OOP

Steps: 1) identify all the object
2) Generalized as a class of object

OOP is a programming approach which is alligned towards objects.

Date _____

Object oriented programming tries to map code instructions with real world making the code short and easier to understand.

Motivation for Object Oriented Approach: Code Reusability, Abstraction, Reduced maintainance, Quick & Efficient, Reliablity, Flexiblity, Real World Modeling.

What is OOP?

Solving a problem by creating objects is one of the most popular approaches in programming, this is called OOP.

Object: it combines both data & functions into a single unit

What is DRY? stands for Donot repeat yourself

↳ Focus on code reusolvility.

Class

A class is a blueprint for creating objects.

Applicat form ⇒ filled by a student ⇒ Application for the student

Class ⇒ Object Instantiation ⇒ Object

Contains info to create a valid object

Object: An object is an instantiation of a class. When a class is defined a template (info) is defined. Memory is allocated only after object instantiation.

How to model a problem in OOP

We identify the following

Noun → Class → Employee

Adjective → Attributes → name, age, salary.

Verbs → Methods. → getsalary(), increment()

1) Unstructured Programming
2) Structured/Procedural programming
3) Modular Programming
4) Object Oriented Programming
} Programming Approaches.

Date _____

The Four Pillars:

OOPs Terminology:

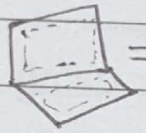
1) Abstractions → Hiding internal details (show only essential info!)

Eg: Android app development.

Smart Phone ← [ ] ⇒ Use this phone without bothering about how it was made.

2) Encapsulation: The act of putting various components together (in a capsule)

Eg: Car.



⇒ Laptop is a single entity with wifi + speaker + storage in a single box!

In Java, encapsulation simply means that the sensitive data can be hidden from the user.

3) Inheritance: The act of deriving new things from existing things.

Rickshaw ⇒ E-Rickshaw

Phone ⇒ SmartPhone

Implement DRY!

4) Polymorphism: One entity many forms.

Smartphone → Phone

Smartphone → Calculator.
↳ Camera
↳ Mp3

Writing a custom class

We can write a custom class as follows:

```
public class employee {
    int id;          → Attribute 1
    string name;     → Attribute 2
}
```

•) Aik Java file mae aik Public class hoti hai.

Any real world object = Properties + Behaviour.

Object in OOPs = Attributes + Methods.

Date

A class with methods:

We can add methods to our class employee as Follows:

```
public class employee{
    public int id;
    public string name;

    public int getsalary(){
        //code
        }
    public void getDetails(){
        //Code
        }
}
```

Programming Approaches: Unstructured Program: It consist of sequence of statement in the main program which modifies the program data state which maintain in global data.

Structured: Make use of structured control flow constructs of selection (if/then/else) and repetition (while and for) and subroutines. (Sequence, Selection, Iteration, Recursion)

Modular: Procedures are commonly functionality are grouped together into separate module.

Problem: ① Functions have unrestricted access to global data.

② In large application there will be many functions & global data items.

-) Difficult to conceptulize
-) Difficult to modify.

③ Unrelated Function, & data.

OOP: Align or position (something) relative to the points of a compass or other specified positions.

) So, OOP is a programming approach which is alligned towards object.

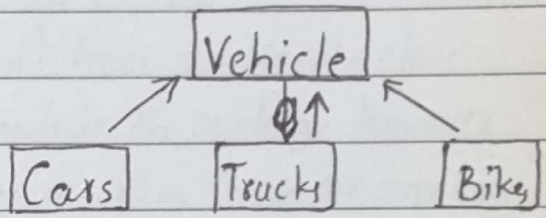
OOP created to manage complicity of procedural codebase.

Anatomy of Class

```
modifier class myclass {
    //class header
    //field constructor
    //method declarations
};
```

Date

An object is a:
- tangible or intangible thing.
- Living thing or non-living thing
- It can be classified (must belong to some class)
- Uniquely identified (a name)
- Characteristics.
- Functions (behaviour).



A system is a group or collection of objects interacting in some way with common themes.

What is a Model?

A model is an abstraction of some real-world system.

It is a representation of the objects within the system and the rules that govern the interaction of those objects.

Models can be purely physical or logical as represented in a computer program.

Object Oriented Data Modeling

- Uses the object as a metaphor for conceptualizing modeling & implementing a system.
- The goal is to design the application the way humans view system.
- Another goal is to develop systems with component that can be reused.

