* OOPS Info
* Inheritance
* Type – Casting
* Access-Modifiers
* Final keyword(interview questions in eclipse)
* Super and this keyword
* Polymorphism
* Abstraction(interview questions in eclipse)
* Encapsulation
* String class
* String builder and buffer
* Exception handling
* OOPS(Object – oriented programming/paradiagram system)

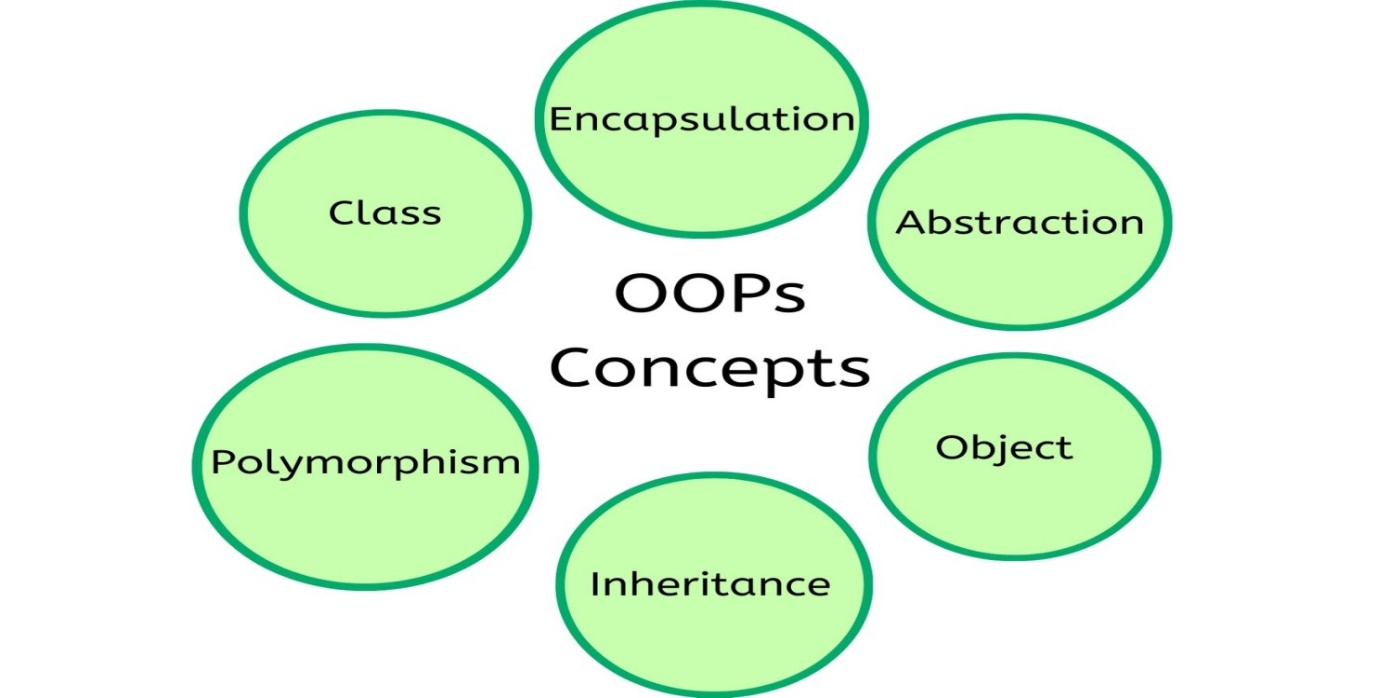
The programming paradigm where everything is represented as an object is known as oop language.

Java is not 100% object oriented because of primitives dataypes

Because primitives ka hum object nhi create kar sakte and oops pura classes or objects mai depend hota hai

Q:Why we need oops?

Tomanage our code (student managemnet system) and code reusability (using methods of addition)



* Object
* An object can be defined as an instance of class.

Or

* Any entity that has state and behavior is known as object.

for example, bike, pen.

Real-world objects

* state- represent the data(value) of an object – eg: variables
* Behavior –represent the functionality of an object eg: methods like walk run etc
* Identity – a unique id.(Hashcode) it is used internally for jvm to identify the object.
* Class

Collection of object is called class. It is called logical entity.

A class is a template or blueprint from which object are created.

A class contains

* Fields
* Methods
* Constructors
* Blocks
* Nested Class and Interface.
* Inheritance:
* Inheritance in Java is a mechanism in which one object acquires all the properties and behaviors of a parent objects.
* Inheritance represents the IS-A relationship

Direct relationship

i.e parent child relationship

* using extends keyword we can achieve inheritance
* A class which inherit the properties is called as –child class,sub class
* A class from which inherited is called – parent class,super class
* We cant inherit constructors (Because constructor similar hota hai uske class name se)
* We cant override variables
* We cant overide static method (it refers to the class itself)
* We cant ovveride constructor (Because constructor same class mai hi banta hai so overid e nhi hoga)
* Inheritance: Supports tight coupling means if we change parent behaviour it will refelect in thr child
* Uses

-for code reusability

- for method overriding.

Types of inheritance in Java:

Java suports 3 types aisa inheritance 5 types ka hai

* Single inheritance

Supported by java



* multilevel inheritance



* hierarchical inheritance

1. Single Inheritance :

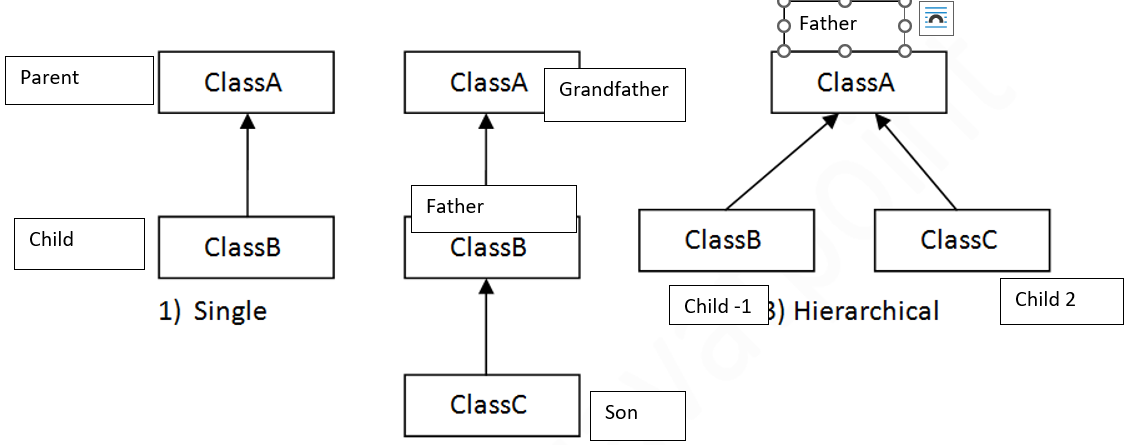
When a class inherits another class, it is known as a single inheritance.

