CORE JAVA ANS

(1) Explain New Keyword ?

> it sends request to the class to create object.

> once object is created, 'new' keyword will get object address and store that in a reference variable.

(2) Explain Garbage Collector ?

In java , garbage collector is a component of JVM which automatically keeps on removing objects from heap memory on regular bases which are not in use which helps to avoid overflow of memory and memory management becomes efficient.

(2.1) HOW GARBAGE COLLECTION WORKS?

Garbage collection works in 2 steps:

> marking: unreferenced objects in heap are identified first and marked as ready for garbage collection.

> deletion : objects marked previously is deleted.

(2.2) CAN YOU NAME COMMONLY USED ORACLE'S JVM & WHICH GC STRATEGY IS USED BY IT?

"HOTSPOT" is the most commonly used JVM by oracle,

All hotspot's GC implements generational GC strategy ex- it categorizes the objects by ages. Like
(a) Young generation

(b) old generation

(c) permanent generation.

(2.3) WHY MOST GC USES GENERATION GC STRATEGY?

Reason for most GC strategy is that most of the objects are short lived and hence die shortly just after memory allocation. Ex – local variable uses new keyword and when method execution end it will die.

(3) Static vs non-static variable ?

Static variable:

- Static variables are also called class variable and belong to class.
- Static variables declare outside of method but inside of class using static keyword.
- Static variable can be access directly with class name , object creation not mandatory.
- Static variable has global access directly with class name.
- Not mandatory to initialize.

Non-static variable:

- Non-static variables are also called instance variable and belong to object.
- Non-Static variables declare outside of method but inside of class without using static keyword.
- To access non-static variable object creation mandatory.
- Non-static doesn't have global access, after creation object can be accessed.
- Not mandatory to initialize.

(4) static vs local variable ?

Static variable:

- Static variables are also called class variable and belong to class.
- Static variables declare outside of method but inside of class using static keyword.
- Static variable can be access directly with class name , object creation not mandatory.
- Static variable has global access directly with class name.
- Not mandatory to initialize.

Local variable:

- Local variables are belong to Method/ block-level.
- Local variables are declare inside method and should be used only within the method.
- Local variables can be accessed directly without object reference or class name.
- mandatory to initialize.

(5) Reference variable ?

- reference variable can store either null value or object address.
- Data type of reference variable is class name.
- Reference variable can be treaded as : static ,non-static and local

(6) WHAT IS STATIC IN JAVA ?

Static is used to declare member of a class. It can be variable or method.

(7) WHY MAIN METHOD IS STATIC ?

By declaring main method as static to allow JVM to call main method directly without creating instance of the class.

(8) INSTEAD OF STRING [] AS MAIN METHOD ARGUMENT CAN WE PASS INT ARRAY ?

No

- Main method to have exact signature.
- The arguments passed to a Java program via the command line are always treated as strings. Even if you pass numbers, they are received as strings and must be converted to integers if needed

(9) DIFFERENCE BETWEEN PRIMITIVE VS NON-PRIMITIVE DATA TYPES?

Feature	Primitive Data Types	Non-Primitive Data Types
Definition	Built-in data types provided by Java for basic values like numbers, characters, and booleans.	User-defined or complex data types that reference objects or arrays
Storage	Stores the actual value directly in memory.	Stores a reference (or address) to the memory location where the object or array is stored.
Memory Location	Stored in **stack memory**.	Stored in **heap memory** (reference stored in stack).
Nullability	Cannot be null	Can be assigned `null` to indicate no object reference.
Examples	`byte`, `short`, `int`, `long`, `float`, `double`, `char`, `boolean`.	`String`, `Array`, `Class`, `Interface`, `HashMap`, `List`.

(10) WHAT IS DEFAULT VALUE ?

It we don't store any value of a variable then depending on it's data type, automatically some value store by compiler, called default value.

(11) DIFFERENCE BETWEEN HEAP AND STACK MEMEORY ?

Both **stack** and **heap memory** are parts of the **RAM (Random Access Memory)** used by a Java program to manage memory during execution

Feature	Stack Memory	Heap Memory
Definition	used to store **method call frames, local variables, and references**.	used for **dynamically allocated objects and class instances**.
Allocation	**Automatically allocated** when a method is called, and deallocated when the method ends.	**Manually allocated** by the programmer using `new` keyword or other mechanisms, and deallocated by the **Garbage Collector**.
Memory Size	Generally **small** in size.	Generally **large** in size.
Error Handling	May result in **StackOverflowError** if memory is exhausted (e.g., deep recursion).	May result in **OutOfMemoryError** if heap memory is exhausted.
Usage Example	Stores **local variables** and method call details.	Stores **objects, arrays**, and global data.

(12) DIFFERENCE BETWEEN JVM,JRE AND JDK ?

Aspect	JVM	JRE	JDK
Definition	A **virtual machine** that runs Java bytecode on any platform, providing platform independence.	A **runtime environment** that provides all the necessary libraries and tools to run Java applications.	A **development kit** that includes the tools required to develop, compile, and debug Java applications.
Purpose	Responsible for **executing Java programs** by converting bytecode into machine code.	Provides the environment required to **execute Java programs**.	Provides the tools needed to **develop Java programs**.
Includes	Only the JVM.	JVM + libraries and files to run Java applications.	JRE + development tools like `javac` (compiler), `javadoc`, `jdb`, etc.
Components	- Class Loader - Bytecode Verifier - Just-In-Time (JIT) Compiler - Garbage Collector	- JVM - Core libraries - Supporting files	- JRE - Compiler (`javac`) - Debugger (`jdb`) - Other development tools
Installation Requirement	Comes as part of JRE and JDK.	Installed on systems to **run** Java	Installed on systems to **develop** Java

		applications.	applications.

(13) WHAT IS CONSTRUCTOR AND WHAT IT DOES?

Constructor is a special type of method in object-oriented programming like, java,python… this is automatically called when object is created.

- It should have same name as that of class name.
- It is permanently void, if we use void keyword then it becomes method.
- Inside constructor we can create object, but mainly used for object initialization.

(14) HOW MANY TYPES OF CONSTRUCTOR ?

Mainly there are two types of constructor:

DEFAULT/ NON-PARAMETERIZED:

- It is automatically created by the compiler if no other constructor is defined.

PARAMETRIZED :

It is used to initialize the objects with user defined values.

(15) WHAT IS CONSTRUCTOR OVERLOADING & CONSTRUCTOR CHAINING ?

CONSTRUCTOR OVERLOADING:

We create more than one constructor in same class provided that they have different numbers of arguments or different types of arguments.

CONSTRUCTOR CHAINING:

Constructor Chaining refers to the technique in which a constructor in a class, calls another constructor of the same class or the superclass to reuse the initialization logic. This can be achieved using the `this()/super()` keyword.

(17) CAN WE INHERIT CONSTRUCTOR ?

No, because constructor should have same name as that class name which will make cause , instead of inheritance we can use super keyword.

(18) DOES CONSTRUCTORY RETURN ANY VALUE?

No, permanently void.

(19) DIFFERENCE BETWEEN CONSTRUCTOR AND METHOD?

CONSTRUCTOR:

- Used for object initialization.
- It runs when object created.
- Permanently void

METHOD:

- Used for breaking the code in smaller reusable modules.
- It runs only when called.
- It has various return types.

(20) WHAT IS UNREACHEABLE CODE ERROR ?

When the code is not compiled due to control back to method calling statement then the rest of code which is uncompiled called, unreachable code error.

(21) DOES MULTIPLE INHERITANCE ALLOW AT CLASS LEVEL?

No, due to complexity as soon as project grows bigger in size.

(22) CAN WE INHERIT STATIC METHOD ?

