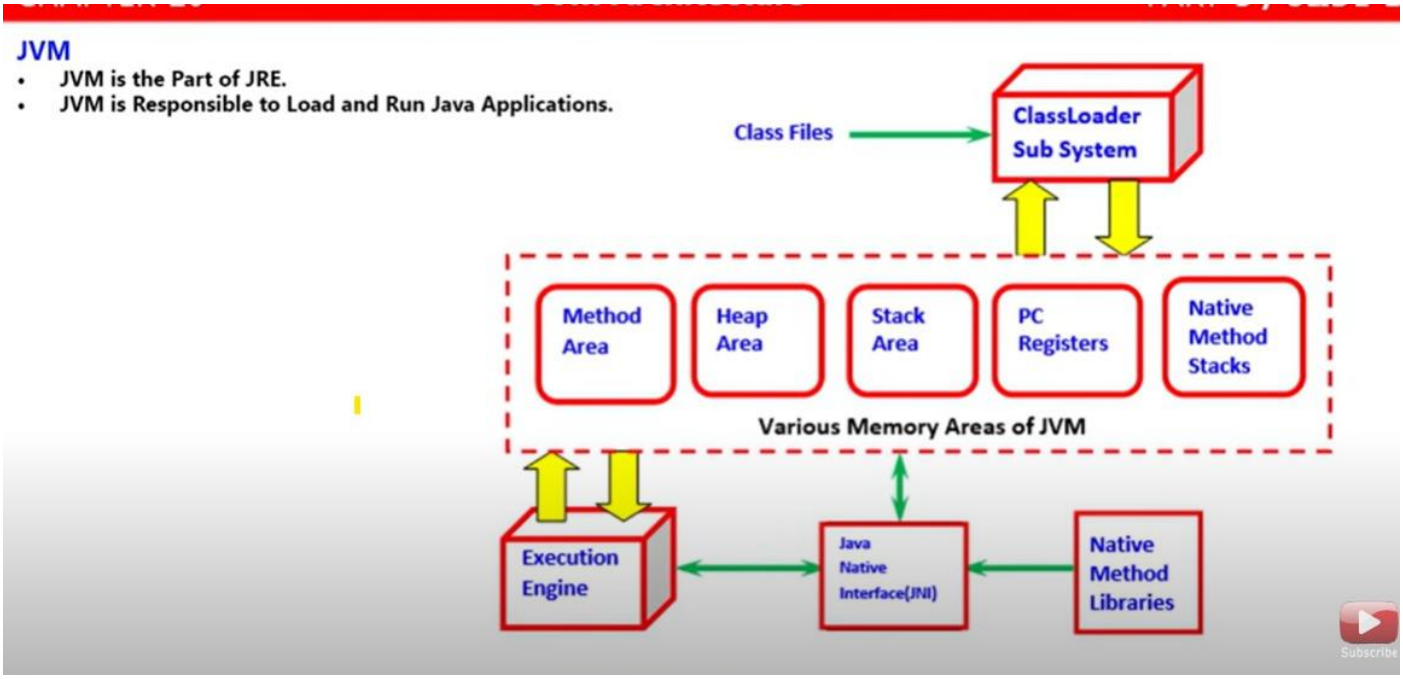
TYPES OF STATEMENTS(control flow) IN JAVA

Selective statements – if else, Switch

Transfer statements – break, continue

Iterative statements – for, while, do while, for each

JVM ARCHITECTURE



JVM Memory

1. **Method area:** In the method area, all class level information like class name, immediate parent class name, methods and variables information etc. are stored, including static variables. There is only one method area per JVM, and it is a shared resource. From java 8, static variables are now stored in Heap area.
2. **Heap area:** Information of all objects is stored in the heap area. There is also one Heap Area per JVM. It is also a shared resource.
3. **Stack area:** For every thread, JVM creates one run-time stack which is stored here. Every block of this stack is called activation record/stack frame which stores methods calls. All local variables of that method are stored in their corresponding frame. After a thread terminates, its run-time stack will be destroyed by JVM. It is not a shared resource.
4. **PC Registers:** Store address of current execution instruction of a thread. Obviously, each thread has separate PC Registers.
5. **Native method stacks:** For every thread, a separate native stack is created. It stores native method information.

What is class?

Class is a user defined template or blueprint or prototype from which objects are created.

Class is a group of fields(variable) and methods(functions)
Class is not a real world entity.

It is a logical entity

Class does not occupy memory.