1. Multilevel inheritance :

When there is a chain of inheritance, it is known as multilevel inheritanc

1. Hierarchial inheritance :

When two or more classes inherits a single class, it is known as hierarchical inheritance



* Multiple inheritance

Not supported

* Hybrid inheritance



If we write parameterized constructor. In parent

Because, we are not defining constructor for Class B. So, compiler will be providing default constructor. In that default constructor, first statement is super() – it is a calling statement to default constructor of Class A. But it is not defined in Class A. Therefore you will get a compile time error. To avoid this error, write the constructor for sub class.

* Type Casting:

Type casting is when you assign a value of one primitive data type to another type.

1. Widening Casting (automatically)/Implicit Casting

* Converting a smaller type to a larger type.
* Done automatically by java. Because data lose nhi hota
* byte -> short -> char -> int -> long -> float -> double

1. Narrowing Casting (manual)/Explicit Casting

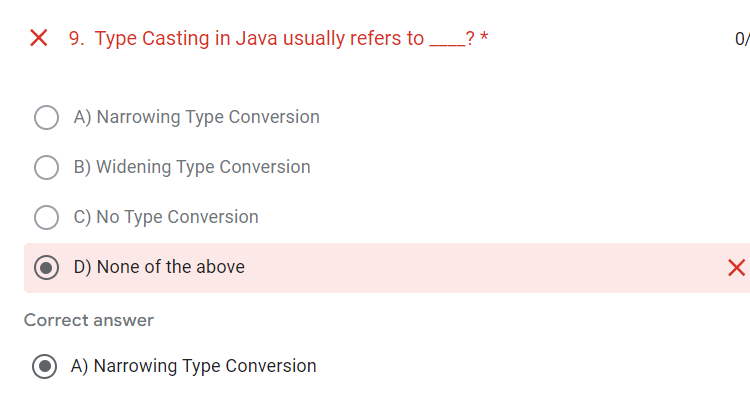
* Done manually by us
* Lose of data
* double -> float -> long -> int -> char -> short -> byte

REAL LIFE SCENERIO OF CASTING:

In calculator program user int value daal rha hai but in case of divide value can be in point . so we convert int to divide

NOTE: If we cast char in any datatype we get its ASCII value (Used to represent text,letter,symbol).

ASCII -> Number defined .



* Covariant returntype (Interview Question):
* Introduce in jdk 1.5
* Here we can make returntype as parent and we can return different child against it.
* Established only with inheritance.

Q.Why it was introduced?

Till 1.4 ----

-> As we know if we use inheritance thane we getXX parent + child both data so the concept of not allowing to return child data was conflicting with the inheritance concept . hence it was introduced

NOTE:

* We cant perfrom explicit casting without covariant type .
* If we did directly it will give as runntime error but wont show error in compile time
* Widening can be done via directly or by covariant.
* Access Modifier

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Fields | Public | Protected | Default | Private |
| Within class |  |  |  |  |
| Within Package |  |  |  |  |
| Outside Package(Subclass) |  |  |  |  |
| Outside Package(Superclass) |  |  |  |  |

1. Public :

Declaration can be done : variables, method ,class Constructor

Accesible to everyone even to another application but we need to upload the jar file to another class

If the class we declare is public than the class name and file name should be same

One java file should contain only one or zero public class. It should not contain more than one public class.

1. Protected : Outside pacakage rules

Through inheritance (P-c relation)



Only child class object



Declaration can be done : variables, method ,Constructor

1. Default:

If we don’t use any access modifier than by default consider as default

We don’t need to write any keyword as default

If we write it will give error

Declaration can be done : variables, method ,class Constructor

1. Private :

Declaration can be done : variables, method ,Constructor

Our internal data should not go out directly that is outside person can't access our internal data directly.

 By using private modifier we can implement data hiding.

Q:Define largest scope and smallest scope

Smallest Scope : Private

Largest Scope : Public

Q:What will be the access modifier of an constructor

Depends on class access modifier

Q:What happen if we declare constructor as private and protected

If we declare constructor has private or protected we cant make object from another class

private 🡪Because scope is within the class

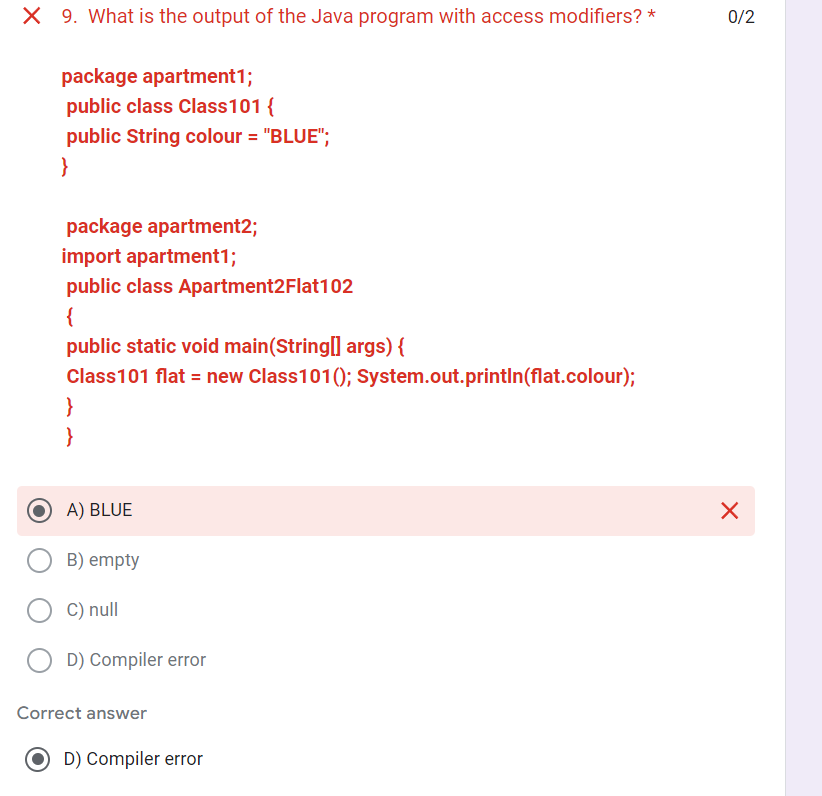
protected -> Because as we know protected needs child class so that is why we cant make its object (parent class object).

