Java

Java Architecture - JDK, JVM and JRE(platform dependencies)
Variables, Operators, Control Flow statements, Expressions
Bitwise and Ternary
Java treats null and zero differently
Method Overloading
Parameter passing-static, object, variables
Static Class, Static Block
Nested classes
Encapsulation
Constructor
Destructor
Exception Handling
Exception Hierarchy
Object Class
I/O streams

Do all versions of Java support Static Block
Popular methods that have been removed from Java 11 and so on
Exception handling v/s error
Predefined exception in Exception class and Throwable class
All are derived at run time
Checked exception vs unchecked exception
JVM always creates an object for an exception and throws It
This has to be caught using try and except block
finally block is optional (you can do some custom things here)
Is final method inherited?
What is blank or uninitialized final variable?
Can we initialize blank final variable?
Can we declare a constructor final?
Can we overload java main() method?
Why multiple inheritance is not supported in java?
Does constructor return any value?
Why is the Java main method static?
Can we execute a program without main() method?
while using super.method() it calls parent, but that can only be done in the method of the child class. So both parent and then child function get executed. Is there a way to just make the parent function get executed an return to the original function call? (REDUNDANT NOT PRACTICAL)

FINAL MODIFIER IN A CLASS
The final modifier for finalizing the implementations of classes, methods, and variables. The main purpose of using a class being declared as final is to prevent the class from being subclassed. If a class is marked as final then no class can inherit any feature from the final class. You cannot extend a final class.

Introduction

Java is a programming language and platform
High level, robust, OO oriented and secure
James Gosling is father of Java

Why is Java called a platform?
Platform: Any hardware or software environment in which a program runs, is known as a platform. Since Java has a **runtime environment (JRE)** and API, it is called a platform.
Java Application
4 types:

1) **Standalone**
2) **Web**
3) **Enterprise**
4) **Mobile**

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

1) Inheritance
2) Polymorphism
3) Abstraction
4) Encapsulation

A platform is the hardware or software environment in which a program runs.
There are two types of platforms software-based and hardware-based. Java provides a software-based platform.
It has two components:

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

Java code can be run on multiple platforms, for example, Windows, Linux, Sun Solaris, Mac/OS, etc.
Java code is compiled by the compiler and converted into **bytecode**.
This **bytecode is a platform-independent code** because it can be run on multiple platforms, i.e., Write Once and Run Anywhere(WORA).

Java is secured because:

1) No explicit pointer
2) Java Programs run inside a virtual machine sandbox
3) Bytecode Verifier
4) Security Manager

There is automatic garbage collection in java which runs on the Java Virtual Machine to get rid of objects which are not being used by a Java application anymore.

Java supports dynamic compilation and automatic memory management (garbage collection).
Does not support

1) goto statement.
2) multiple inheritance through class. It can be achieved by interfaces in java.
3) operator overloading.
4) has restricted pointer support in java.
5) default arguments like C++.
6) Java does not support header files like C++.(uses import to include classes

JDK, JVM and JRE

JVM
Abstract machine
Doesn't physically exist
run time environment in which Java bytecode is executed
Run programs written in other languages and convert to Bytecode

JVM, JRE, and JDK are platform dependent because the **configuration of each OS is different** from each other. However, **Java is platform independent.**

JVM provides definitions for the:
1. Memory area
2. Class file format
3. Register set
4. Garbage-collected heap
5. Fatal error reporting etc.

The JVM performs the following main tasks:
1. Loads code
2. Verifies code
3. Executes code
4. Provides runtime environment

JAVA RUNTIME ENVIRONMENT
JRE
Set of software tools for developing java applications.
Provides Run time environment
Implementation of JVM
Physically Exists
Contains a set of libraries + other files that JVM uses at run time

Java Development Kit(JDK)
software development environment which is used to develop Java applications and applets.
It physically exists.
It contains JRE + development tools.