A class in java can contain:
• data member
• method
• constructor
• nested class and
• interface

```
Syntax to declare a class:

access_modifier class<class_name>

{

    data member;

    method;

    constructor;

    nested class;

    interface;

}
```

Eg:
• Animal
• Student
• Bird
• Vehicle
• Company

OBJECT

It is a basic unit of Object-Oriented Programming and represents real-life entities. A typical Java program creates many objects, which as you know, interact by invoking methods. An object consists of :

1. **State**: It is represented by attributes of an object. It also reflects the properties of an object.
2. **Behavior**: It is represented by the methods of an object. It also reflects the response of an object with other objects.
3. **Identity**: It gives a unique name to an object and enables one object to interact with other objects.

Example of an object: dog

Objects correspond to things found in the real world. For example, a graphics program may have objects such as “circle”, “square”, and “menu”. An online shopping system might have objects such as “shopping cart”, “customer”, and “product”.

Declaring Objects (Also called instantiating a class)

When an object of a class is created, the class is said to be **instantiated**

The new operator instantiates a class by allocating memory for a new object and returning a reference to that memory. The new operator also invokes the class constructor.

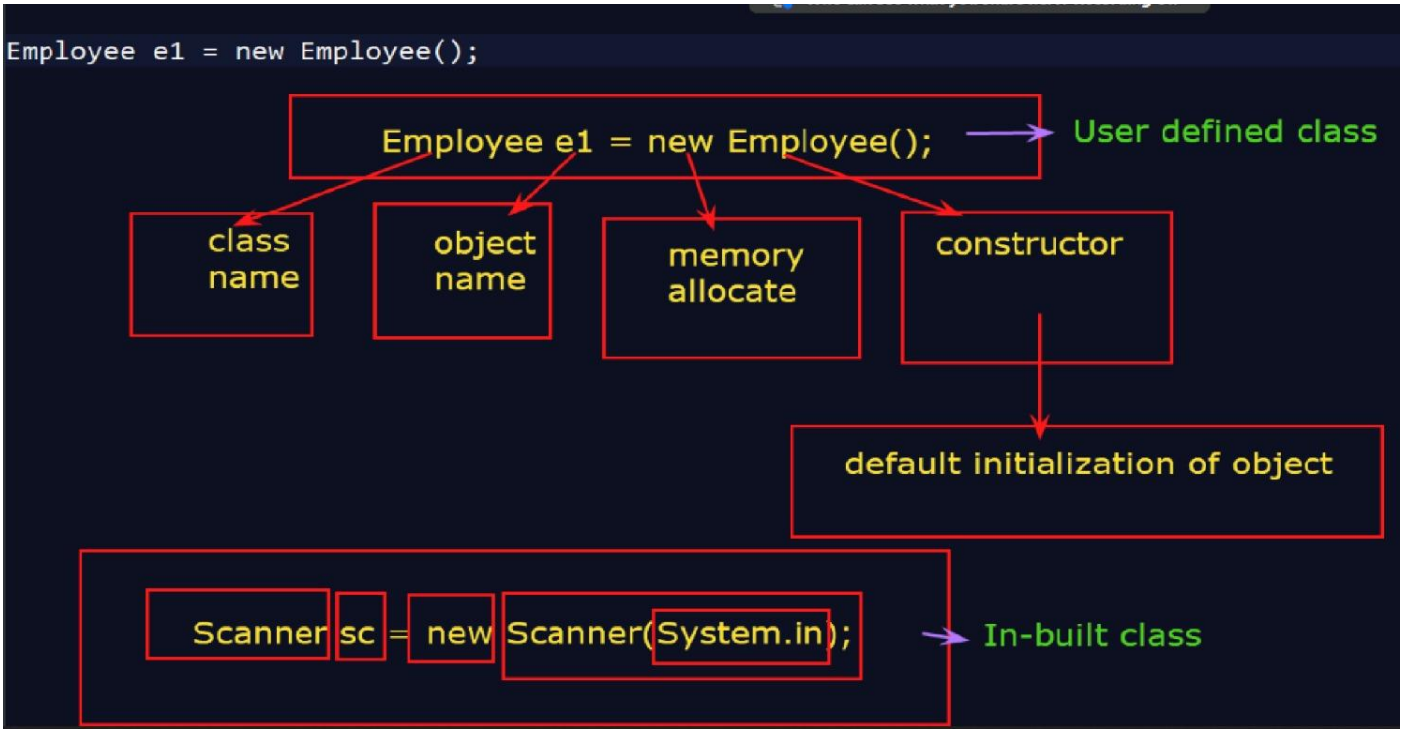
```
// creating object of class Test
Test t = new Test();
```

Test = class name

t = object name

new = memory allocated

Test() = constructor(to initialize values)



What is constructor ?