Q:Why we cant write private and protected as class

Private 🡪 will hide the class from other class

Protected 🡪 If class protected than it will only be accessible to sub class only .

Q:Why main method is public

So that it can be accesible to other classes as well as to the jvm .

* Final keyword: (More in concept of the day)
* Final = constant
* Restricts further modification
* We can get the data but cant change it.
* basically it is use for to restrict the user data.
* we can use final keyword before

-class

-data member (variable)

-method

Variables : needs to intiliaze

Value doesn’t change

Only getter hota hai

Class : We can’t extend i.e no inheritance

Eg: String class

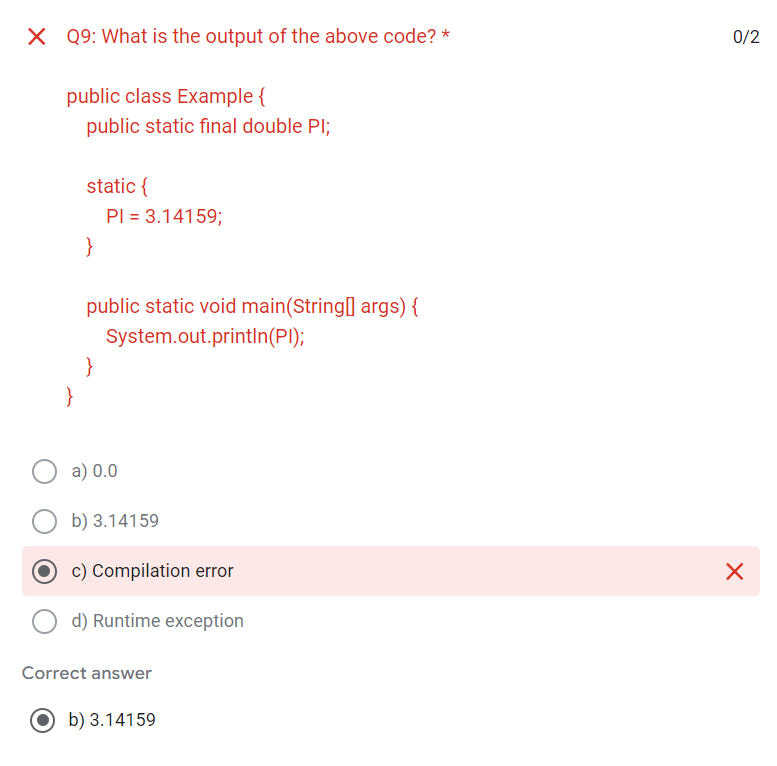
Methods : We cant override in subclass

Q:Why we cant make constructors as final

Because it doesn’t inherit and final keyword doesn’t change the logic or inherit nhi hoga toh logic kaisa change karege

Q:Can we make immutuable class of your own

Yes by decalring calss as final its varibale as final , methods as final .



Finalize() ->

* this is a method
* Before going to destroy any unused method will be called.

10 points related to final from concept of the day final keyword (imp)

**7)** If the global variables are not initialized explicitly, they get default value at the time of object creation. But final global variables don’t get default value and they must be explicitly initialized at the time of object creation. Uninitialized final field is called **Blank Final Field**.

**7)** If the global variables are not initialized explicitly, they get default value at the time of object creation. But final global variables don’t get default value and they must be explicitly initialized at the time of object creation. Uninitialized final field is called **Blank Final Fiel**

**class** A

{

**int** i;   //Non-final global variable, no need to initialize them

**final** **int** j;         //Blank Final Field

    A()

    {

        j=20;

        //final global variable must get a value at the time of object creation.

    }

}

**public** **class** UseOfFinalKeyword

{

**public** **static** **void** main(String[] args)

    {

        A a = **new** A();

    }

}

* Super and this keyword: (Always use for data calling)(90% for constructor chainig)

1. Super : Refer to immediate parent class

We need inheritance

Why it was introduce?

Because as we know jab overide hota hai koi method toh hume overide method milta hai from child and p-c mix object toh agar usma hume parent class ka method call karna hai tab super use hota hai or else when we need to call the instance variable of parent class because varible bhi override nhi hota or object create kiya to h uss class mai jo var declare kiya who hoga print if varibale is of same name

uses:

* refer immediate parent class instance variable (global variable )
* invoke the parent class method

invoke the parent class constructor

1. this : current object

uses:

* instance variable
* current class method
* current class constructor
* eliminate confusion between local and global in any blocks .

NOTE : IF WE CALL ANYTHING INSIDE A CLASS WHETHER WE HAVE WRITTEN OR NOT THE THIS KEYWORD IS BYDEFAULT PRESENT.

* Polymorphism

If one task is performed in different different ways, it is called polymorphism.

That is, the same entity (method or operator or object) can perform different operations in different scenarios.

**Real-life Illustration Polymorphism: A person at the same time can have different characteristics. Like a man at the same time is a father, a husband, and an employee. So the same person possesses different behavior in different situations. This is called polymorphism.**

There are two types of polymorphism in Java.

* Static polymorphism / method overloading / static binding / early binding / compile time polymorphism.
* Dynamic polymorphism / method overriding / dynamic binding /late binding /run time polymorphism.(runtime mai samjta hai konsa method call huv hai )

1. Method Overloading:

We can have different forms of same method in the same class is called method overloading.

Requirement for the method overloading:

* method name same and parameter is always different.

Rule for method overloading:

* method overloading always happen in the same class.
* access modifiers doesn’t matter in the method overloading.
* return types doesn’t matter in the method overloading.
* we can overload main, static, final and private method in java.
* Different datatypes

JAVA EXAMPLE OF METHOD OVERLOADING:

**System.out.println()** 🡪 here println() use method overloading.

**System** :

.class present in java.lang package

It is final class i.e why It has private constructor (We cant make mainits object outside the class)

**out :**

variable (i.e obj) of printstream class which is written in System class as :

**static Printstream out** 🡪 static because we cant create its object to accisible to other class.

**println():**

method written in Printstream class

it has 13 methods 🡪 to print different different int,string,object.