Object Oriented Development Process (OODP)

- Requirment Engineering
- Object Oriented analysis (OOA)
- Object Oriented design (OOD)
- Object Oriented Programming (OOP)
- Object-oriented Testing
- Deployment
- Support

What we have in a class?

A name: A class must have a name, unique name

Attributes: The characteristics or features of the object are known as attributes.

Methods {Behaviour, funtionality}

Methods are programs, algorithms or operations that are part of the class definition.

Date

Access Specifiers/Modifiers:

where a property/method is accesible.

Access specifier define how the members (attributes & methods) of a class can be accessed. It provides three levels of visiblity. This is how we implement information hiding.

- Default: members are accessible only within the same package.
- Public: members are accessible from outside the package.
- Private: members can't be accessed (or viewed) from outside the class
- Protected: members cannot be accessed from outside the package, however they canbe accessed in inherited classes or within the same package.

Class Methods

Methods are functions that belong to the class.

there are 4 types of access specifier.

public

protected

private

default

Date _____

Getters and Setters

Getter → Return the value [accessors]

Setter → Sets/Updates the value [mutator]

Example:

```
public class Employee{
    private int id;
    private string name;

    public string getName(){
        return name;
    }
    public void setName(){
        this.name = "XYZ";
    }
    public void setName(String n){
        this.name = n;
    }
}
```

Date

Constructor:

A constructor in Java is a member function of a class which initilize objects and executed automatically whenever object is created.

-) Sometimes however it's convenient if an object can intialize itself when it's first created, without requiring a separate call to a member function.
-) Automatic initialization is carried out using a special ~~number~~ member function called a constructor - At the time of calling the constructor, memory for the object is allocated in the memory.

Note: Even if we donot define any constructor explicitly the compiler will automatically provide a default constructor implicity.

How constructor are different from a normal member functions?

-) Constructor has same name as the class itself.
-) Constructor don't have return type.
-) Constructor is automatically called when an object is created.
-) If we do not specify a constructor, compiler generates a default constructor for us (expects no parameters and has an empty body)
-) Constructor are called only at the time of Object creation while method(s) can be called any number of times.

-) Primarily there are two types of constructor in Java:
  - -) No argument constructor.
  - -) Parameterized constructor.

Constructor Overloading: Compiler can overload constructor for creating objects in different ways - Compiler differentiate constructor on the basis of number of parameters, types of the parameter and order of parameters.

Date

Garbage Collector: In place of destructor, Java provides garbage collector that works the same as the destructor.

- The garbage collector is a program that runs on the JVM.
- It automatically detects the unused objects (objects that are no longer used) and free-up the memory.
- The programmer has no need to manage memory, manually. It can be error-prone, vulnerable, and may lead to a memory leak.

When destructor is called?

It is a special method that automatically gets call when an object is no longer used.

- When an object complete its life-cycle the garbage collector deletes that object and deallocates or release the memory occupied by the object.

Advantage:

It release the resource occupied by the object.

No explicit call is required, it is automatically invoked at the end of the program execution.

It does not accept any parameter and cannot be overloaded.

- It is difficult for the programmer to forcefully executed the garbage collector to destory the object.

But Java provides alternative way to do the same. The Java object class provides the finalize() method that works the same as destructor.

```
protected void finalize()
 {
  // resource to be closed
 }
```

Date

Static Data Member:

Static attribute are used when a single value for a data member applies to all objects of the class - These variables are automatically initialized to 0.

Syntax:

```
static int numberofObjects;
```

- A function can also be static - Since a static function is not "owned" by any single object of the class, it can be called even before any objects are declared.

Static Block:

- Is used to initialize the static data member.
- It is executed before the main method at the time of classloading.

```
class demo{
static{System.out.println("Static block is invoked");}
public static void main(String[] args){
 System.out.println("Hello main"); }}
```

Structure & Classes:

- structure as a way to group data.
- Classes as a way to group data & functions.
- The only formal difference between class and structure is that in a class the members can have access modifiers, while in structure they are public by default.

Date

1 Static Class Data:

•) If a data item in a class is declared as static, only one such item is created for the entire class, no matter how many objects there are.

•) A static data item is useful when all objects of the same class must share a common item of information.

•) It continues to exist even if there are no objects of the class.