Yes, However, static methods belong to the class itself, not to an instance, which means they are not bound to objects but to the class itself.

(23) CAN WE OVERRIDE STATIC METHOD?

No, If a static method in the subclass has the same signature as the static method in the superclass, it **hides** the superclass's static method rather than overriding it.

(24) CAN WE CREATE MORE THAN ONE MAIN METHOD IN SAME CLASS ?

No, main method should exact one with same signature for JVM but through overloading concept we can create.

(25) CAN YOU EXPLAIN MAIN METHOD SIGNATURE ?

public static void main(String[] args)

(26) WHAT IS DATA HIDING ?

When we make variable private so that can't access outside the class is called data hiding, In Java, the `private` access modifier is typically used to hide the data and To allow controlled access to these fields, **getter** and **setter** methods are used.

(28) DIFFERENCE BETWEEN SUB CLASS AND NON-SUB CLASS?

SUB CLASS:

- A sub-class has an inheritance relationship with its super class.
- A class which is derived from another class is called sub-class.

NON-SUB CLASS:

- A non-sub class does not have inheritance relationship.
- Without inheritance when we access one class member in another class called non-sub class.

(29) HOW MANY TYPES OF VARIABLE IN JAVA ?

- Static variable.
- Non-static variable.
- Local variable.
- Reference variable

(30) DIFFERENCE BETWEEN RETURN AND RETURN VALUE?

Return :

- It should be used only inside a void method.
- It is optional to use
- It returns control back to method calling statement.
- It can be used inside main method because main method is void.

Return Value:

- It should be used except void method.
- It is mandatory to use.
- It returns both value and control to method calling statement.

(32) WHAT IS TYPE CASTING/CASTING & HOW MANY TYPES OF IT?

Type casting is the process, converting a particular data type into required data type, is called type casting or casting.

THERE ARE TWO TYPES OF CASTING:

- AUTO-UP CASTING/IMPLICIT CASTING: converting a smaller data type into bigger data type without losing of data , is called auto-up casting.
- DOWN CASTING/EXPLICIT CASTING: converting a bigger data type into smaller data type which might result in data loss, called down casting.

Note: Class casting in Java refers to converting an object of one type to another within the inheritance because without inheritance ,Casting is **not possible** unless the objects are related by inheritance.

Example:

```
Parent parent = new Child(); // The object is actually a Child

Child child = (Child) parent; // Downcasting works fine

System.out.println("Downcasting successful");
```

Explanation:

When `Parent parent = new Parent();`:

- The object is a **pure `Parent` object**, so casting it to `Child` will cause a `ClassCastException`.

When `Parent parent = new Child();`:

- The object is **actually a `Child` object**, so casting it to `Child` works.

(33) WHAT IS TYPE CONVERSION?

type conversion happens automatically, while type casting is done manually by the programmer.

Ex - char ch = ‘a’;

Int I = ch;

Sop(i); // 97

(35) WHAT ARE MARKER INTERFACE AND WHY IT USE?

Empty interface are called marker interface. No methods or fields.

WHY USE IT:

Marker interface is used as a tag to inform a message to the compiler so that the Classes implementing marker interfaces are treated differently during execution.

Marker Interface	Purpose
Serializable	Marks a class as serializable, enabling its objects to be converted to a byte stream.
Cloneable	Marks a class as cloneable, allowing it to support the `clone()` method from `Object`.

(36) EXPLAIN BLANK FIELD ERROR?

When we make variable static and final , and do not initialize then we get blank field error.

(37) EXPLAIN FINAL, FINALLY AND FINALIZED ?

Keyword	Definition	Usage
final	Used to define constants, prevent inheritance, and prevent method overriding.	Applied to variables, methods, and classes.
Finally - block	A block used for cleanup code after a `try-catch` block, always executed.	Used in exception handling.
finalize() - method	Method in the `Object` class called by the garbage collector before object destruction.	Used for object cleanup before garbage collection.

(38) EXPLAIN SERIALIZATION, DESERIALIZATION AND TRANSIENT KEYWORD?

SERIALIZATION:

Serialization is the process of converting an **object's state** into a **byte stream** (not necessarily "byte code") so that it can be saved to a **file**, sent over a network, or stored in a database.

DESERIALIZATION:

Deserialization is the reverse process of serialization where we read back from the file using `ObjectInputStream`, and it is reconstructed into its original object state.

TRANSIENT KEYWORD:

The **`transient`** keyword in Java is used to mark instance variables that should **not** be serialized. When an object is serialized, the `transient` fields are excluded from the serialization process.

(39) IF I WANT SOME FIELD TO STORE INTO DATABASE AS ENCRYPTED FORM HOW WILL BE DONE?

in Java, **`AttributeConverter`** is an interface that exists within the **JPA (Java Persistence API)** specification. It allows you to convert an entity attribute to and from the database column value during persistence operations.

(40) DIFFERENCE BETWEEN PARAMETERS & ARGUMENTS ?

PARAMETERS:

- Parameters are nothing but the copy of the arguments.

ARGUMENTS:

- Arguments are actual value passed to a function.

(41) DIFFERENCE BETWEEN PASS BY VALUE AND PASS BY REFERENCE?

Note: In Java, **everything is pass-by-value**, but it can be confusing when dealing with objects, as the value being passed is the **reference** to the object, not the object itself.

Aspect	Pass-by-Value	Pass-by-Reference
Definition	A copy of the value is passed to the method.	A reference (address) to the object is passed.
Effect on Primitives	Changes inside method do not affect the original value.	N/A
Effect on Objects	You can change the object's state, but cannot reassign the reference.	Changes inside method affect the original object because both share the same reference.

Example in Java (Pass-by-Value with Primitives):

```
public class PassByValueExample {

    public static void main(String[] args) {

        int num = 10;

        modifyValue(num);    // Passes the value of 'num' to the method

        System.out.println("Value of num after modification: " + num); // num is still 10
```

```
    }

    public static void modifyValue(int number) {
        number = 20; // Modifying the local copy of 'num'
        System.out.println("Value of number inside modifyValue method: " + number); // 20
    }
}
```

Example in Java (Pass-by-Value with Objects):

```
class Person {
    String name;
    int age;

    Person(String name, int age) {
        this.name = name;
        this.age = age;
    }
}

public class PassByValueExample {
    public static void main(String[] args) {
        Person p1 = new Person("Alice", 25);
        modifyPerson(p1);    // Passes a copy of the reference to the 'p1' object
        System.out.println("Person's name after modification: " + p1.name); // Name is
                                                                                modified
    }

    public static void modifyPerson(Person person) {
        person.name = "Bob"; // Modifying the object that both references point to
        person = new Person("Charlie", 30); // This does NOT affect the original p1 reference
```

```
    }
}
```

(42) DIFFERENCE BETWEEN IIB AND SIB ?

Aspect	IIB (Instance Initialization Block)	SIB (Static Initialization Block)
Purpose	Used to initialize **instance variables**.	Used to initialize **static variables** or perform static initialization.
Execution Time	Runs when an **object is created** (i.e., during object instantiation).	Runs when the **class is loaded** into memory (i.e., before any object is created).
Keyword	No special keyword. It's a block of code.	Declared using the `static` keyword.
Access to Instance Variables	Can access **instance variables** and **instance methods**.	Cannot access **instance variables** or **instance methods**.
Invocation	Invoked **automatically** when an instance of the class is created.	Invoked **automatically** when the class is loaded into memory, before any instance is created.
Use Case	Used when you need to initialize **instance variables** for each object.	Used when you need to initialize **static variables** or execute static code onc

(43) HOW TO PREVENT A CLASS NOT BE CLONED?