NOTE:

Hum main method ko bhi overload kar sakte hai but jisma string hai wahi run hoga hum uske andar se who jo hume main overload kiya usko run kar sakte.

1. Method Overriding:

When a super class method is modified in the sub class, then we call this as method overriding.

Requirement for the method overriding:

* Need parent child relationship.(P-C Reln)

Rules for method overriding:

* Need parent child relationship.
* the method parameter is always same.
* must have same return type or covariant return is also work.
* access modifiers same or greater than.
* we can’t override private, final, and static method.

Q:why static methods donot override?

Because it follows compile time polymorphism . because it refers to class.

Q:Why we cant overide variables?

In Java, variables cannot be overridden because overriding is a concept that applies to methods, not variables. Methods define behaviors that can be replaced in subclasses to provide specific implementations. Variables hold data and can be hidden by redeclaring them in subclasses with the same name, but this doesn't change the original variable's behavior in the superclass.

* Abstraction

Hiding internal details and showing functionality is known is abstraction.

For example: sending text message.

Real time definitions: Showing essential data at essential time is known as abstraction.

for example: student example (i.e. 10th mark,12th mark, height, weight)

* **By using abstract class and Interface we can achieve abstraction in java.**

1.Abstract class:

A class which is declared as abstract is called as **abstract class.** It can have abstract and non-abstract methods.

Matlab khuch cheeze hum hide kar sakte or khuch nhi



* By using abstract class, we can achieve partial abstraction
* It can have abstract(non-concreate) and non-abstract method (concreate method)
* It cannot be instantiated

Cannot create object

* We can only make methods and class as abstract.
* We mostly prefer parent child mix object
* If a class contain at least on abstract method then compulsory the corresponding class should be declare with abstract modifier.
* We can only use public , private modifier for abstract methods
* We can’t create objects to those classes which are declared as abstract. But, we can create objects to sub classes of abstract class, provided they must implement abstract methods.

Q:Why to use abstract class?

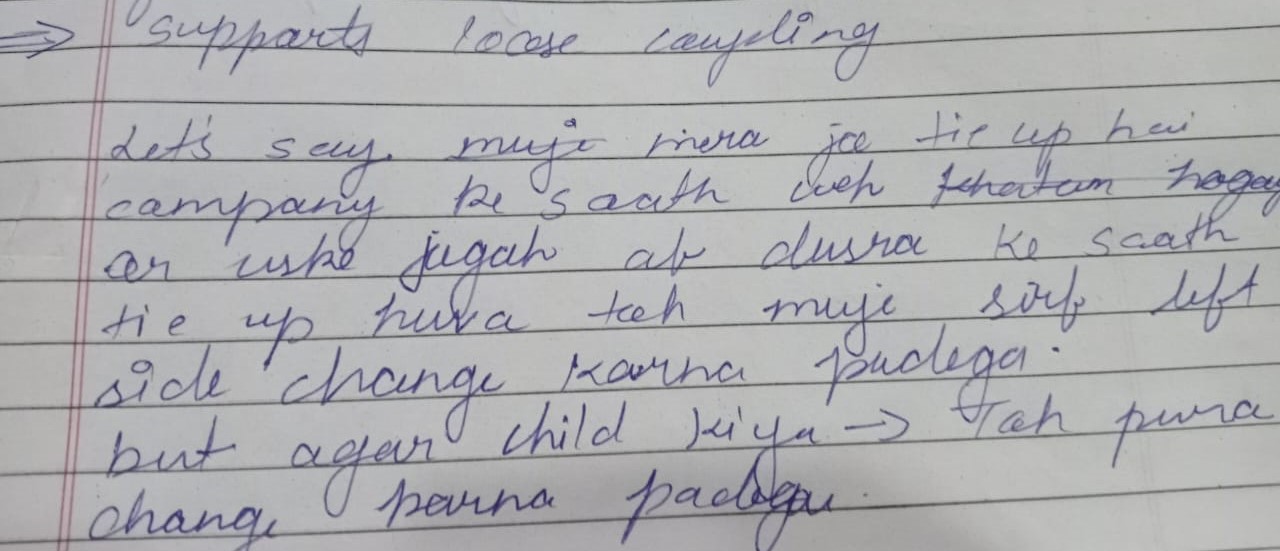
When we know half implementaion and we don’t know half implementation .

Q:Why we cant make constructor as abstract?

Because abstract mai hume dusra class means child class mai implementaion likhte hai or constructor toh overide nhi hota toh abstract likha matlab nhi .

Q:Why to use P-C mix object ?

Because agar future mai agar changes huve toh right side change kiya toh bhi implementation wahi hoga



Q:Why we cant instantiate abstract class via parent object

As we know if we need object we need complete methods .

Q: Why there is constructor present in abstract class ?

Because as we know there are both abstract and non-abstract

So non-abstract methods ko call karneko

Q: why no static methods?

Because static methods donot override.

2.Interface:

An interface in Java is blueprint of a class.

* Interface is used to defined user defined data type (Only variables ,methods)
* Interface ka heirarchy khud interface se start hota hai
* By using interface, we can achieve 100% abstraction.
* By using interface, we can achieve multiple inheritance in java
* Interface is also representing the IS-A relationship
* Since java 8 we can have default and static method in the interface
* Since java 9 we can have private method in an interface