Variables, Operators, Precedence, Unknown Loops

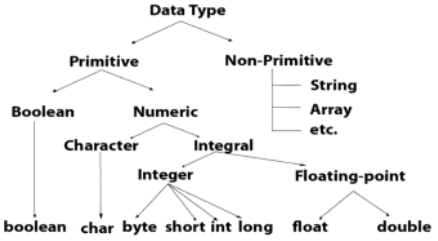
Java Variables
container which holds the value while the Java program is executed.
A variable is assigned with a data type.
Variable is a name of memory location.
local, instance and static.
primitive and non-primitive.

1. instance variable:
   instance specific and is not shared among instances.

2. static variable:
   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.**

Operations on variables
1. Widening: making it store more data,
2. Type Casting: converting to different type of data(especially smaller type)
3. Overflow
4. Adding lower type

Byte overflow
If the **result is greater than 127 or less than -128,** then **the byte variable overflows** (i.e., it cannot contain the resulting value in a single byte). The **remainder result is then displayed instead of the original result.**



Data Type	Default Value	Default size
boolean	false	1 bit
char	'\u0000'	2 byte
byte	0	1 byte
short	0	2 byte
int	0	4 byte
long	0L	8 byte
float	0.0f	4 byte
double	0.0d	8 byte

Why char uses 2 byte in java and what is \u0000 ?
It is because java uses Unicode system not ASCII code system. The \u0000 is the lowest range of **Unicode system.**

Unicode System

Unicode is a universal international standard character encoding that is capable of representing most of the world's written languages.

In unicode, character holds 2 byte, so java also uses 2 byte for characters.
lowest value:\u0000

highest value:\uFFFF

Operators

Operator Type	Category	Precedence
Unary	postfix	expr++ expr--
	prefix	++expr --expr +expr -expr ~ !
Arithmetic	multiplicative	* / %
	additive	+ -
Shift	shift	<< >> >>>
Relational	comparison	< > <= >= instanceof
	equality	== !=
Bitwise	bitwise AND	&
	bitwise exclusive OR	^
	bitwise inclusive OR	|
Logical	logical AND	&&
	logical OR	||
Ternary	ternary	? :
Assignment	assignment	= += -= *= /= %= &= ^= |= <<= >>= >>>=

The Java **left shift operator <<** is used to shift all of the bits in a value to the left side of a specified number of times.
The Java **right shift operator >>** is used to move left operands value to right by the number of bits specified by the right operand.
The **logical && operator doesn't check second condition if first condition is false.** It checks second condition only if first one is true.
The **bitwise & operator always checks both conditions whether first condition is true or false.**
The logical || operator doesn't check second condition if first condition is true. It checks second condition only if first one is false.
The bitwise | operator always checks both conditions whether first condition is true or false.
Java Ternary operator is used as one liner replacement for if-then-else statement and used a lot in Java programming. it is the only conditional operator which takes three operands.

LOOPS

For-each loop
The for-each loop is used to traverse array or collection in java. It is easier to use than simple for loop because we don't need to increment value and use subscript notation.
It works on elements basis not index. It returns element one by one in the defined variable.

Labelled For Loop
We can have a name of each Java for loop. To do so, we use label before the for loop. It is useful if we have nested for loop so that we can break/continue specific for loop.

Usually, break and continue keywords breaks/continues the innermost for loop only.

paradigm to design a program using classes and objects. It simplifies software development and maintenance by providing some concepts:

Inheritance
Polymorphism
Abstraction
Encapsulation

A class in Java can contain:

1. Fields
2. Methods
3. Constructors
4. Blocks
5. Nested class and interface

Instance variable doesn't get memory at compile time. It gets memory at runtime when an object or instance is created. That is why it is known as an instance variable.

Method in Java
In Java, a method is like a function which is used to expose the behavior of an object.

new keyword in Java
The new keyword is used to allocate memory at runtime. All objects get memory in Heap memory area.

object gets the memory in heap memory area. The reference variable refers to the object allocated in the heap memory area. Here, s1 and s2 both are reference variables that refer to the objects allocated in memory

Anonymous Object
Anonymous simply means nameless. An object which has no reference is known as an anonymous object. It can be used at the time of object creation only.

If you have to use an object only once, an anonymous object is a good approach.