To prevent a class from being cloned despite implementing the `Cloneable` interface in Java, you can override the `clone()` method in the class and throw a `CloneNotSupportedException` explicitly

OVERRIDE METHOD:

```
@Override public Object clone() throws CloneNotSupportedException {
throw new CloneNotSupportedException("Cloning is not supported for this class.");
}
```

MAIN METHOD IN ANOTHER CLASS:

```
public static void main(String[] args) {
        MyClass obj = new MyClass(); // ONE OBJECT.
        try {
            // Attempt to clone the object
```

```
            MyClass clonedObj = (MyClass) obj.clone();
        } catch (CloneNotSupportedException e) {
            System.out.println(e.getMessage());
        }
    }
```

(44) CAN WE MAKE CONSTRUCTOR PRIVATE IF YES THEN HOW IT OBJECT WILL BE CREATED.?

Yes, we can create object using single ton approach.

(45) WHAT IS DIFFERENCE BETWEEN INTERFACE AND FUNCTIONAL INTERFACE?

Aspect	Interface	Functional Interface
Number of Abstract Methods	Can have multiple abstract methods.	Must have only one abstract method.
Default Methods	Can include multiple default methods.	Can also include default methods, but they don't affect its functional nature as long as it has one abstract method.
Static Methods	Can include multiple static methods.	Can include multiple static methods.
Annotation Usage	Typically not annotated.	Marked with the `@FunctionalInterface` annotation to indicate its intent and enforce compliance.
Examples	`Comparable`, `Iterator`, `List`, `Map`.	`Runnable`, `Callable`, `Supplier`, `Consumer`, `Function`.
Purpose	Used to define a contract that implementing classes must follow.	Used primarily in functional programming to represent a single functionality or action.

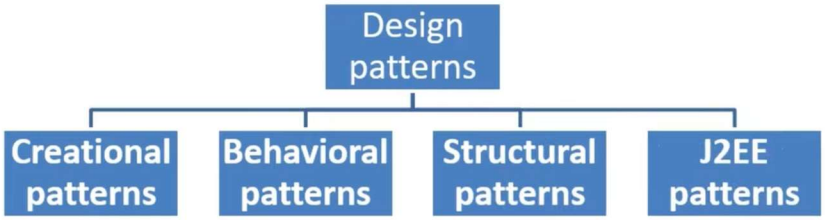
(46) WHAT ARE DESIGN PATTEN?

a design pattern is a reusable solution to a commonly occurring problem within a given context in software design. It is a template or blueprint for solving problems that can be adapted to specific situations.

(47) EXPLAIN TYPES OF DESIGN PATTERN?

Q) Categories Java Design patterns?

- We can categorize design patterns into the following categories.



(48) WHAT ARE THE CREATIONAL DESIGN PATTERN?

Q) What are the Creational Patterns?

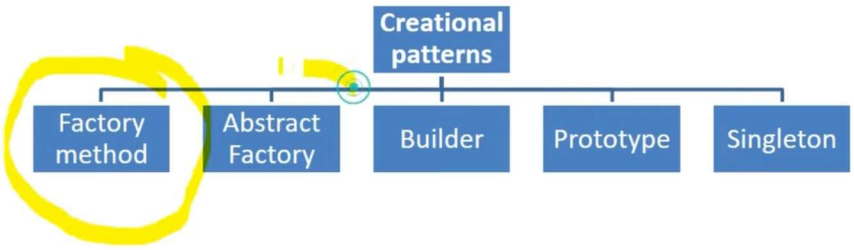
- Creational design patterns are related to the way of creating objects.
- This pattern is used to define and describe how objects are created at class instantiation time

- E.G.
  - Employee emp = new Employee();

    here we are creating the instance using the new keyword.

(49) HOW MANY TYPES OF CREATIONAL JAVA DESIGN PATTERN?

Q) Categories Java Design patterns?



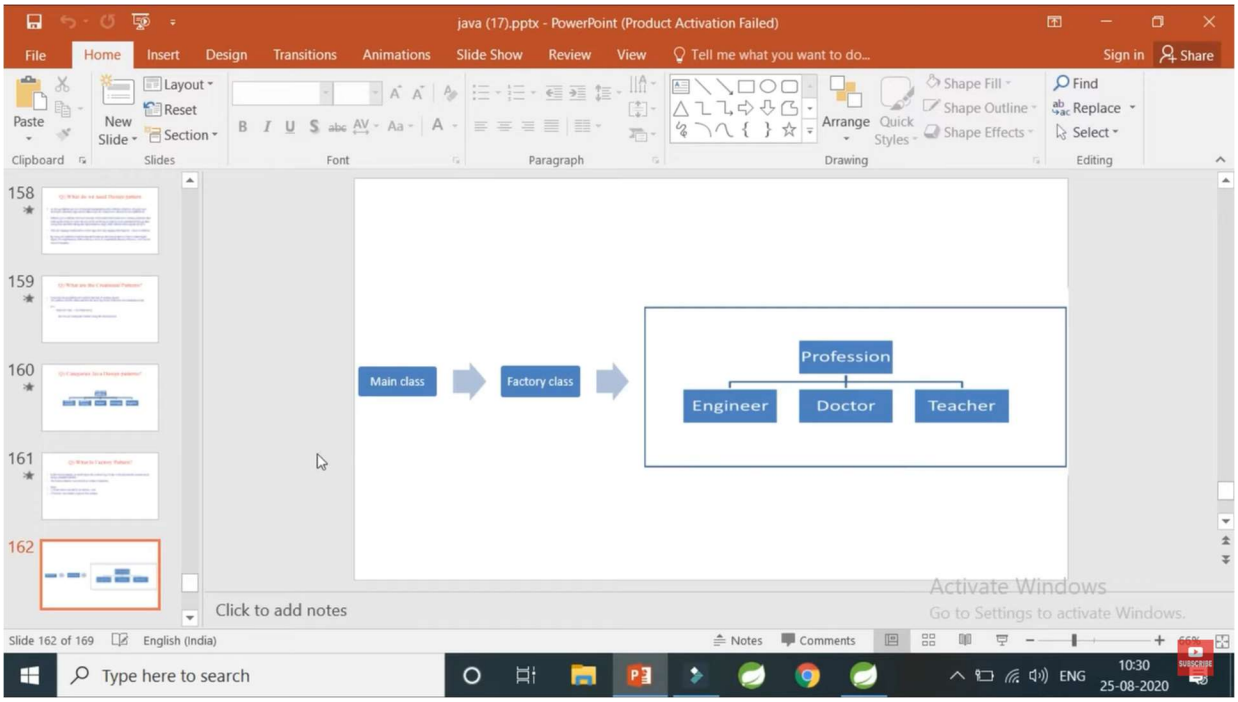
(50) WHAT IS FACTORY DESIGN PATTERN?

Q) What Is Factory Pattern?