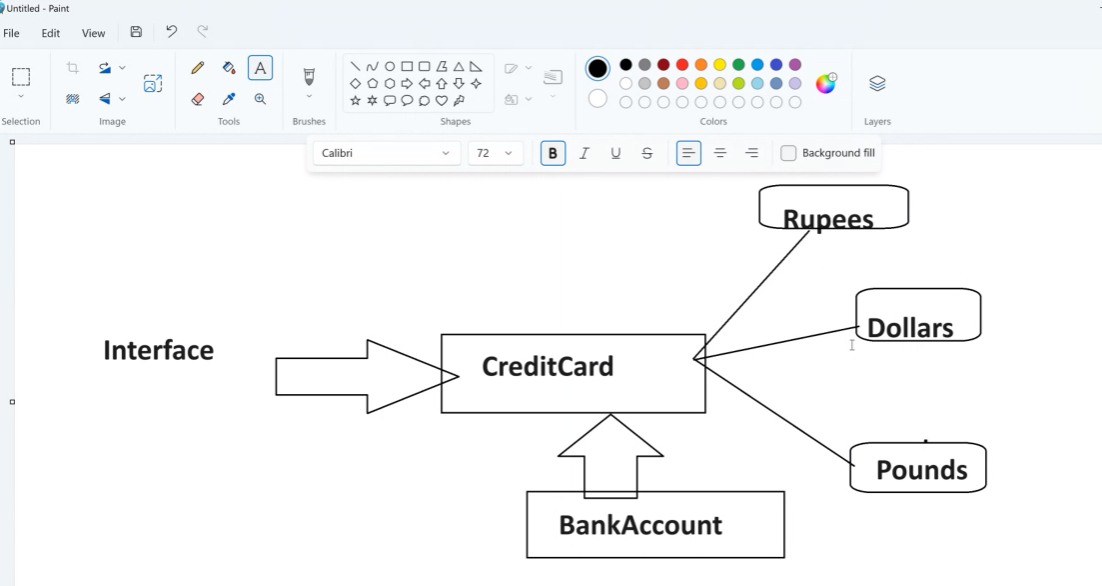
Uses of Interfaces:

-to achieve 100% abstraction

-support multiple inheritance

-achieve loose coupling

REAL LIFE EXAMPLE



NOTE:

* Class implements interface
* Class extends class
* Interface extends interface

Interface implements class x not possible because abstract hai.

RULES FOR INTERFACE:

Varibales-> By default it is public static final //jvm.

Methods-> By default it has public abstract.

Child class must declare the implemntation or else we need to make the child class also abstract

Q:We don’t have default constructor

Because varibale final decalre karte hai hum jo hum decide karoge.

Q:Why interface?

Use by the architecture team to show us what methods,fields should be present in java

Q:Why varibale is static ?

Because we cant make its object.

Q:By default how is abstract class?

Public abstract.

Difference between abstract class and interface

|  |  |
| --- | --- |
| Abstract class | Interface |
| default const present | no default const |
| public protected private | public static final |
| 0-100 % abstract | 100% abstraction |
| abstract and implemented method | only abstract method(jdk 1.7) |
|  | static , default |

MARKER INTERFACE: an interface which doesn't contain any methods and by implementing that interface it provide some extra information to jvm

Eg: Serializable: serilization

Cloneable: cloning object

RandomAccess: randomly access index

* Aggregation/composition
* if a class have an entity reference() ->nonprimitive data type it is called as aggregation in java.
* Aggregation represent HAS-A relationship in java.

When we aggregations:

* Code reuse is also best achieved by aggregations, when there is no is-a relationship.

Eg: A a; here A is class and a is varible name .

Q:Why aggregation over inheritance?

Because follows loose coupling: Because there is no relation between classes only we use other class as an variable.so we can create refernce of two or more classes.

Inheritance: no multiple inheritance

* Encapsulation: (setter/getter)

Encapsulation is defined as the wrapping up of data under a single unit (class)is called encapsulation.

It is mechanism that binds together code and the data it manipulates.

Another way to think about the encapsulation is, it a protective shield that prevents the data from being accessed by the code outside this shield.

* Technically in encapsulation, the variables or data of a class is hidden from any other class and can be accessed only through any member from any other class and can be accessed only through any member function of its own class in which it is declared
* Real life example : The bag contains different stuffs like pen, pencil, notebook etc within it, in order to get any stuff you need to open that bag

Advantages of Encapsulation:

* **Data Hiding**: Keeping internal data private so it can't be accessed directly from outside the class
* Increased Flexibility
* Reusability
* Testing code easy
* **String classes: (Final)**

­String is a sequence of characters.

In java, objects of string is immutablewhich means a constant and cannot be changed once created.

present in java owns package i.e present in java.lang

They are thread safe.

This class implements serializable,comparable,charsequence

Immutable String : String ka content hum change nhi kar sakte hai or agar kiya toh naya string create hota hai.

Creating a String

There are two ways to create string in java

* String literal
* String s =”abc”;
* Used for memory efficiency:-Because same content ke liye baar baar memory nhi create karta uske refernce ko uss object (content ) ko point karta hai bas.
* Stored in constant pool
* One object one reference
* Using new Keyword

String s = new String(“Java”);

Create in heap memory

Two objects one refernce

Memory management of String

1: Storing the string :

Eg : String s1 = “Java”

String s2 = new String(“Java”).

Constant pool area

{Part of heap}

s1



“Java”

Heap memory



“Java”

s2

2: Same object name different refernce variable :

Eg : String s3 = “Java”

String s4 = new String(“Java”).

No new object created only new refernce

s1



“Java”



s3



“Java”

“Java”

s2



s4

Different object created



3:Case sensitive in literal :

New keywrod se object same ho ya different same case sensitive ho ya nhi hamesha naye memory mai create hota hai use liye

Eg : String s1 = “Java”

String s2 = “java” 🡪 new space allocated in constant pool because java is case sensitive

s1



“Java”

“java”

“java”



s2

Java String class methods:

trim() -> removes last and front whitespaces.

toUppercase() -> converts into capital letters

toLowercase() -> converts into smaller letters.

toCharArray ()- > converts the string into charcter array (save in char []).

charAt(index number) -> gives the character at that index.

indexOf (char) -> gives the character index if present or else if not than -1.

split() -> split string into sub string(save in String array )

length() -> give size of the string

valueOf ()-> give s string representaion in short saare datatypes ko string hi consider karta hai agar int pass kiya toh bhi string consider karega. Use to convert value to string also use value of is satic method of String so use String.valueOf();

replace(old char , new char) -> changing one char to another

old char 🡪 the character to replace

new char -> jis character se replace karna hai

replaceAll(“old word” ,”new word”) -> changing whole word to new word.

substring(beginingindex) -> this will divide the string from the index you passedand return it. Or substring(beginning index , end index. end wale index ko neglect

subsequence(beginning index , end index) -> retrun the char from star to end . end wale index ko neglect

isEmpty() ->The isEmpty() method checks whether a string is empty or not.

This method returns true if the string is empty ([length()](https://www.w3schools.com/java/ref_string_length.asp) is 0), and false if not.

Contains() -> returns true if specified word present in the string.

Join() -> The join() method joins one or more strings with a specified separator.

String fruits = String.join(" ", "Orange", "Apple", "Mango");

System.out.println(fruits);

format() : -The format() method in Java is used for formatting strings. It allows you to insert values into a string in a specified format.