Difference between constructor and method in Java

There are many differences between constructors and methods. They are given below.

Java Constructor	Java Method
A constructor is used to initialize the state of an object.	A method is used to expose the behavior of an object.
A constructor must not have a return type.	A method must have a return type.
The constructor is invoked implicitly.	The method is invoked explicitly.
The Java compiler provides a default constructor if you don't have any constructor in a class.	The method is not provided by the compiler in any case.
The constructor name must be same as the class name.	The method name may or may not be same as the class name.

INHERITANCE

Why use inheritance in java

1. For Method Overriding.
2. For Code Reusability.

Types of inheritance in java
On the basis of class, there can be three types of inheritance in java: single, multilevel and hierarchical.

In java programming, multiple and hybrid inheritance is supported through interface only.

Why multiple inheritance is not supported in java?
To reduce the complexity and simplify the language, multiple inheritance is not supported in java.

Consider a scenario where A, B, and C are three classes. The C class inherits A and B classes. If A and B classes have the same method and you call it from child class object, there will be ambiguity to call the method of A or B class.

Since compile-time errors are better than runtime errors, Java renders compile-time error if you inherit 2 classes. So whether you have same method or different, there will be compile time error.

extends keywords is used to inherit a class
B extends A implies B inherits A

POLYMORPHISM
Polymorphism in Java: compile-time polymorphism and runtime polymorphism. perform polymorphism in java by method overloading and method overriding.

If you overload a static method in Java, it is compile time polymorphism.

Method Overloading
If a class has multiple methods having same name but different in parameters, it is known as Method Overloading.

Different ways to overload the method
1. By changing number of arguments
2. By changing the data type

In Java, Method Overloading is not possible by changing the return type of the method only.
Compile Time Error is better than Run Time Error.

Can we overload java main() method?
Yes, by method overloading. You can have any number of main methods in a class by method overloading. But JVM calls main() method which receives string array as arguments only.

Type Promotion:
If there are matching type arguments in the method, type promotion is not performed.
If there are no matching type arguments in the method, and each method promotes similar number of arguments, there will be ambiguity.

Data type is not de-promoted implicitly for example double cannot be de-promoted to any type implicitly.

Method Overriding
If subclass has the same method as declared in the parent class, it is known as method overriding in Java.

Usage of Java Method Overriding
1. Method overriding is used to provide the specific implementation of a method which is already provided by its superclass.
2. Method overriding is used for runtime polymorphism

Java method overriding is mostly used in Runtime Polymorphism

Can we override static method?
No, a static method cannot be overridden. It can be proved by runtime polymorphism
static method is bound with class

Runtime Polymorphism in Java
Runtime polymorphism or Dynamic Method Dispatch is a process in which a call to an overridden method is resolved at runtime rather than compile-time.

In this process, an overridden method is called through the reference variable of a superclass. The determination of the method to be called is based on the object being referred to by the reference variable.

Upcasting: If the reference variable of Parent class refers to the object of Child class, it is known as upcasting.

Down casting
When Subclass type refers to the object of Parent class, it is known as down casting.

perform it directly, compiler gives Compilation error.

ClassCastException is thrown at runtime.

Since method invocation is determined by the JVM not compiler, it is known as runtime polymorphism.

A method is overridden, not the data members, so runtime polymorphism can't be achieved by data members.

Binding
Two types
1. Static - Early Binding
2. Dynamic - Late Binding

static binding
When type of the object is determined at compiled time(by the compiler), it is known as static binding.

If there is any private, final or static method in a class, there is static binding.

Dynamic binding
object type cannot be determined by the compiler this happens ar Run time
So compiler doesn't know its type, only its base type.

Abstraction
Abstract class in Java
1. It can have abstract and non-abstract methods (method with the body).
2. It needs to be extended and its method implemented.
3. It cannot be instantiated.
4. It can have constructors and static methods also.
5. It can have final methods which will force the subclass not to change the body of the method.

Abstraction is a process of hiding the implementation details and showing only functionality to the user.