•) a normal static variable is used to retain information between calls to a function, static class member data is used information among the objects of the class.

An instance method
- → invoke an instance method
- → access an instance data field
- → invoke a static method.
- → assess a static data field.

•) A static method can access static methods and variables:
A static method can call only other static methods; it cannot call a non-static method - A static method can be called directly from the class, without having to create an instance of the class.

Inheritance: OOP allows you to define new classes from existing classes.

Parent or base class

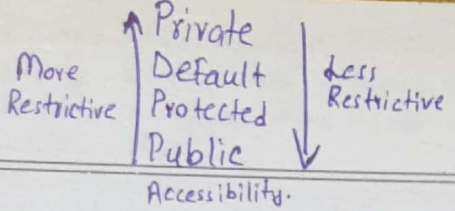
Inheritance enables you to define a general class (i.e superclass) and later extends it to more specialized classes (i.e subclasses).

Child or extended class

•) extends keyword is used.

•) Inheritance is an is-a relationship Example: Car is a Vehicle, Dog is an Animal

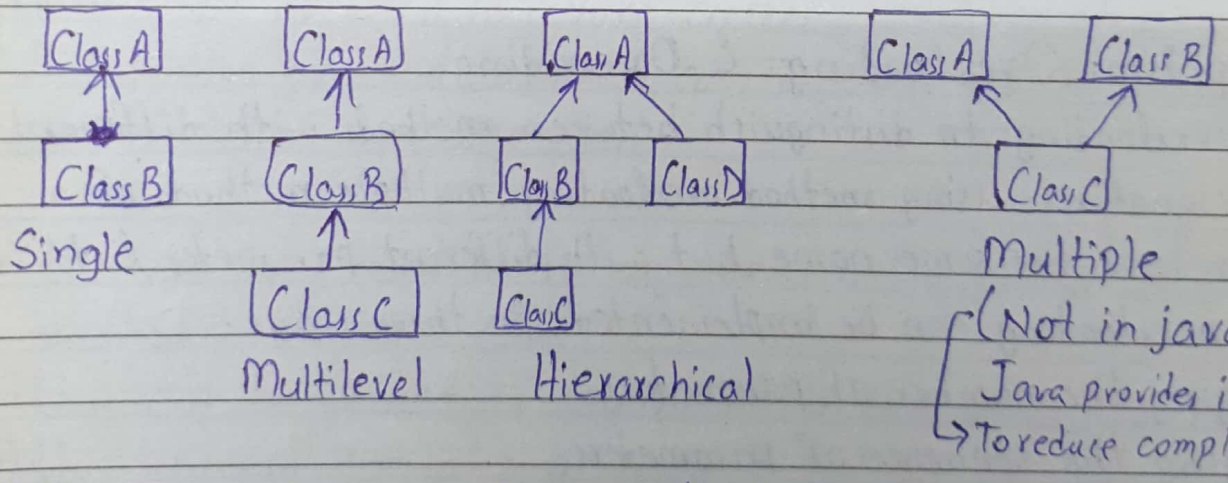
•) Super keyword is used to call method of parent class.



Date

- A subclass inherits all the members such as fields, nested classes and methods from its super class except those with private access specifier.
- However, constructor of class are not considered as a member of class and are not inherited by subclass.
- The child class can however invoke the constructor of the superclass from its own constructor.
- Members having default accessibility in the super class are not inherited by subclasses of other packages.
- These members can only access be accessed by subclasses within the same package as the superclass.

Types of inheritance:



- One can write new instance method with the same signature in the subclass as the one in the super class - This is called method overriding.
- A sub class constructor can be used to invoke the constructor of the superclass, either implicity or by using the keyword super.

Rules for overriding: The overriding method must have the same name, type and number of arguments as well as the return type as the super class method - An overiding method cannot have a weaker access specifier than the specifier of the super class method.

Date _____

Encapsulation: It describes the idea of bundling data and methods that work on that data within one unit.
-) Encapsulation links data with the code that manipulates it.
-) Fundamental concept of OOP.
-) Class abstraction is the separation of class implementation from the use of a class - The detail of implementation are encapsulated and hidden from the user - This is known as encapsulation -
e.g: a class in java -
The concept of encapsulation is also often used to hide the internal representation, or state, of an object from the outside - This is called information hiding -
-) It keep data and methods safe from outside interference & misuse.
  •) Publicize •) hides.

