VIVA PREPARATION

How many types of memory areas are allocated by JVM?

- **Class(Method) Area:** Class Area stores per-class structures such as the runtime constant pool, field, method data, and the code for methods.
- **Heap:** It is the runtime data area in which the memory is allocated to the objects
- **Stack:** Java Stack stores frames. It holds local variables and partial results, and plays a part in method invocation and return. Each thread has a private JVM stack, created at the same time as the thread. A new frame is created each time a method is invoked. A frame is destroyed when its method invocation completes.
- **Program Counter Register:** PC (program counter) register contains the address of the Java virtual machine instruction currently being executed.
- **Native Method Stack:** It contains all the native methods used in the application.

Access Specifiers in Java

- **Public** The classes, methods, or variables which are defined as public, can be accessed by any class or method.
- **Protected** Protected can be accessed by the class of the same package, or by the sub-class of this class, or within the same class.
- **Default** Default are accessible within the package only. By default, all the classes, methods, and variables are of default scope.
- **Private** The private class, methods, or variables defined as private can be accessed within the class only.

What is an Object ?

The Object is the real-time entity having some state and behavior. In Java, Object is an instance of the class having the instance variables as the state of the object and the methods as the behavior of the object. The object of a class can be created by using the new keyword.

What is the constructor?

The constructor can be defined as the special type of method that is used to initialize the state of an object. It is invoked when the class is instantiated, and the memory is allocated for the object. Every time, an object is created using the **new** keyword, the default constructor of the class is called.

Types of Constructor in JAVA

- Default Constructor
  A default constructor is invoked implicitly by the compiler if there is no constructor defined in the class.

- Parameterized Constructor
  The parameterized constructor is the one which can initialize the instance variables with the given values. In other words, we can say that the constructors which can accept the arguments are called parameterized constructors.

  - The constructor implicitly returns the current instance of the class
  - Constructor can't be final.
  - Constructors can be overloaded by changing the number of arguments accepted by the constructor or by changing the data type of the parameters.

What is the static variable ?

The static variable is used to refer to the common property of all objects (that is not unique for each object), e.g., The company name of employees, college name of students, etc. Static variable gets memory only once in the class area at the time of class loading. Static variable belongs to the class rather than the object.

What is Static Method ?

- A static method belongs to the class rather than the object.
- There is no need to create the object to call the static methods.
- A static method can access and change the value of the static variable.
- The static method cannot use non-static data member or call the non-static method directly.
- this and super cannot be used in static context as they are non-static.
- We can't override static methods.

What is Instance Method ?

- A method that is not declared as static is known as the instance method.
- The object is required to call the instance methods.
- Static and non-static variables both can be accessed in instance methods.

this keyword in JAVA

- The **this** keyword is a reference variable that refers to the current object. There are the various uses of this keyword in Java.
- It can be used to refer to current class properties such as instance methods, variable, constructors, etc.
- It can also be passed as an argument into the methods or constructors. It can also be returned from the method as the current class instance.

What is the Inheritance ?

Inheritance is a mechanism by which one object acquires all the properties and behavior of another object of another class.

There are five types of inheritance in Java.

- Single-level inheritance
- Multi-level inheritance
- Multiple Inheritance
- Hierarchical Inheritance
- Hybrid Inheritance

Multiple inheritance is not supported in Java through class.

What is super in JAVA ?

- super can be used to refer to the immediate parent class instance variable.
- super can be used to invoke the immediate parent class method.
- super() can be used to invoke immediate parent class constructor.

What is method Overriding ?

If a subclass provides a specific implementation of a method that is already provided by its parent class, it is known as Method Overriding. It is used for runtime polymorphism and to implement the interface methods.

Rules for Method overriding

- The method must have the same name as in the parent class.
- The method must have the same signature as in the parent class.
- Two classes must have an IS-A relationship between them.

What is the final variable ?

The final variable is used to restrict the user from updating it. If we initialize the final variable, we can't change its value. In other words, we can say that the final variable once assigned to a value, can never be changed after that. The final variable which is not assigned to any value can only be assigned through the class constructor.

Compile time & Runtime Polymorphism

Compile Time	Runtime
• In compile-time polymorphism, call to a method is resolved at compile-time. • It is also known as static binding, early binding, or overloading. • Overloading is a way to achieve compile-time polymorphism	In runtime polymorphism, call to an overridden method is resolved at runtime. It is also known as dynamic binding, late binding, overriding, or dynamic method dispatch. Overriding is a way to achieve runtime polymorphism

Dynamic Method Dispatch

Runtime polymorphism or dynamic method dispatch is a process in which a call to an overridden method is resolved at runtime rather than at compile-time. In this process, an overridden method is called through the reference variable of a superclass.

Dynamic Method Lookup

Dynamic method lookup is the process of determining which method definition a method signature denotes during runtime, based on the type of the object.

What is an Abstract Class ?

- A class that is declared as abstract is known as an abstract class.
- It needs to be extended and its method implemented. It cannot be instantiated
- It can have abstract methods, non-abstract methods, constructors, and static methods.
- If there is an abstract method in a class, that class must be abstract.
- Abstract class can never be instantiated even if it contains a constructor and all of its methods are implemented.

What is an Interface ?

