

1) What do you know about JVM, JRE & JDK

→ JVM → It is a part of JRE & is responsible for providing environment for running Java bytecode.

JRE → It is used for running Java application. JVM with core Java API together is known as JRE.

→ JDK → It is a complete Java development kit which consists of development tools, standard library libraries, libraries, runtime environment and documentation.

2) Is JRE platform dependent or independent?

→ JRE is platform dependent because it creates platform dependent files to create JVM in the desired OS.

3) Which is ultimate base class in java class hierarchy? List the names of methods of it.

→ Object class is the ultimate base class.

Methods: equals(), hashCode(), finalize()

4) Which are reference types in Java?

- Class type → Reference point to object

- ArrayList type → Reference point to an ArrayList

Interface Type → Point to an object which implement an interface.

5) Explain narrowing and widening?

- Storing the smaller sized datatype into bigger sized datatype is called as widening.

Storing the bigger size datatype into smaller sized datatype.

is called as narrowing

Casting is required in narrowing.

6) How will you print "Hello DAc" without semicolon.

```
public class Main {
```

```
    public static void main(String[] args) {
```

```
        for (int i = 1; i < 2; System.out.print("Hello DAc"));
```

```
{ i++; } }
```



7) Can You write Java code "will main method ? If yes, how ?

- * No we can't - We can write but it won't be executed but will compile because during execution JVM searches for main method and if not found then gives runtime errors.

8) What will happen if we call main method inside static block?

- * We can give explicit call to main method from static block and due to this main method is called twice, one by JVM and other by static block.

9) In System.out.println, Explain every term)

System → Public final class in java.lang

out → Object of PrintStream

println → method of PrintStream

10 How will you pass object to the function by reference?

→ In java we pass the reference of an object by value which means the copy of reference is created while passing the object into the method parameters.

11 Explain constructor chaining process?
How can we achieve it in C++?

→ Constructor chaining is a technique to reuse the other constructors of same or parent classes.

In C++ constructor chaining is not possible in same class but possible from child to parent constructor calling with the help of base constructor initialization list.

Page (5) :

(12) What are the rules to overload method in subclass?

→ If we have have to overload method in a subclass then.

a] We should have same access method signature in subclass as that of method in superclass.

b] Parameters list should be different in subclass as that of superclass.

c] Less restrictive access modifier in subclass as that of parent class.

d] Static methods can also be overloaded as they are resolved at compile time.

(13) Difference between finalize & dispose

finalize

dispose

1) Implicit called.

by garbage collector

2) Explicitly called

by the developer

when needed.

2) Defined in

java.lang.Object class

3) Defined by developer

in specific class or library

- 3) Used for automatic garbage collection and releasing resource
- 3) Used to explicit resource management and cleanup.
- 4) Its calling cannot be determined.
- 4) Its recall can be determined.
- 5) ~~Unreliable~~ Unreliable
- 5) Reliable

~~Q1]~~ Difference between final, finally and finalize

~~final~~
Used to make class constant, and define the method for overriding.

~~finally~~

~~finalize~~

Instance Reference

PAGE NO.: 7
DATE:

- 14) Difference b/w final / finally and finalize

final - keyword final is used to define with variables, method & class) that cannot be modified, overridden, or extended.

finally - used in exception handling. The final block is used to specify a block of code that is executed regardless of whether an exception is thrown or not. It is typically used for clean up operation, resource release, or ensuring that certain code is always executed.

finalize - The finalize method is used for performing cleanup and resource release operations on an object before it is collected by the JVM.

This is called automatically by the JVM before returning the memory occupied

by the object.

15] Difference b/w among checked and un checked exception?

Checked Exception
Happens in
compile time

Unchecked Exception
Happens at
run time.

Checked by
compiler

Not checked by
compiler

This exception
is treated as a
Sub-class of the
class

This exception happens
in runtime, and
hence it is not
included in the
exception class -

JVM requires
the exception
to be
caught
or handled

JVM does not need
the exception to be
caught or handled

(16) Explain exception chaining?

→ It is technique of handling exception by re-throwing a caught exception after wrapping it inside a new exception. This is done to provide additional context or info about the original exception while still propagating the original exception up the call stack.

Exception chaining helps to preserve the original stack trace and error message, it easier to diagnoses and debug issue.

Difference b/w throw & throws?

throw:- used explicitly throw an exception within a method or block code. It indicates a specific exceptional condition and is followed by an instance of an exception or a subclass of 'Throwable'.

Used in method declarations to indicate that a method might throw one or more exceptions. It specifies the exception types that the method can throw but does not actually throw them. This is used for informing callers of potential exceptions that need to be handled.

Q8) In which case finally block doesn't execute?

Finally block doesn't execute for following cases:

- 1) If the program exits abnormally due to system crash or 'System.exit(1)' call.
- 2) If the thread executing the code is interrupted or killed forcefully.
- 3) If there is an infinite loop or an infinite blocking operation in the code.

'try' or 'catch' block that prevents control from reaching the 'finally' block.

19) Explain upcasting

→ Process of casting an object of a subclass to a reference of its super class. In short, it involves ~~treatment~~ treating a derived class object as an instance of its base class.

Upcasting is always allowed in object-oriented programming language that support inheritance.

20) Explain dynamic method dispatch?

Dynamic method dispatch is a feature in object-oriented programming that allows a method call to be resolved at runtime based on the actual type of the object being referenced, rather than the type of the ref reference itself. In short, it.

enables the selection of the appropriate method implementation in a subclass.

When an overridden method is calling using a reference to the superclass. This enables runtime polymorphism and is a fundamental concept in many object-oriented programming languages like Java.

2) What do you know about final method.

A final method is a method in a class that cannot be overridden by any subclass.

2) Explain fragile base class problem and how can we overcome it?
→ The fragile base class problem is a software development issue that arises in object-oriented programming when a class serves as a base class, and subclasses are created to extends its functionality.

The problem occurs when changes to the base class can inadvertently break functionality in the existing subclass, leading to unexpected errors or bugs.

- 1) Avoid modification in the base class.
- 2) Use interface and abstract classes to define contracts that subclasses must adhere to. This way you can make changes to implementations without affecting the contract.
- 3) Documentation & communication: Clearly document the intended behaviors and constraints of the base class, and communicate these to developers who create subclasses. This can help them anticipate potential issues.

Testing & Regression Testing :-

Implementation through testing, including regression testing, to ensure that changes to the base class do not introduce unintended side effects in the subclass.

Composition over Inheritance:

Consider using composition instead of inheritance when appropriate. This can reduce the reliance on a fragile base class.

By carefully managing the base class and using good design practices, you can mitigate the fragile base class problem and ensure a more robust and maintainable code base.

2B) Why java does not support multiple implementation inheritance?

→ Java does not support multiple implementation inheritance primarily to avoid the complexities and ambiguities that can arise from inheriting implementations from multiple classes. Instead, Java focuses on single inheritance of classes and multiple inheritance of interfaces to promote simplicity, maintainability, and to prevent issues like the "diamond problem" that can occur in languages supporting multiple implementation inheritance.

24) Explain marker interface? List the name of some marker interfaces?

→ A marker interface is a Java interface that does not declare any method or field but is used to indicate that a class implementing it possesses a certain capability or should be treated in a specific way by the compiler or runtime environment.

Marker Interface are also known as tag interface.

- 1) Serializable
- 2) Clonable
- 3) Remote
- 4) Random Access
- 5) Annotation
- 6) Auto closeable