Intern-> sends heaps object to constant pool

Concate()- > adding value : it will add the object in exisiting object and then create new memory in the constant pool and we decide what it will point to

//OR

Via + operator but note if we use

EG: s1 = “abc”

Syso(100+50+s1+100+50)

Before s1 it will perform

Eg : java eclipse

Java String compare:

we can compare string in Java on the basis of content and reference.

There are three ways to compare string in java:

* By using equals() method :
* check content
* the equals method here is override by objects class equals method
* By using == operator:
* check refernece
* By compareTo() method:
* Returntype type int
* Comparable interface ka method
* Method signature public int compareTo(Object obj)
* Checks the String if equal than “0”.
* If string length is bigger than positive number output(+).
* If string length is smaller than negative number output

(-).

* **String Buffer and String Builder:**

String buffer :

* introduce in 1.0
* Provides mutablity i.e String ka content change kar sakte hai hum.
* It is thread safe.
* We cant make literals here because it is mutable.
* It uses the actual equal method of object hence they check reference.

String builder :

* introduce in 1.5
* Provides mutablity i.e String ka content change kar sakte hai hum.
* It is not thread safe.
* We cant make literals here because it is mutable.
* It uses the actual equal method of object hence they check refernce.

Both string buffer and string builder as append method

It is use to concate two strings.

Regex in Java

+ : one or more time.

“ \\ . “: the actual dot character used (naki match karo kisi other thing ke sath like eg : “1.” Here dot will match the and dot replace by 1) .

^ :negation (indicates sequence start)

$ : indicates sequence end

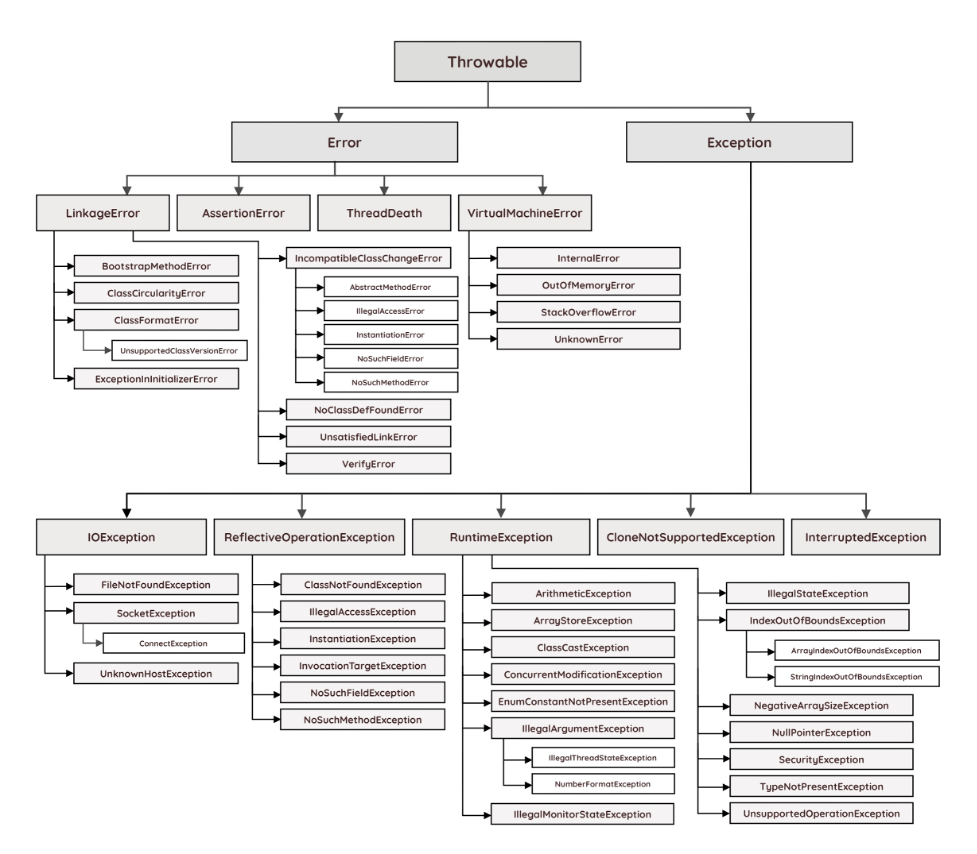
? : mathces any of the one character (hypen or space) in🡪 bank application.

* **Exception handling:**

What is an Exception

* An exception is an unwanted or unexpected event, which occurs during the execution of a program i.e at run time, that disrupts the normal flow of the program instructions.

Hierarchy of Java Exception Classes:



Types of exception:

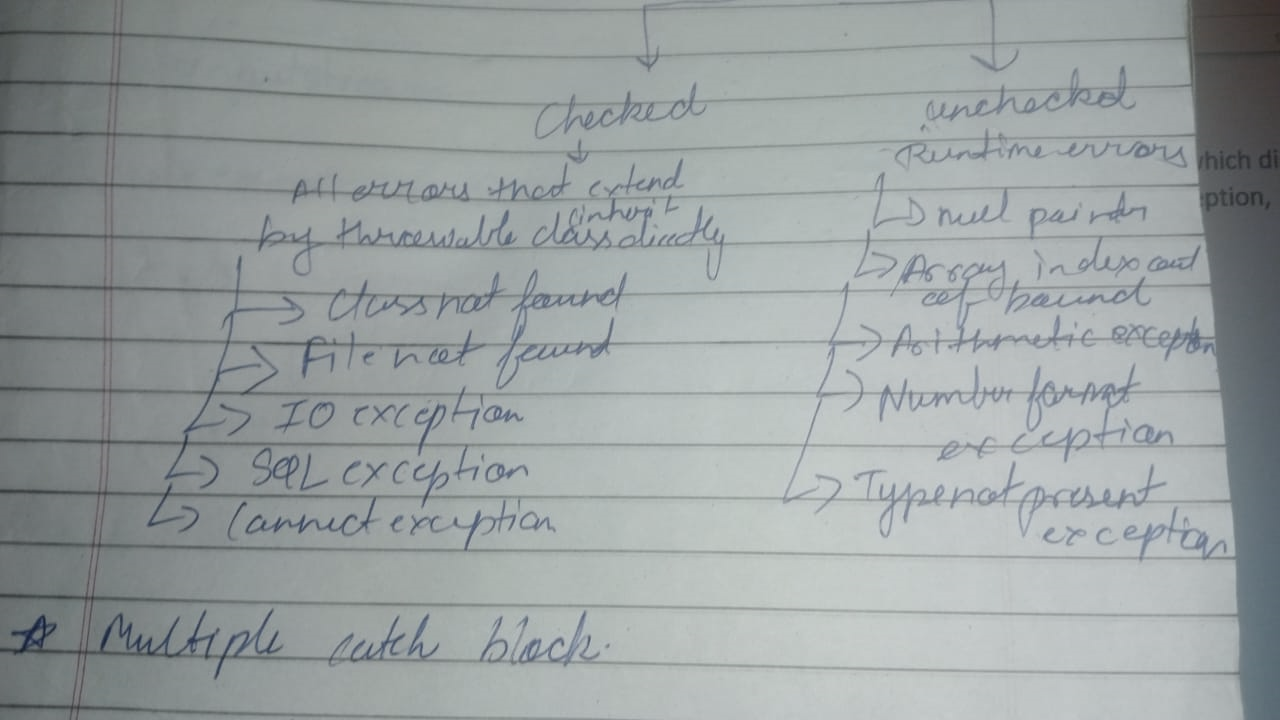
* Checked Exception (Compile time exception)
* Unchecked Exception (Runtime exception)

