1. **What are static members in java?**

Ans:

In Java, **static members** are **class-level members** that belong to the **class itself** rather than any specific object. They are declared using the static keyword. They can be called without creating an object of the class and cannot access non-static (instance) members directly.

1. **What is static block?**

Ans:

Static block is used for initializing static variables. They execute once when the class is loaded.

1. **Can static member function access non-static data ?**

**Ans:**

**No**. because static member function can be called without creating object. When object is not created, non-static member is not allocated memory.

**4)Can non-static member function access static data ?**

Ans:

**Yes**. Because in order to invoke non-static member function u need to create object and by that time static members are already allocated memory

1. **What is non-static block in java?**

Ans:

A **non-static block** (also known as an **instance initializer block**) is a block of code inside a class that executes **whenever an object is created**. It is used for **initializing instance variables** before the constructor runs.

1. What is this? And what is its importance in object oriented programming?

Ans:

"this" is a reference which refers to the current or invoking object.

when u create multiple objects, u have those many copies of instance members however , u have only one copy of member function/s. In that case how will member function keep a track of invoking object ?

member function/s will come to know about invoking or current object through "this" reference.

1. **Why can’t we use “this” with static members?**

Ans:

Because ,“this” refers to object and static are not associated with objects.

1. **What happens when we say**

**java Sample [ consider “Sample” class contains “main” function ]**

ans:

a) JVM searches "Sample.class" inside current working directory

if found, "Sample.class" will be loaded by JVM.

b) JVM will invoke "main()" function.

since main() function is static, JVM invokes it with

Sample.main

1. **Explain System.out.println("hello");**

Ans:

System.out.println("hello")

class System

{

public static PrintStream out;

}

class PrintStream

{

public void print(){}

public void println(){}

}

explanation:

a) we want to print "hello"

so we write ("hello")

b) using println() method we can print "hello"

so we write println("hello");

c) println() is a non-static method of PrintStream , hence it needs to be called using reference of PrintStream.

d) do we have any reference of PrintStream available?

yes it's "out"

so we write out.println("hello");

e) what is "out"?

"out" is a public static member of "System" class, hence we write

System.out

so that's why we have

System.out.println("hello");

1. **What is class "Class" in java ?**

Ans:

In java whenever any class gets loaded it is represented by instance of class "Class".

This instance holds information about loaded class such as member variables, methods,constructors etc.

1. **Where exactly in java code we can use class "Class"?**

Ans: we use class "Class" in case of Reflection API.

1. What is the use of non-static block in java?

Ans:

NON-STATIC BLOCK:is used for - if we have many constructors inside a class and those constructors need to have some common statements.

instead of repeate those statements in each constructor,we place those statements in non-static block.

e.g counter which is incremented in each constructor , to keep a track of number of objects created.

1. **When a class gets loaded ?**

Ans:

a) implicitly

java MyClass8

or

emp e=new emp(); when u create first object of that class

or

emp.staticmemberfun();

or

before deserialization if the class is not loaded it gets loaded implicitly during deserialization

b) explicitly

Class.forName("classname");

1. **What is NullPointerException?**

Ans:

When a reference contains null and if u invoke a method on it , u get NullPointerException.

1. **Explain Singleton class.**

Ans:

is a class where :

a) only one object is created and that too inside the class itself by the developer.

b) users are not allowed to create the object or instance of the class. They can use the same instance or object which has been created inside the class.

c) may have some non-static methods.

d) developer must share one and only one object created of the class among all the users , so that they can invoke non-static methods of the class.

1. **How many places final keyword can be applied in java?**

Ans:

final keyword can be applied to

a) instance member

b) class variable

c) local variable

d) member function

e) class

1. What is the meaning of constructor?

Ans:

It is a special member function .

Special because

a) it is used to initialize (construct) the instance member/s.

b) it has got the same name as of class

c) it does not have a return type.

1. **How many types of constructors are allowed in Java?**

Ans:

There are 2 types of constructor

a) default or no-arg constructor

constructor having no arguments.

if no constructor is defined in the class, compiler by default provides "no-arg constructor". But the moment we define some constructor in the class, compiler does not provide "default or no-arg constructor".

b) parameterized constructor

constructor with at least one argument.

1. **What happens when object is created?**

Ans:

a) memory is allocated for instance member/s.

b) constructor is called

1. **Does java have destructor?**

Ans:

java does not have destructor.

just before object gets garbage collected, following method gets called. Programmer can override(define) this finalize method in order to release resources such as file, database connection or sockets.

protected void finalize()

{

}

Since there is no guarantee as to when exactly object will get garbage collected, u can not rely upon "finalize" method to release the resources such as Connection , Socket etc.

Garbage collection can not be forced in java. U can just make a request for Garbage Collection , by invoking a method "System.gc()". or "Runtime.getRuntime().gc()".

1. **Explain abstraction with the real life example.**

Ans:

Abstraction means Hiding an implementation and showing only essential features.

It means providing information about what object does instead of how it does.

**Real-Life Examples of Abstraction**

1. **Car Operation**:
   * **Abstraction**: When you drive a car, you interact with the steering wheel, pedals, and gear shift. You don’t need to know the intricate details of how the engine, transmission, or braking system works internally.
   * **Underlying Complexity**: The car's internal systems are complex, but the user interface (steering wheel, pedals) abstracts away this complexity.
2. **Television Remote Control**:
   * **Abstraction**: A TV remote provides buttons for changing channels, adjusting volume, and powering the TV on/off. The remote abstracts the complex operations of the TV and its internal mechanisms, allowing users to perform these actions without understanding how they are executed internally.
   * **Underlying Complexity**: The remote communicates with the TV using signals to control various functions, but the user interacts with a simplified interface.
3. **Explain encapsulation with the real life example.**

Ans:

**Binding the data and functions together in one unit is called as "Encapsulation".**

In java "class" is the example of Encapsulation.

e.g.

class Car

{

**private** color,weight,price,model etc. // data

**public** start() // member function

{

}

**public** stop() // member function

{

}

**public** accelerate() // member function

{

}

}

one more example:

class TV

{

**private** wires,circuits,TV Tube etc. // data

**public** on() // member functions

{

}

**public** off() // member functions

{

}

}

**In a Car**: The car’s internal systems are encapsulated within its body. This means the driver does not interact directly with the internal mechanisms but instead uses controls like the steering wheel, pedals, and gear shift. These internal systems (engine, transmission, brakes) are managed and operated internally by the car's design.