Ways to achieve Abstraction
There are two ways to achieve abstraction in java
1. Abstract class (0 to 100%)
2. Interface (100%)

An abstract class can have a data member, abstract method, method body (non-abstract method), constructor, and even main() method.

Encapsulation
Provides you the control over the data.
It is a way to achieve data hiding in Java because other class will not be able to access the data through the private data members.

Autoboxing
The automatic conversion of primitive data type into its corresponding wrapper class is known as autoboxing, for example, byte to Byte, char to Character, int to Integer, long to Long, float to Float, boolean to Boolean, double to Double, and short to Short. Since Java 5, we do not need to use the valueOf() method of wrapper classes to convert the primitive into objects.

Unboxing
The automatic conversion of wrapper type into its corresponding primitive type is known as unboxing. It is the reverse process of autoboxing. Since Java 5, we do not need to use the intValue() method of wrapper classes to convert the wrapper type into primitives.

Static, This, Final, Super Keyword

static KEY WORD
used for memory management
The static keyword belongs to the class than an instance of the class.

The static can be:
1. Variable (class variable)
2. Method (class method)
3. Block
4. Nested class

Static Variable
refer to the common property of all objects
The static variable gets memory only once in the class area at the time of class loading
Memory Efficient

Java static method

1. class rather than the object of a class.
2. invoked without the need for creating an instance of a class.
3. can access static data member and can change the value of it.
4. The static method cannot use non static data member or call non-static method directly.
5. this and super cannot be used in static context.

Why is the Java main method static?
It is because the object is not required to call a static method. If it were a non-static method, JVM creates an object first then call main() method that will lead the problem of extra memory allocation.

getclass() method returns name of the class

this keyword

this is a reference variable that refers to the current object.
Usage of java this keyword
Here is given the 6 usage of java this keyword.
1. refer current class instance variable.
2. invoke current class method (implicitly)
3. invoke current class constructor.
4. passed as an argument in the method call.
5. passed as argument in the constructor call.
6. return the current class instance from the method.

When parameters (formal arguments) and instance variables are same we are use this keyword to distinguish local variable and instance variable.

super Key Word
reference variable which is used to refer immediate parent class object.
Whenever you create the instance of subclass, an instance of parent class is created implicitly which is referred by super reference variable.
Usage of Java super Keyword
1. refer immediate parent class instance variable.
2. invoke immediate parent class method.
3. super() can be used to invoke immediate parent class constructor.

super keyword to access the data member or field of parent class.
used if parent class and child class have same fields.

super can be used to invoke parent class method
The super keyword can also be used to invoke parent class method. It should be used if subclass contains the same method as parent class. In other words, it is used if method is overridden.
By default because priority is given to local.
To call the parent class method, we need to use super keyword.

super() constructor
super() is added in each class constructor automatically by compiler if there is no super() or this().
By default constructor is provided by compiler automatically if there is no constructor. But, it also adds super() as the first statement.

final Keyword
is used to restrict the user.
cannot override final method
cannot extend final class
Stops value, denies inheritance and orveridding

Is final method inherited?
Yes, final method is inherited but you cannot override it.

What is blank or uninitialized final variable?
A final variable that is not initialized at the time of declaration is known as blank final variable.

If you want to create a variable that is initialized at the time of creating object and once initialized may not be changed, it is useful.

Can we initialize blank final variable?
Yes, but only in constructor.

static blank final variable
A static final variable that is not initialized at the time of declaration is known as static blank final variable. It can be initialized only in static block.

What is final parameter?
If you declare any parameter as final, you cannot change the value of it.

Can we declare a constructor final?
No, because constructor is never inherited.

Constructor and Destructor

Constructor
1. Default
2. Copy
3. Parameterized

Constructor Chaining
When a series of constructors are invoked by the declaration of a new object

For array of objects, the constructors must be called separately for each element in that array

Does constructor return any value?
Yes, it is the current class instance (You cannot use return type yet it returns a value).

Can constructor perform other tasks instead of initialization?
Yes, like object creation, starting a thread, calling a method, etc. You can perform any operation in the constructor as you perform in the method.

Is there Constructor class in Java?
Yes.