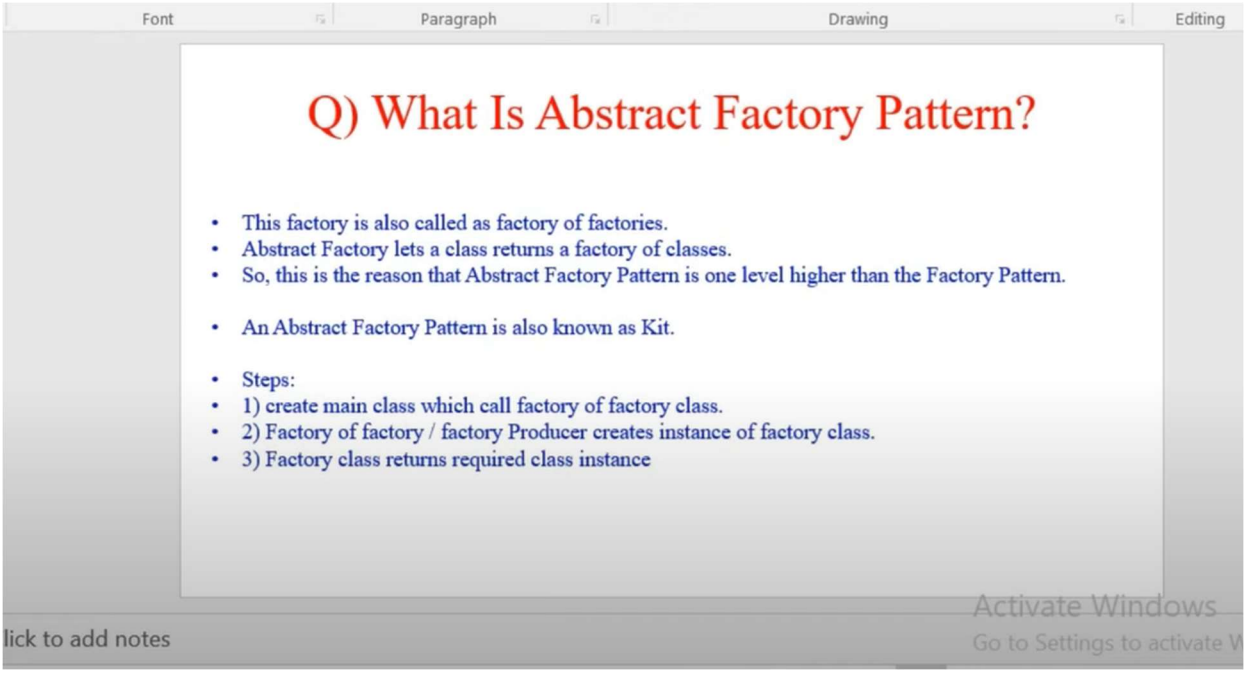
- In the Factory pattern, we don't expose the creation logic to the client and refer the created object using a standard interface.
- The Factory Pattern is also known as Virtual Constructor.

- Steps:
- 1) create main class which call factory class.
- 2) Factory class returns required class instance

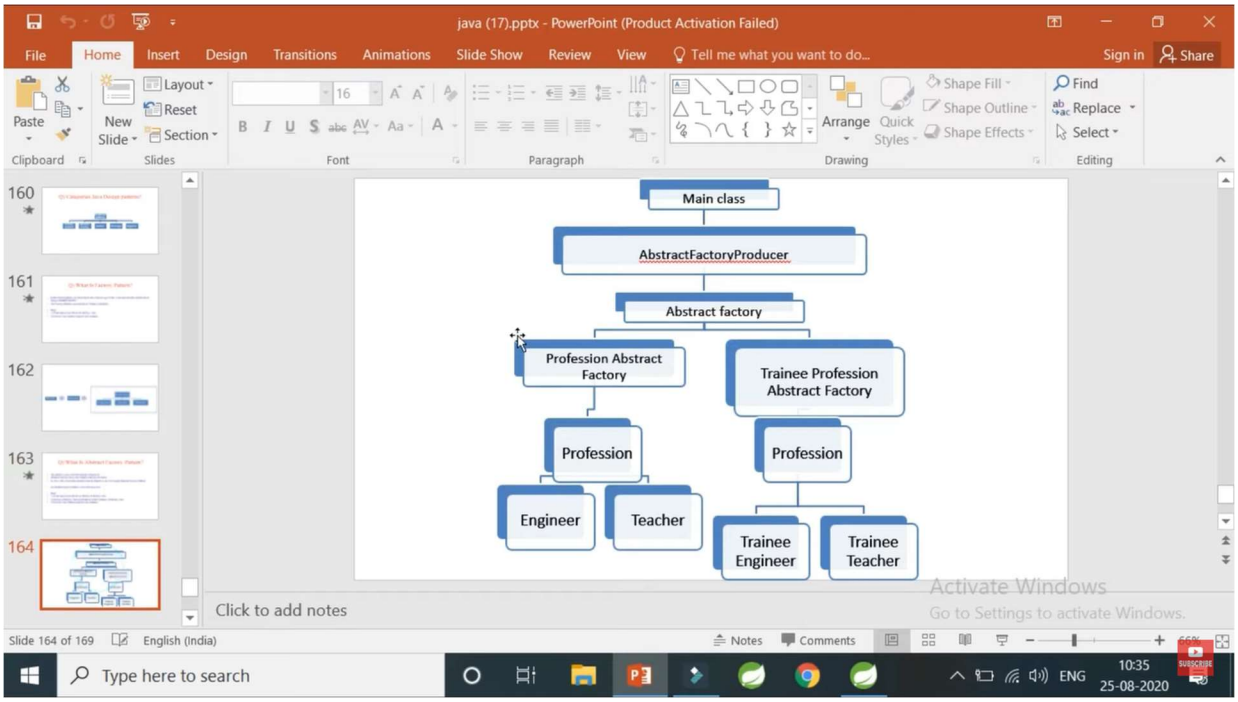
EX-



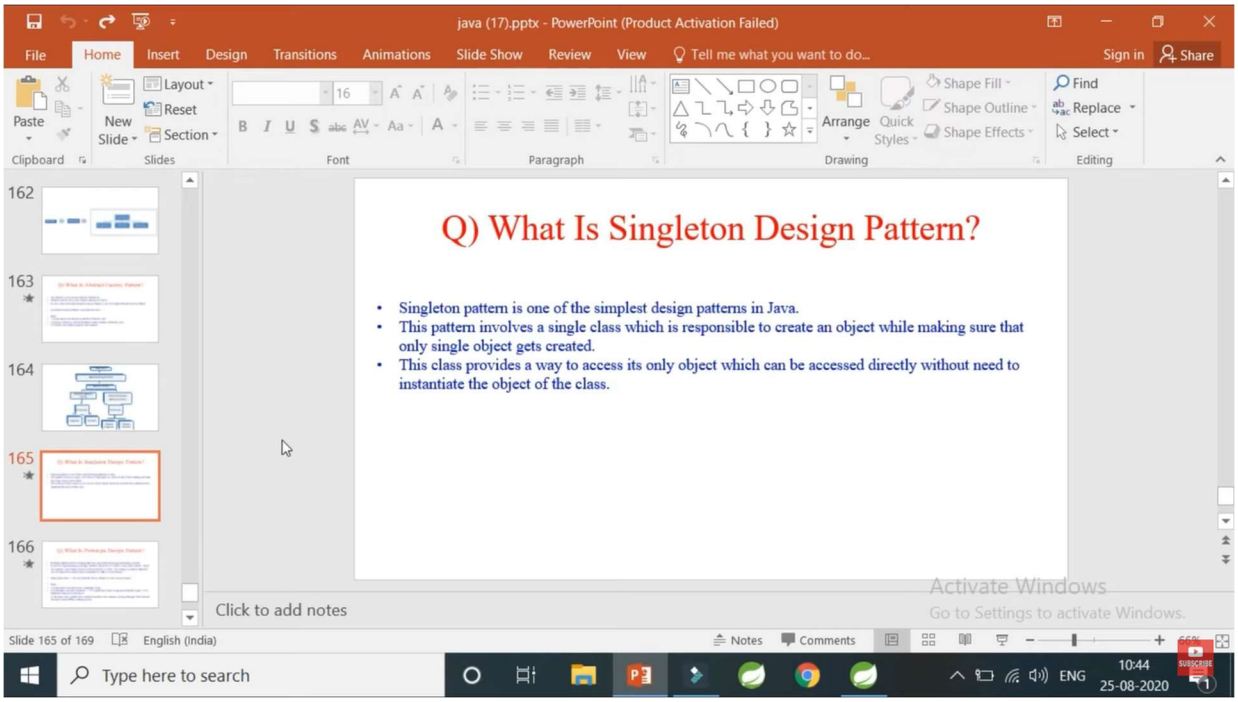
(51) WHAT IS ABSTRACT FACTORY PATTERN?