Checked Exception:

the classes that directly inherit the throwable class except Runtime Exception and error are known as checked exception. Checked exceptions are checked at compile-time.

Unchecked Exception:

The classes that inherit the runtime exception are known as unchecked exception. Unchecked exception is not checked at compile time they are checked at run time.



**What is Exception handling?**

Exception handling is the mechanism to handle errors which distrub the normal flow of the class such as classnotfoundexception, IOException, SQLException etc.

Ways to handle Exception in java:j

* **Try**
* **Catch**
* **finally**
* **throw**
* **throws**
* Try Catch Block:

try{

This is the try block

In this block , keep those statement which may throw run time exception

}

Catch{

This is the catch block ,

It takes one argument of type java.lang.Exception

This block catches the exception thrown by try block

}

* **Nested Try catch block in Java:(Inner Try catch block)**

In java try catch blocks can be nested i.e one try block can contain another try-catch block. The requirement of nested try-catch block arises when an exception occurs in the inner try-catch block is not handled by the inner catch blocks then the outer try-catch blocks are checked for that exception.

try{ //outer try block

//some statement here

Try{ //inner try block

}catch(){//inner catch block

}

}catch(){//outer catch block

}

* Multiple catch blocks

In some cases, a single statement may throw more than one type of exception. In such cases, Java allows you to put more than one catch block. One catch block handles one type of exception. When an exception is thorwn by the try block, all the catch blocks are examined in the order they appear and once catch block which matches with exception thrown will be executed.

**Rule** : starts from subclass to parent

Try {

} catch(){

}catch(){  
}catch(){  
} //n no of catch.

* Try catch using pipe (|) operator:

From java 7 onwards, there is one way for handling multiple exceptions (multiple catch). Multiple exception thrown by the try block can be handled by **a single catch block using pipe (|) operator.**

**Rule starts form subclass to parent**

**NOTE:jedar se exception waha se direct uss type ka exception ke catch block mai jaiga or jis line se exception uske baad ka koi line execute nhi hoga.**

* Try catch finally:

try{

This is the try block

In this block , keep those statement which may throw run time exception

}

Catch{

This is the catch block ,

It takes one argument of type java.lang.Exception

This block catches the exception thrown by try block

}

finally {

will execute even exception occur or not mostly use for closing the statemnet like datatbase connection close

This is finally block

}

NOTE: we can also make combination of try and finally.

* Try with resources

Feature of 1.7 repalcement try with catch with finally.

Automactically closes the resource even we don’t close it

Only pass the resource in the try block. Only close those who extends autocloseable . can be used without catch block

Try (resource) {

}finallyssssssss(){

}

* Throw keyword

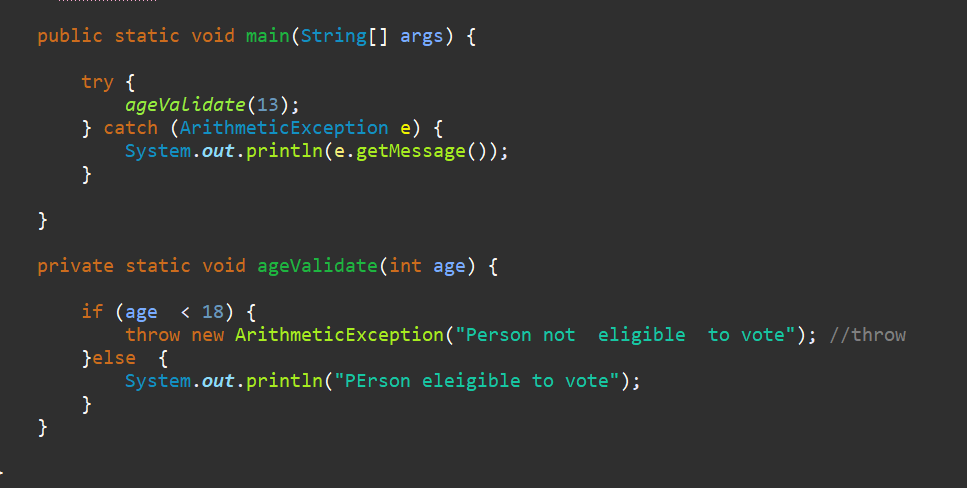
Java throw keyword is used to throw an exception explicitly.