- The interface is a blueprint for a class that has static constants and abstract methods.
- It can be used to achieve full abstraction and multiple inheritance.
- There can be only abstract methods in the Java interface, not method body.
- All abstract, default, and static methods in an interface are implicitly public
- All constant values defined in an interface are implicitly public, static, and final .

What is a Package ?

- A package is a group of similar type of classes, interfaces, and sub-packages. It provides access protection and removes naming collision.
- The packages in Java can be categorized into two forms, inbuilt package, and user-defined package.
- There are many built-in packages such as Java, lang, awt, javax, swing, net, io, util, sql, etc.

What is Exception Handling ?

Exception Handling is a mechanism that is used to handle runtime errors. It is used primarily to handle checked exceptions. Exception handling maintains the normal flow of the program. There are mainly two types of exceptions: checked and unchecked. Here, the error is considered as the unchecked exception.

Java Exception Keywords

Java provides five keywords that are used to handle the exception. The following table describes each.

Keyword	Description
try	The "try" keyword is used to specify a block where we should place an exception code. It means we can't use try block alone. The try block must be followed by either catch or finally.
catch	The "catch" block is used to handle the exception. It must be preceded by try block which means we can't use catch block alone. It can be followed by finally block later.
finally	The "finally" block is used to execute the necessary code of the program. It is executed whether an exception is handled or not.
throw	The "throw" keyword is used to throw an exception.
throws	The "throws" keyword is used to declare exceptions. It specifies that there may occur an exception in the method. It doesn't throw an exception. It is always used with method signature.

Difference between Checked and Unchecked Exceptions

- **Checked Exception**

The classes that directly inherit the Throwable class except RuntimeException and Error are known as checked exceptions. For example, IOException, SQLException, etc. Checked exceptions are checked at compile-time.

- **Unchecked Exception**

The classes that inherit the RuntimeException are known as unchecked exceptions. For example, ArithmeticException, NullPointerException, ArrayIndexOutOfBoundsException, etc. Unchecked exceptions are not checked at compile-time, but they are checked at runtime.

- **Error**

Error is irrecoverable. Some example of errors are OutOfMemoryError, VirtualMachineError, AssertionError etc.

Checked Exceptions

- **Airthmetic Exceptions**

If we divide any number by zero, there occurs an ArithmeticException.

- **Null Pointer Exception**

If we have a null value in any variable, performing any operation on the variable throws a NullPointerException.

- **Number Format Exception**

If the formatting of any variable or number is mismatched, it may result into NumberFormatException. Suppose we have a string variable that has characters; converting this variable into digit will cause NumberFormatException.

- **Array Index out of Bound Exception**

When an array exceeds to it's size, the ArrayIndexOutOfBoundsException occurs.

Java String

Generally, String is a sequence of characters. But in Java, string is an object that represents a sequence of characters. The java.lang.String class is used to create a string object.

Java String class provides a lot of methods to perform operations on strings such as compare(), concat(), equals(), split(), length(), replace(), compareTo(), intern(), substring() etc.

The java.lang.String class implements *Serializable*, *Comparable* and *CharSequence* interfaces.

Java String class methods

The java.lang.String class provides many useful methods to perform operations on sequence of char values.

No.	Method	Description
1	char charAt(int index)	It returns char value for the particular index
2	int length()	It returns string length
3	static String format(String format, Object... args)	It returns a formatted string.
4	static String format(Locale l, String format, Object... args)	It returns formatted string with given locale.
5	String substring(int beginIndex)	It returns substring for given begin index.
6	String substring(int beginIndex, int endIndex)	It returns substring for given begin index and end index.
7	boolean contains(CharSequence s)	It returns true or false after matching the sequence of char value.
8	static String join(CharSequence delimiter, CharSequence... elements)	It returns a joined string.

9	static String join(CharSequence delimiter, Iterable<? extends CharSequence> elements)	It returns a joined string.
10	boolean equals(Object another)	It checks the equality of string with the given object.
11	boolean isEmpty()	It checks if string is empty.
12	String concat(String str)	It concatenates the specified string.
13	String replace(char old, char new)	It replaces all occurrences of the specified char value.
14	String replace(CharSequence old, CharSequence new)	It replaces all occurrences of the specified CharSequence.
15	static String equalsIgnoreCase(String another)	It compares another string. It doesn't check case.
16	String[] split(String regex)	It returns a split string matching regex.
17	String[] split(String regex, int limit)	It returns a split string matching regex and limit.
18	String intern()	It returns an interned string.
19	int indexOf(int ch)	It returns the specified char value index.
20	int indexOf(int ch, int fromIndex)	It returns the specified char value index starting with given index.
21	int indexOf(String substring)	It returns the specified substring index.
22	int indexOf(String substring, int fromIndex)	It returns the specified substring index starting with given index.
23	String toLowerCase()	It returns a string in lowercase.
24	String toLowerCase(Locale l)	It returns a string in lowercase using specified locale.

25	String toUpperCase()	It returns a string in uppercase.
26	String toUpperCase(Locale l)	It returns a string in uppercase using specified locale.
27	String trim()	It removes beginning and ending spaces of this string.
28	static String valueOf(int value)	It converts given type into string. It is an overloaded method.