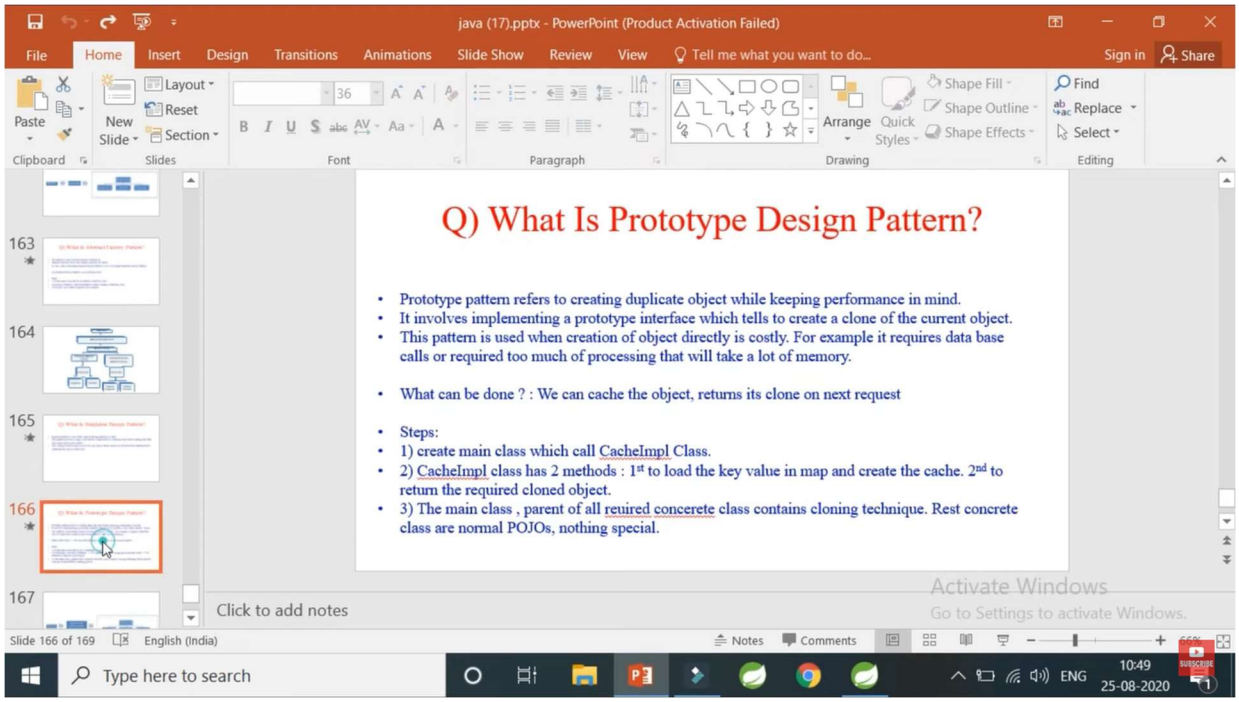
EX-



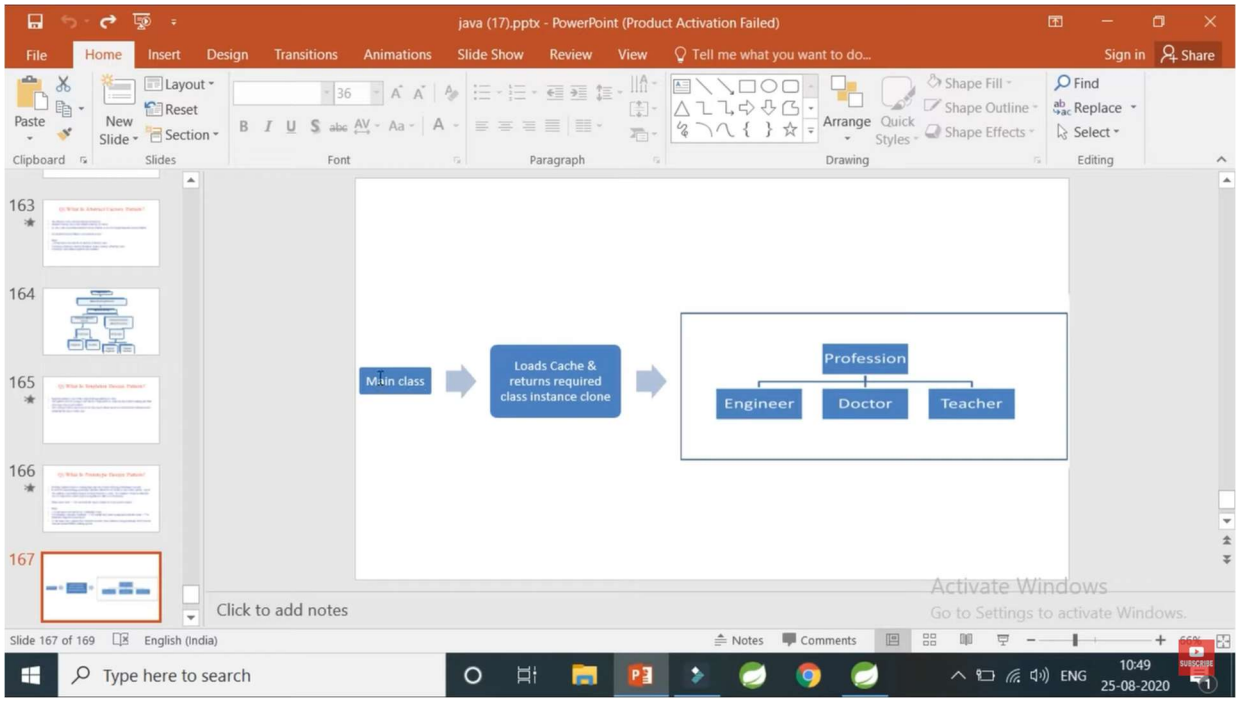
(52) WHAT IS SINGLE TON DESIGN PATTERN?



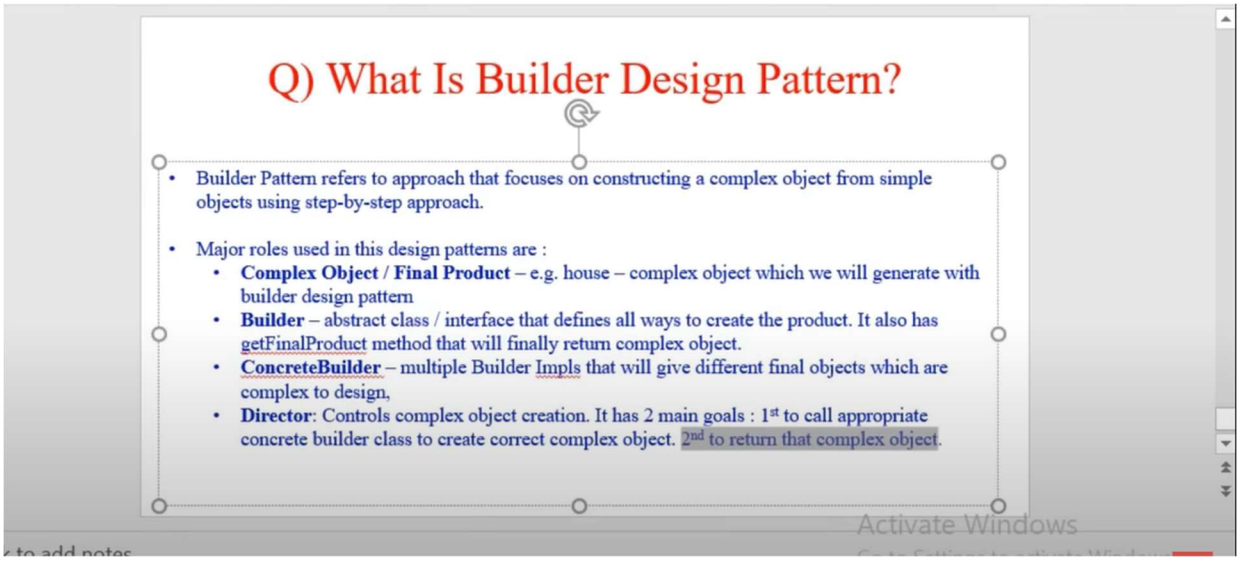
(53) WHAT IS PROTOTYPE DESIGN PATTERN?