A constructor in Java is a **special method** that is used to initialize objects. The constructor is called when an object of a class is created. It can be used to set initial values for object attributes.

Note that the constructor name must **match the class name**, and it cannot have a **return type** (like `void`).

All classes have constructors by default: if you do not create a class constructor yourself, Java creates one for you. However, then you are not able to set initial values for object attributes.

What are fields?

fields are variables that provides the state of the class & it's objects

what are methods?

methods are used to implement the bheaviour of the class & it's objects

Tell something about OOPS

ADD ONE REAL LIFE EXAMPLE TO THIS

Object-Oriented Programming or OOPs refers to languages that use objects in programming, they use objects as a primary source to implement what is to happen in the code.

The main aim of OOP is to bind together the data and the functions that operate on them so that no other part of the code can access this data except that function.

Data Abstraction is the property by virtue of which only the essential details are displayed to the user. The trivial or the non-essential units are not displayed to the user. Ex: A car is viewed as a car rather than its individual components.
Data Abstraction may also be defined as the process of identifying only the required characteristics of an object ignoring the irrelevant details.

Consider a real-life example of a man driving a car. The man only knows that pressing the accelerators will increase the speed of a car or applying brakes will stop the car, but he does not know how on pressing the accelerator the speed is actually increasing, he does not know about the inner mechanism of the car or the implementation of the accelerator, brakes, etc in the car. This is what abstraction is.

1. Provides a clear and simple interface to the user.
2. Increases security by preventing access to internal class details.
3. It reduces the complexity of viewing things.
4. Avoids code duplication and increases reusability.
5. Helps to increase the security of an application or program as only essential details are provided to the user.
6. It improves the maintainability of the application.

Encapsulation is a fundamental concept in object-oriented programming (OOP) that refers to the bundling of data and methods that operate on that data within a single unit, which is called a class in Java. Encapsulation is a way of hiding the implementation details of a class from outside access and only exposing a public interface that can be used to interact with the class.

In Java, encapsulation is achieved by declaring the instance variables of a class as private, which means they can only be accessed within the class.

Encapsulation is defined as the wrapping up of data under a single unit. It is the mechanism that binds together code and the data it manipulates. Another way to think

about encapsulation is, that it is a protective shield that prevents the data from being accessed by the code outside this shield.

Advantages of Encapsulation:

Data Hiding

Increased flexibility

Reusability

1. Improves security of an object's internal state by hiding it from the outside world.
2. Increases modularity and maintainability by making it easier to change the implementation without affecting other parts of the code.

Inheritance is an important pillar of OOP(Object-Oriented Programming). It is the mechanism in java by which one class is allowed to inherit the features(fields and methods) of another class. In Java, inheritance means creating new classes based on existing ones. A class that inherits from another class can reuse the methods and fields of that class. In addition, you can add new fields and methods to your current class as well.

Inheritance in Java: Why do we need it?

- **Code Reusability:** The code written in the Superclass is common to all subclasses. Child classes can directly use the parent class code.
- **Method Overriding:** Method Overriding is achievable only through Inheritance. It is one of the ways by which java achieves Run Time Polymorphism.
- **Abstraction:** The concept of abstract where we do not have to provide all details is achieved through inheritance. Abstraction only shows the functionality to the user.

polymorphism as the ability of a message to be displayed in more than one form.

Polymorphism allows us to perform a single action in different ways. In other words, polymorphism allows you to define one interface and have multiple implementations. The word "poly" means many and "morphs" means forms, So it means many forms.

In Java polymorphism is mainly divided into two types:

- Compile-time Polymorphism
- Runtime Polymorphism

Type 1: Compile-time polymorphism

It is also known as static polymorphism. This type of polymorphism is achieved by function overloading or operator overloading.

Type 2: Runtime polymorphism

It is also known as Dynamic Method Dispatch This type of polymorphism is achieved by Method Overriding.

Advantages of Polymorphism in Java:

1. Increases code reusability by allowing objects of different classes to be treated as objects of a common class.
2. Improves readability and maintainability of code by reducing the amount of code that needs to be written and maintained.

3. Supports dynamic binding, enabling the correct method to be called at runtime, based on the actual class of the object.
4. Enables objects to be treated as a single type, making it easier to write generic code that can handle objects of different types.