Polymorphism: Overloading & Overriding.
Method Overloading: to distinguish between methods with different method signatures - Using method overloading, multiple methods of a class can have the same name but with different parameter lists.
Method overloading can be implemented in three ways.
-) Changing the number of parameters.
-) Changing the sequence of parameters.
-) Changing the type of parameters.

•) The return type alone is insufficient to distinguish two version of methods.
•) Method overloading increases the readability of the program.

Byte > Short > Int > Long > Float > Double → Widening (implicit) Casting → Done automatically

Double -> Float -> long -> int -> char -> short -> byte → Narrowing Casting (explicit) Converting larger data type to smaller data size.
↳ Done manually.

Explicit casting must be done manually by placing the type in parenthesis in front of value.

```
double myDouble = 9.78d;
int myInt = (int) myDouble;
```

Date

An automatic type conversion take place in Java if these two condition are met:
- The data types are compatible with eachother.
- The destination type is bigger than the source type

In implicit type casting is done internally by java whereas explicit type casting is done by the programmer, doesnot perform automatically. In explicit casting cast operator is needed whereas no need of any operator needs in the case of implicit type casting.

The this keyword:

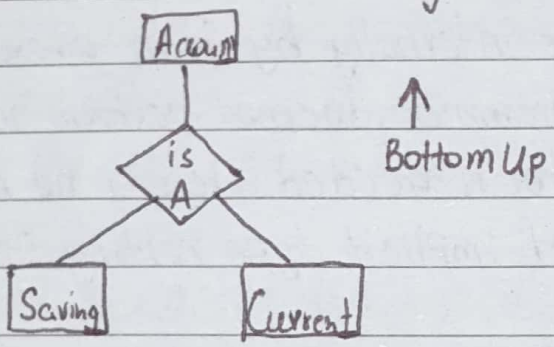
this is alway a reference to the object on which the method was invoked.
- this lets program refers directly to the current object.

this() can be used to invoke current class constructor.
- It can also be used inside a constructor to invoke another constructor of the same class

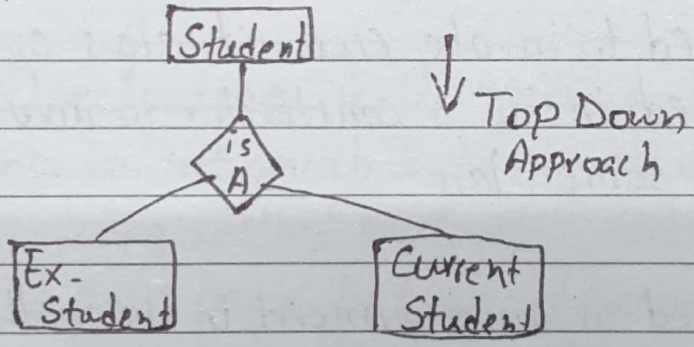
- this can be passed as an argument in the method call for method in the same class.
- this can be passed as argument in the constructor call.
- this keyword can also be used to return the current class instance.

Date _____

Generalization/Inheritance: is a bottom up approach in which two sub classes combine to form a superclass. It is a mechanism for combining similar classes of objects into a single, more general class
Example: a BankAccount - Saving Account and Credit Card Account



Specialization: is opposite to generalization, it means creating new sub classes from an existing class - It is a top down approach in which one super class can be broken down into two sub classes.



Relationships:
Association: is a relationship between two objects in other words association defines the multiplicity between the object - Association establishes relationship between two separate classes through their objects - You may be aware of one-to-one, one to many, many-to-one many-to-many all these words define an association between objects. It refers to how objects are related to eachother and how they are using each other's functionality.
Symbol Representation: ⟶
Example: A student & a faculty are having an association.

- There are 2 types of association.
  1) Aggregation
  2) Composition.

Date

Aggregation (Has A) is a special form of association - It's a directional association, which means it is strictly a one way association - When an object 'has-a' another object, then you have got an aggregation between them - Direction between them specified which object contains the other object - An association is said to be aggregation if both objects can exist independently - For Example: a Team object and a Player object - The team contain multiple players but a player can exist without a team. Symbol : ——◇

Composition: is an association represents a part of a whole relationship where a part cannot exist without whole - If a whole is deleted then all parts are deleted - It has a stronger relationship - Objects of Class2 live and die with Class1 Class2 cannot stand by itself. (HAS-A) relationship.
Symbol : ——◆

Aggregation	Composition
Aggregation indicates a relationship where the child can exist separately from their parent class. Example: Automobile (Parent) and Car (Child). So, if you delete the Automobile the child class still exist.	Composition display relationship where the child will never exist independent of the parent. Example: House (Parent) and Room (child) Rooms will never separate into a House.

cannot be used as data type
- Abstract methods are non-static.
- An abstract class can't be instantiated using the new operator, but you can still defines its constructor which are invoked in the constructor of its subclass.
- A subclass can be abstract even if its superclass is concrete etc.