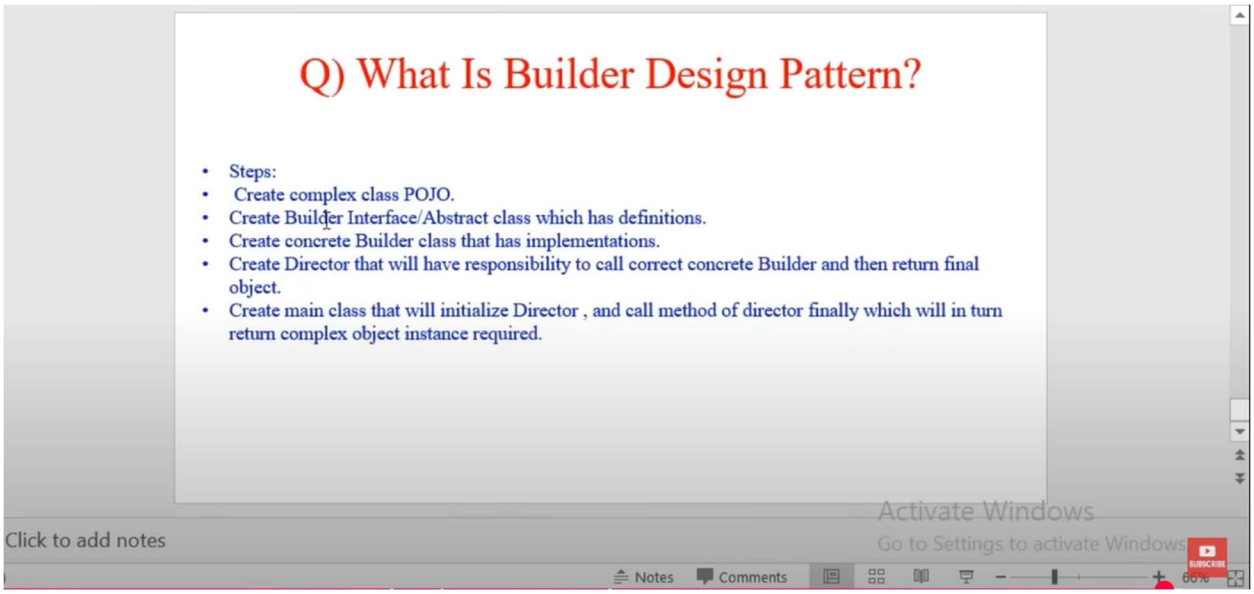


EX-



(54) WHAT IS BUILDER DESIGN PATTERN?





(55) CAN WE USE STATIC VARIBALE INTO NON-STATIC METHOD?

Yes.

(56) CAN WE USE NON-STATIC VARIABLE INTO STATIC METHOD?

Yes, accessing non-static variables from a static method is possible but generally not considered good practice.

(57) TELL METHODS NAME PRESENT IN OBJECT CLASS?

> Clone()

> equals()

> finalize()

> getClass()

> hashCode()

> toString()

> notify()

> notifyAll()

> wait()

(58) WHY WE CREATE DEFAULT CONSTRUCTOR?

The default constructor is used to initialize an object with default values, because When a class has parameterized constructors, Java does **not provide the default constructor** automatically.

(59) IF CONSTRUCTOR & SETTER DO THE SAME WORK THEN WHY CONSTRUCTOR OR SETTER?

Aspect	Constructor	Setter
Purpose	Used to initialize an object immediately when it is created.	Used to set or modify properties of an object after it has been created.
Reusability	Creates a new object every time it's invoked.	Can reuse the same object and modify its state.

(59) WHY JAVA IS NOT 100% OBJECT – ORIENTED ?

Because of primitive data types named like, Boolean, char, int etc to make them object oriented we have wrapper classes which actually wrap the primitive data type into object of that class.

(60) WHY POINTERS ARE NOT USED IN JAVA?

Because JVM is responsible for implicit memory allocation. To avoid direct access to memory by user.

(61) WHAT IS JIT /JIT COMPILER IN JAVA?

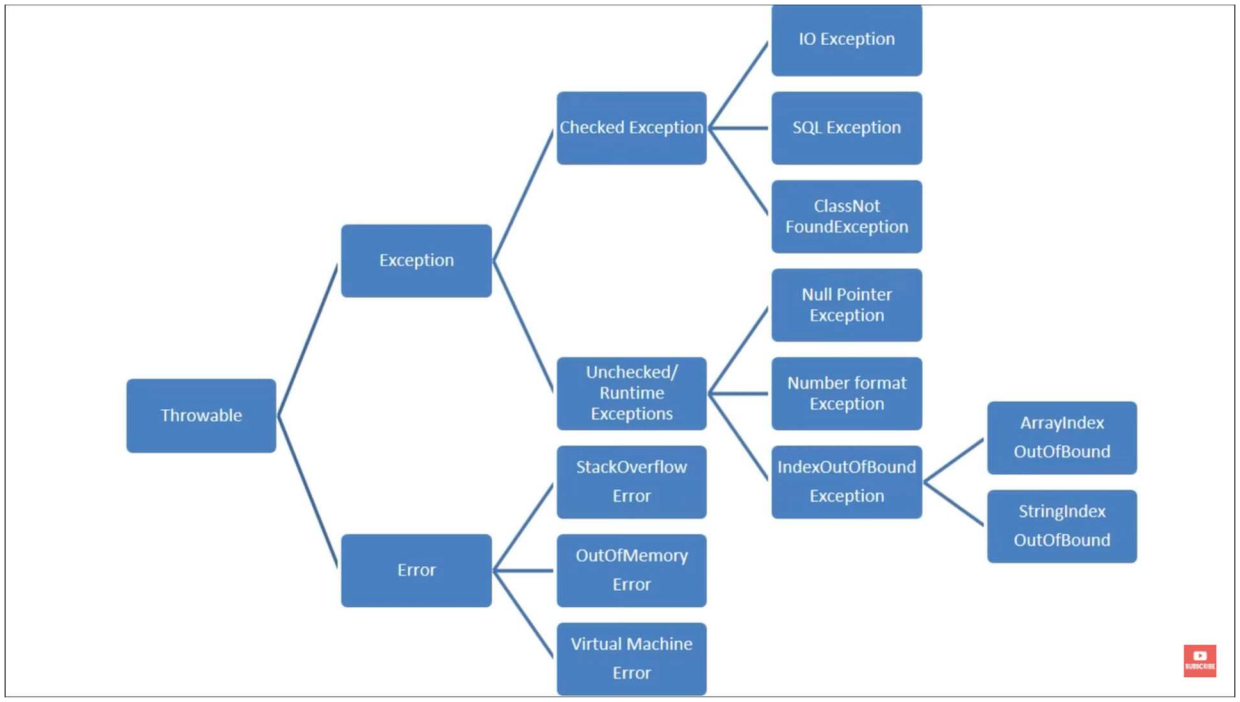
It reduces time in terms of converting byte code into machine code.because interpreter convert line by line .