What is the purpose of Constructor class?
Java provides a Constructor class which can be used to get the internal information of a constructor in the class. It is found in the java.lang.reflect package.

Destructor
Called garbage collection in java
JVM decides when to delete an object
No explicit method to destroy unused objects

Frees heap memory
Non deterministic i.e. user can't predict when the memory is destroyed

Hotspot by oracle is most commonly used JVM

Types of garbage collector
1. Serial
2. Parallel
3. CMS collector (Concurrent Mask Sweep)
4. CI collector (Concurrent Collector)

system.gc() in code for immediate garbage collection
This is static and class specific

Runtime.getRuntime().gc() can also be used
This is instance specific

Object methods

Super class for all classes is Object class

Contained in package java.lang.Object

There exists 2 methods to compare objects:
equals()
hashcode()

The object methods are:
Clone() - copy an object
Finalize() - called for garbage collection
getClass() - Gives class name of the object that invoked it
toString() - Text representation of object

Equals method
x.equals(x) should return true.
x.equals(y) should return true if and only if y.equals(x) returns true.
Multiple invocations of x.equals(y) should return same result, unless any of the object properties is modified that is being used in the equals() method implementation.
Object class equals() method implementation returns true only when both the references are pointing to same object

Hashcode method
Returns the integer hash code value of the object

An object hash code value can change in multiple executions of the same application.
If two objects are equal according to equals() method, then their hash code must be same.
If two objects are unequal according to equals() method, their hash code are not required to be different. Their hash code value may or may-not be equal.

Error and Exception

Errors — These are not exceptions at all, but problems that arise beyond the control of the user or the programmer. Errors are typically ignored in your code because you can rarely do anything about an error. They are also ignored at the time of compilation.

Exception Handling
Handles run time errors

Exception is an occurrence of an event that disrupts program flow
Object is thrown at run time

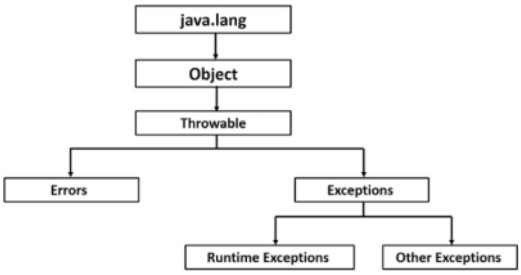
Exception Handling types
ClassNotFoundException, IOException, SQLException

Exceptions can occur in these scenarios
invalid data.
file that needs to be opened cannot be found.
A network connection has been lost in the middle of communications
JVM has run out of memory

Checked Exception
A checked exception is an exception that is checked (notified) by the compiler at compilation-time, these are also called as compile time exceptions.

Unchecked Exception
unchecked exception is an exception that occurs at the time of execution. These are also called as Runtime Exceptions

All exception classes are subtypes of the java.lang.Exception class.
exception class is a subclass of the Throwable class.



Use try and catch to handle exceptions

```
try {
   // Protected code
} catch (ExceptionName e1) {
   // Catch block
}
```

A catch statement involves declaring the type of exception you are trying to catch. If an exception occurs in protected code,
the catch block (or blocks) that follows the try is checked.
If the type of exception that occurred is listed in a catch block, the exception is passed to the catch block much as an argument is passed into a method parameter.

throw an exception by using the throw keyword.
newly instantiated one or an exception just caught

The finally block follows a try block or a catch block. A finally block of code always executes, irrespective of occurrence of an Exception.
finally block appears at the end of the catch blocks

```
try {
   // Protected code
} catch (ExceptionType1 e1) {
   // Catch block
} catch (ExceptionType2 e2) {
   // Catch block
} catch (ExceptionType3 e3) {
   // Catch block
}finally {
   // The finally block always executes.
}
```

I/O streams

Input Devices:
Hardware - keyboard, scanner, mouse
Software - files, socket

Output Devices:
Hardware: Monitor, Speaker
Software - files, socket

Input Stream - File input stream
Obtaining input from files

To get path of the file
getpath() : obtains path relative to the path of the current code
getAbsolutepath() : obtains path from the origin