* We can throw either checked or unchecked exception
* It’s mainly used to throw a custom exception

for example

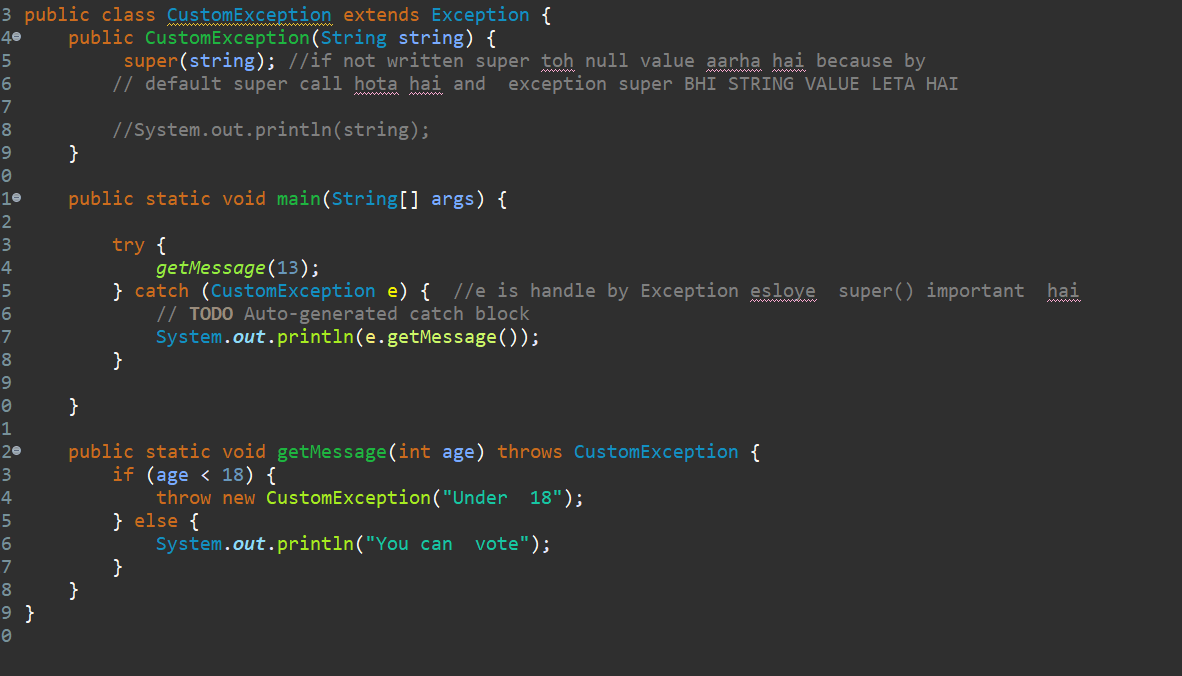
throw new exception class (“error message”);

* Propogate exception i.e exception handle calling method karega
* Always written inside methods
* Always inside conditions if -else condition
* Affects end user like direct browser mai error show hoga
* Does not handle exception
* Only single checke /unchecked -> exception throw kar sakte hai



Java Custom Exception:(User Defined Exception)

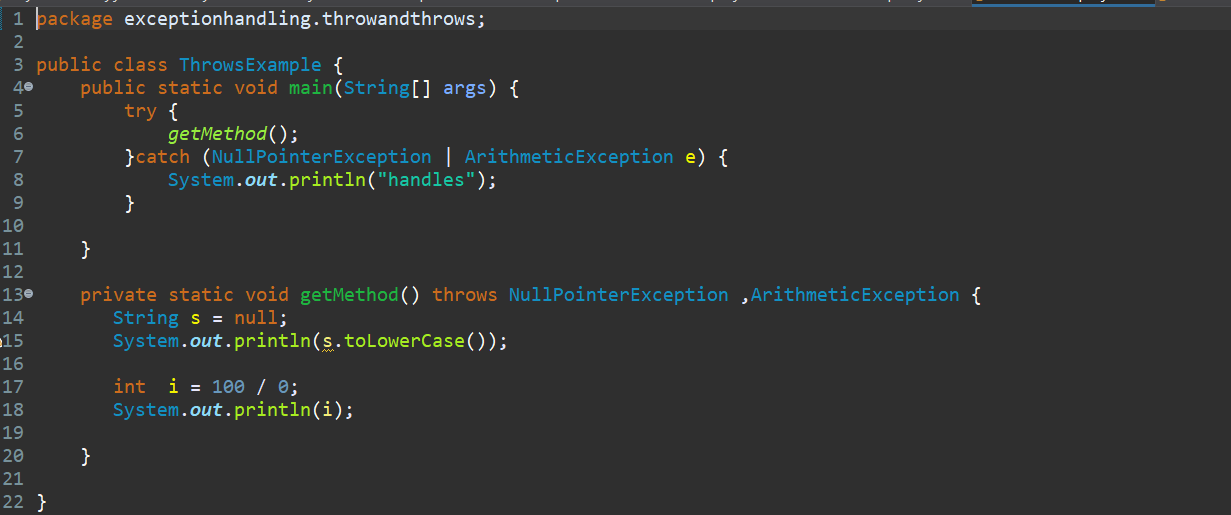
In java, we can have defined our own exception classes as per our requirements. These exceptions are called user defined exception in java or customized exceptions.

****

* Throws keyword

The java throws keyword is used to declare an exception. It gives an information to the programmer that there may occur an exception. So it is better for the programmer to provide the exception handling code so that the normal flow of the program can be maintained**.**

* **Method signature**
* **Do not handle exception**
* **Declare and exception**
* **Single,multiple exception throws**
* **Maximum checked exception**
* **Exception will ne handle by the callers**
* **Helps devloper to know that ki yeh exception aa sakta hai.**

****

RULES :

For inheritance -> child : same level class or same class ka subclass parent not allowed or same class

For constructor -> child :same level class or same class ka subclass or parent or same class

Printing Exception message ways:

* printStackTrace() method : It prints the name of the exception, description and complete stack trace including the line where exception occurred .
* getMessage() method: Mostly used, It prints the description of the exception.



* e.toString(): It prints name and description of the exception

use inside syso.



Q : Can we use try block without catch

Yes with the help of try with resources which do not require catch block.

Q: explain oops concept tusing with real life example ?

**1. Encapsulation (Hiding Details)**

👉 Think of a **car’s engine**. You don’t need to know how it works internally; you just use the **start button** to turn it on.

🔹 **In programming**, we keep important details **hidden** inside a class and provide methods to access them safely.

**2. Inheritance (Reusing Features)**

👉 A **sports car** and a **truck** both have wheels, engines, and seats because they **inherit** these features from a general **Car** class.

🔹 **In programming**, one class can inherit properties and behaviors from another to avoid writing the same code again.

+

**3. Polymorphism (Same Action, Different Forms)**

👉 A **car’s accelerator** works differently for a **normal car** and a **sports car**—one speeds up slowly, while the other speeds up quickly.

🔹 **In programming**, the same function can behave differently based on the situation.

**4. Abstraction (Hiding Complexity)**

👉When driving a car, you use the **steering wheel** to turn, but you don’t worry about how the tires move.

🔹 **In programming**, abstraction hides unnecessary details and shows only what’s needed.