(62) Can we restrict visibility of derived method in Java ?

you can’t make it less accessible or restricted but You can make it more accessible.

EXCEPTION

(1) EXPLAIN HERIRARCY OF EXCEPTION &THROWABLE?



THROWABLE:

Throwable is the parent class of all exceptions and errors , it can be used to create custom exception by extending it. It has two sub classes exception and error.

(2) DIFFERENCE BETWEEN EXCEPTION AND ERROR?

Aspect	Exception	Error
Definition	Represents a condition that Can be handled using `try-catch` blocks or by declaring `throws`.	Represents serious issues that occur outside the application's control, usually JVM-related. Usually not handled within the application code.
Common Causes	Invalid user input, file not found, incorrect database connection, etc.	Infinite recursion, insufficient memory, hardware failures, etc.
Scope of Occurrence	Application-level issues	System-level or environment-level issues.

(3) HOW MANY TYPES OF EXCEPTION EXPLAIN FEW EXCEPTION OF DIFFERENT TYPES?

For Ans look into hierarchy of exception.

(4) WHAT IS TRY CATCH BLOCK ?

Try catch block is a basic exception handling mechanism in java, which allows to handle exception which might be thrown in your code.

(5) CAN WE WRITE MULTIPLE CATCH BLOCK WITH SINGLE TRY BLOCK?

Yes, we can write multiple catch blocks with a single try block in Java but Order of `catch` Blocks will be first-matching approach then parent class like exception or throwable.

(6) CAN WE WRITE FINALLY BLOCK WITHOUT CATCH BLOCK?

Yes, you **can write a `finally` block without a `catch` block** The **`finally` block** is always executed after the `try` block, regardless of whether an exception is thrown or not.

(7) CAN WE WRITE TRY BLOCK WITHOUT CATCH OR FINALLY BLOCK?

No, **a `try` block cannot exist without either a `catch` block or a `finally` block.**

(8) HOW MANY WAYS TO HANDLE EXCEPTION IN JAVA?

there are **two primary ways** to handle exceptions:

- Using `try-catch` Block.
- Using `throws` Keyword.
- Using `finally` Block: Additional Techniques for Exception Handling because The `finally` block is used to execute cleanup code (like closing resources) regardless of whether an exception was thrown or not. It works with `try` and optionally `catch`.

(9) EXPLAIN TRHOW VS THROWS KEYWORD?

Aspect	throw	throws
Purpose	Used to explicitly throw an exception.	Used to declare exceptions that a method might throw.
Usage	Inside a method or block.	In the method signature.

Number of Exceptions	Can throw only one exception at a time.	Can declare multiple exceptions separated by commas.
Example	throw new IOException("Error");	public void method() throws IOException, SQLException { }

(10) WHAT IS EXCEPTION CLASS?

It is the parent class of all exceptions which can handle any types of exception where as a particular exception handler like , numberformat, nullpointer or other can handle only its exception.

(11) HOW EXCEPTION HANDLE BY TRY CATCH BLOCK?

We write the codes inside try block if any exception cause in try block then try block create exception object and then that objects reference passed to the catch block and catch block suppress the exception and further code run continue.

(12) WHAT IS PRINT STACK TRACE()?

printStack trace is a method which give us exact line where exception is occurs.

(13) WHAT IS TRY WITH RESOURCE?

The try-with-resources statement was introduced in Java 7 and allows automatic closing of resources (such as files, network connections, database connections, etc.) that implement AutoCloseable. When you use this mechanism, resources are automatically closed at the end of the try block, even if an exception occurs.

EX-

```
import java.io.*;


public class TryWithResourcesExample {

    public static void main(String[] args) {

        // The try-with-resources statement automatically closes the resource (file reader) at the end.

        try (BufferedReader br = new BufferedReader(new FileReader("example.txt"))) {

            String line;

            while ((line = br.readLine()) != null) {

                System.out.println(line);

            }
```

```
            } catch (IOException e) {
                  e.printStackTrace();
            }
      }
}
```

(14) TRY WITH RESOURCE WHICH ONE INTERFACE USED?

AutoClosable.

(15) CAN WE WRITE ANY OTHER STATEMENTS BETWEEN TRY CATCH OR FINALLY BLOCK?

No , try must be followed directly by either catch or finally.

(16) DOES REMAINING STATEMENTS IN TRY BLOCK EXECUTES AFTER EXCEPTION OCCURS?

No, if exception occurs at a particular point in try block, then all statements after that statement where exception Is occurred will not be executed and flow goes directly to either catch block or finally block.

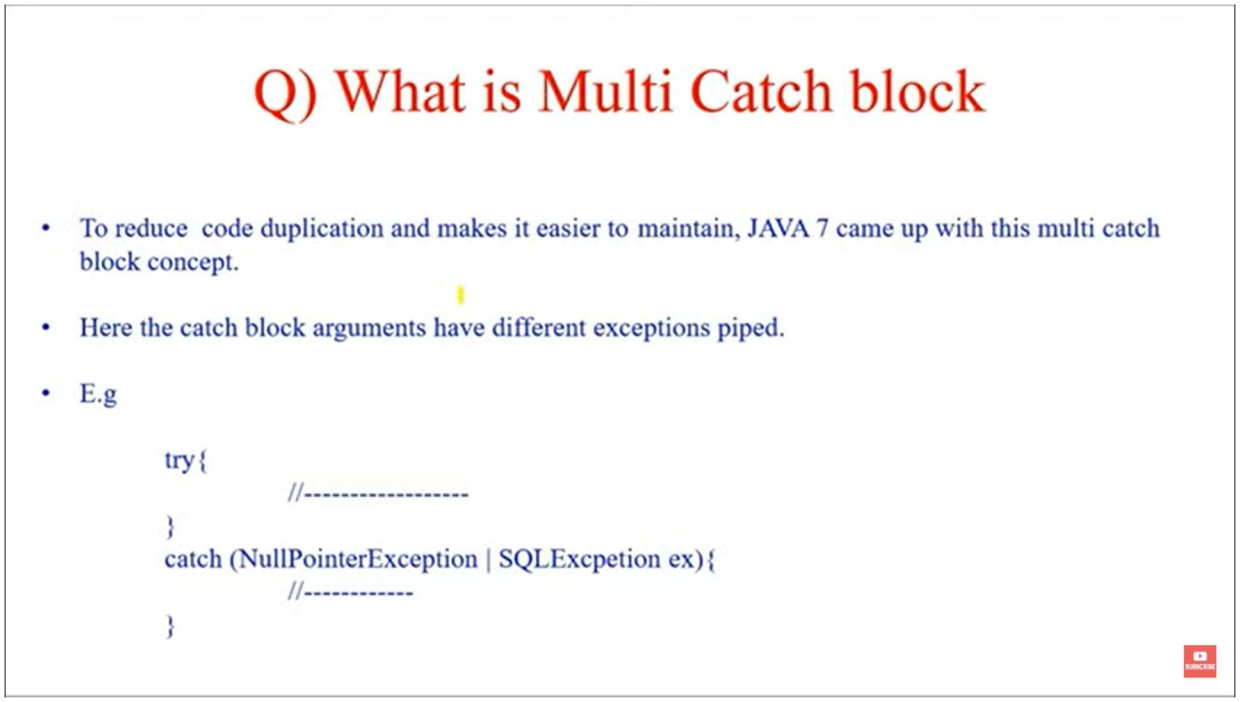
(17) WHAT HAPPENES WHEN AN EXCEPTION IS THROWN BY MAIN THE MAIN METHOD ?

When an exception is thrown by the main method , java runtime (JRE) terminates the programs and print the exception message and stack trace in-system console.