File() is used to create or delete a file but not for manipulation

Buffer Reader: creates a buffer for sending data
Buffer Writer: creates a buffer for receiving data

Serialization

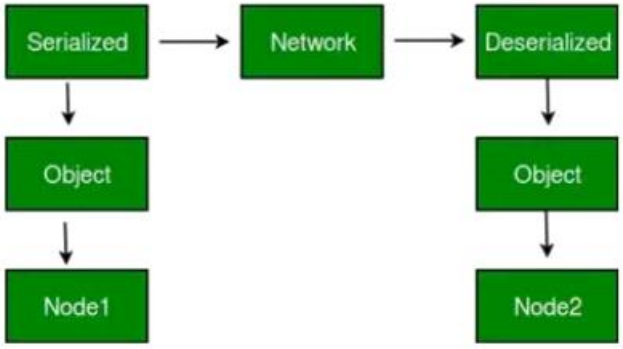
17 October 2020 09:21

Serialization: Object State is stored as byte stream
De serialization: object State is retrieved from byte stream

Marker Interface: no methods no data members
It is used to mark classes so that that the objects may certain capability
The serializable object is used by implementing java.io.Serializable

readObject() - for deserialization in ObjectInputStream (public final)
[IO Exception, ClassNotFound Exception]
WriteObject() - for serialization in ObjectOutputStream (public final)
[IO Exception]

Advantages of Serialization
1. To save/persist state of an object.
2. To travel an object across a network.




```
1.    import java.io.Serializable;
2.    public class Student implements Serializable{
3.     int id;
4.     String name;
5.     public Student(int id, String name) {
6.      this.id = id;
7.this.name = name;  8.}
9.    }
```

```
1. import java.io.*;
2. class Persist{
3.  public static void main(String args[]){
4.   try{
5.   //Creating the object
6.   Student s1 =new Student(211,"ravi");
7.   //Creating stream and writing the object
8.   FileOutputStream fout=new FileOutputStream("f.txt");
9.   ObjectOutputStream out=new ObjectOutputStream(fout);
10.  out.writeObject(s1);
11.  out.flush();
12.  //closing the stream
13.  out.close();
14.  System.out.println("success");
15.  }catch(Exception e){System.out.println(e);}
16.} 17.}
```

Steps to serialize:-

1. Create a sort of <u>any output stream object</u>
   Example: FileOutputStream fout = new FileOutputStream("f.txt");
2. Wrap it inside an object OutputStream

ObjectOuotuStream out = new ObjectOutputStram (fout);

3. Serialize the object by calling write object method
   out.writeObject(s1);
4. Flush and close all the stream objects
   out.flush();
   out.close();

What happens if you do not use flush and close??

```
1.  import java.io.*;
2.  class Depersist{
3.   public static void main(String args[]){
4.    try{
5.    //Creating stream to read the object
6.    ObjectInputStream in=new ObjectInputStream(new FileInputStream("f.txt"));
7.    Student s=(Student)in.readObject();
8.    //printing the data of the serialized object
9.    System.out.println(s.id+" "+s.name);
10.   //closing the stream
11.   in.close();
12.    }catch(Exception e){System.out.println(e);}
13.  }
14.  }
```

Steps to deserialize:-

1. Create a sort of any input stream object
   Example: FileInputStream fin = new FileOutputStream("f.txt");
2. Wrap it inside an object InputStream
   ObjectInputStream in = new ObjectInputStream (fin);
3. Deserialize the object by calling read object method
   Student s= (Student) in.readObject();
4. Close all the stream objects
   out.close();

Points to remember

1. If a parent class has implemented Serializable interface then child class doesn't need to implement it but vice-versa is not true.
2. **Only non-static data members are saved via Serialization process.**
3. Static data members and transient data members are not saved via Serialization process. So, if you don't want to save value of a non-static data member then make it transient
4. Constructor of object is never called when an object is deserialized.
5. Associated objects must be implementing Serializable interface. **?????**

Controlling/Restrict serialization in a class:

1. Using transient on the data members
2. Read External and Write External