(18) WHAT DO YOU UNDERSTAND BY UNREACHABLE CATCH BLOCK ERROR?

This error comes when you keep super class exception first(Exception e) , and sub class exception later (NullPointerException). Hence the order of catch blocks must be from most specific to most general one.

(19) WHAT IS MULTI CATCH BLOCK ?



STRING

(1) WHAT IS STRING?

String is a final class in Java (java.lang.String). It represents a sequence of characters/ text
String is often treated as a primitive data type . string is immutable

(2) HOW MANY WAYS TO CREATE STRING?

There are two ways to create string :

- **String Constant Pool (SCP)**:
  When a string literal (e.g., `"Hello"`) is created, it is stored in a special memory area called the String Constant Pool. If another string literal with the same value is created, it will reference the same object instead of creating a new one.

  Ex –

  String s1 = "Hello";

String s2 = "Hello";

System.out.println(s1 == s2); // true (same object in SCP)

- **Heap Memory**:
  Strings created using the `new` keyword are stored in heap memory. They do not share references with existing literals in the SCP.

  Ex –

  String s1 = new String("Hello");

  String s2 = new String("Hello");

  System.out.println(s1 == s2); // false (different objects in heap)

(3) WHAT MUTABLE AND IMMUTABLE REFERS IN TERMS OF STRING?

Mutable :

Mutable is something where keeps object state changing.

Immutability:

Once a String object is created, its value cannot be changed.

(4) WHY STRING IS IMMUTABLE?

- **String Pool Optimization**:
  Immutability allows sharing of strings in the String Constant Pool, which saves memory and avoids duplication.
- **Thread Safety**:
  Since strings cannot be modified, they are inherently thread-safe.
- **Security**:
  Strings are used in many sensitive areas, such as file paths, network connections, and keys in hashmaps. If strings were mutable, malicious code could alter them.

(6) STEPS TO CREATE IMMUTABLE CLASS IN JAVA?

Steps:

- Declare the class as `final`.
- Mark all fields as `private` and `final`.
- Initialize all fields using private constructor
- Do not provide setters.

- Provide only getters for the fields.

(7) DIFFERENCE BETWEEN EQUALS METHOD AND EQUAL OPERATOR?

- `==` Operator: Compares **references** for objects or **values** for primitives.

  Ex-

  For primitives,

  int a = 10;

  int b = 10;

  System.out.println(a == b); // true (same value)

  Ex-

  For object reference,

  String s1 = new String("Hello");

  String s2 = new String("Hello");

  System.out.println(s1 == s2); // false (different memory locations)

- `equals` Method: Compares the **contents (values)** of two objects for equality. By default, the equals method in the Object class behaves like == (checks memory references). However, many classes (e.g., String, Integer, etc.) override equals to compare logical equality (i.e., the content).

  Ex-

  String s1 = new String("Hello");

  String s2 = new String("Hello");

  System.out.println(s1.equals(s2)); // true (contents are the same)

(9) EXPLAIN METHOD LIKE, CHARS(), CHARAT(), VALUEOF()?

String Methods are:

- Length(): Returns the number of characters in the string.

  Ex-

  String s = "Hello";

System.out.println(s.length()); // 5

- CharAt(): Returns the character at a specified index.

  Ex-

  System.out.println(s.charAt(1)); // e
- substring(): Extracts a portion of the string.

  Ex-

  System.out.println(s.substring(1, 4)); // ell
- concat(): Concatenates two strings.

  Ex-

  String s2 = "World";

  System.out.println(s.concat(s2)); // HelloWorld
- equals(): Compares two strings for equality.

  Ex-

  String s3 = "Hello";

  System.out.println(s.equals(s3)); // true
- EqualsIgnoreCase(): Compares two strings, ignoring case.

  Ex-

  String s4 = "hello";

  System.out.println(s.equalsIgnoreCase(s4)); // true
- indexOf(): Finds the index of a character or substring.

  Ex-

  System.out.println(s.indexOf('e')); // 1
- toUpperCase()/toLowerCase(): Converts the string to upper/lower case.

  Ex-

  System.out.println(s.toUpperCase()); // HELLO
- trim(): Removes leading and trailing spaces.

  Ex-

  String s5 = " Hello ";

System.out.println(s5.trim()); // "Hello"

- replace(): Replaces occurrences of a character or substring.

  Ex-

  System.out.println(s.replace('l', 'p')); // Heppo

- split(): Splits the string into an array of substrings based on a delimiter.

  Ex-

  String s6 = "Java,Python,C++";

  String[] parts = s6.split(",");

  for (String part : parts) {

  System.out.println(part);

  }

- contains(): Checks if a substring is present.

  Ex-

  System.out.println(s.contains("ll")); // true

- startsWith()/endsWith(): Checks if the string starts or ends with a specific substring.

  Ex-

  System.out.println(s.startsWith("He")); // true

  System.out.println(s.endsWith("lo")); // true

- valueOf(): Converts other data types to string.

  Ex-

  int num = 10;

  System.out.println(String.valueOf(num)); // "10"

- chars(): The `chars()` method of `String` returns an `IntStream` of Unicode code points representing each character in the string.

  Ex-

  "Hello".chars().forEach(ch -> System.out.print((char) ch)); // Output: Hello

(10) EXPLAIN DIFFERENCE BETWEEN STRING, STRINGBUILDER AND STRINGBUFFER?

Aspect	String	StringBuilder	StringBuffer
Mutability	Immutable (any modification creates a new object).	Mutable (modifications occur in the same object)	Mutable (modifications occur in the same object).
Thread Safety	Not thread-safe.	Not thread-safe.	Thread-safe (synchronized methods).
Memory Usage	Creates a new object for each modification.	Uses the same object for modifications (efficient memory usage).	Uses the same object for modifications (efficient memory usage).
Speed	Slow for string manipulations.	Faster for string manipulations in single-threaded applications	Slower than `StringBuilder` due to synchronization.

(11) WAYS TO CREATE STRINGBUILDER & FEW METHODS OF IT?

WAYS TO CREATE STRINGBUILDER:

- Default constructor.
- Constructor with initial capacity.
- Constructor with string input.

METHODS:

Method	Description	Example
append(String s)	Adds the given string (or other data types) to the end of the current sequence.	`sb.append(" World");` → Appends " World" to `sb`.
insert(int offset, String s)	Inserts the given string at the specified position (offset).	`sb.insert(5, " Java");` → Inserts " Java" at index 5.
replace(int start, int end, String s)	Replaces the characters in the specified range with the given string.	`sb.replace(0, 5, "Hi");` → Replaces characters at index 0–4 with "Hi".
delete(int start, int end)	Removes the characters in the specified range.	`sb.delete(0, 5);` → Removes characters at index 0–4.
deleteCharAt(int index)	Removes the character at the specified index.	`sb.deleteCharAt(0);` → Removes the character at index 0.

reverse()	Reverses the sequence of characters in the `StringBuilder`.	`sb.reverse();` → Reverses the content.
charAt(int index)	Returns the character at the specified index.	`sb.charAt(4);` → Returns the character at index 4.
substring(int start, int end)	Returns a substring from the specified start to end indices (similar to `String`)	`sb.substring(0, 5);` → Returns the substring from index 0 to 4.
length()	Returns the number of characters in the `StringBuilder`.	`sb.length();` → Returns the length of the sequence.
toString()	Converts the `StringBuilder` content into a `String`.	`String s = sb.toString();` → Converts the sequence into a `String`

(12) WHAT DOES THE STRING INTERN() METHOD DO?

The task of intern() method is to put String (which is passed to intern method) into the string constant pool.

When the intern method is called, if the string constant pool already contains a string equal to the string object then the string from the pool is returned. Otherwise the string object is added to the pool , and reference to the string object is returned.