Date _____

Abstraction: is a process of hiding the implementation details and showing only functionality to the user. Abstraction lets you focus on what the object does instead of how it does it.
Two ways to achieve abstraction: - Abstract class - Interface.

A class which is declared as abstract is known as an abstract class. It can have abstract and non-abstract methods. It needs to be extended and its method implemented. It cannot be instantiated.
- Any class that contain one or more abstract method must also be declared abstract. 'abstract' keyword is used to declare class.

Properties:
There can be no objects of an abstract class. That is, an abstract class cannot be directly instantiated with the new operator. Objects would be useless, because an abstract class is not fully defined. Abtract constructors, or abstract static methods cannot be declared. Abstract classes can include non-abstract method also. It can have final methods. It can have constructors & static methods also

Abstract Method: A method which is declared abstract and does not have implementation is known as an abstract method. Superclass only defines a generalized form that will be shared by all of its subclass, leaving it to each subclass to fill in the details. In order to ensure that a subclass does, indeed, overrides all necessary method, Java provides abstract methods. Certain methods can be made mandatory to be overridden by subclasses by specifying the abstract type modifier. Declare: abstract type name(parameter list);
- Abstract methods are sometimes referred to as subclasses responsibility because they have no implementation specified in the superclass.

o) It is illegal to declare a class as both abstract and final since abstract is incomplete itself.

Date

Final - with inheritance:

The keyword final has three uses.

1.) It can be used to create the equivalent of a named constant.
2) Using final to Prevent Overriding
3) Using final to Prevent inheritance.

Final Method (Using final to Prevent Overriding)

o) To disallow a method from being overridden, specify final as a modifier at the start of its declaration - Methods declared as final can sometimes provide a performance enhancement. Normally, java resolves calls to methods dynamically, at run time - This late binding.
However, since final methods cannot be overridden a call to one can be resolved at compile time - This is called early binding.

-) Early Binding: The binding which can be resolved at compile time by the compiler is known as static or early binding - Binding of all the static, private and final methods it is done at compile-time.

-) Late Binding: In the late binding or dynamic binding, the compiler doesn't decide the method to be called - Overriding is a perfect example of dynamic binding - In overriding both parents and child classes have the same method.

Early	Late
the method definition and method calling are linked during the compile time.	The method defination and method call are linked during the run time.
Actual object is not used for binding.	Actual object is used for binding.
Example: Method overloading	For example: Method overriding
Program execution is faster.	Program execution is slower.

Date

Interface:

An interface is used to specify what a class must do, but not how it does it - Interfaces are syntactically similar to classes, but •) they lack instance variables, •) and their methods are declared without any body. Once an interface is defined, any number of classes can implement it - To implement an interface, a class must create the complete set of methods defined by the interface - However each class is free to determine the details of its own implementation. Interface fully abstract 'class' structure from its implementation Interface cannot create its object.

•) Variables can be declared inside of interface declaration - They are implicitly final and static, meaning they cannot be changed by the implementing class - They must also be intialized.

All methods and variables are implicitly public.

Once an interface has been defined, one or more classes can implement that interface - To implement an interface, include the implements clause in a class definition and then create the methods defined by the interface - The general form of a class that includes the implements clause looks - If a class implements more than one interface the interfaces are separated with a comma - The method that implement an interface must be declared public. The type of signature of the implementing method must match exactly the type signature specified in the interface definination. POLYMORPHIC : The version of method that is called is determined by the type of object that reference variable refers to at runtime - Interface fully utilize the "one interface, multiple methods" aspect of polymorphism. Partial Implementation: If a class includes an interface but doesnot fully implement the methods defined by that interface, then that class must be declared as abstract.

Date

Nested Interface: An interface can be declared a member of a class or another interface. Such an interface is called a member interface or nested interface. A nested interface can be declared as public, private or protected. This differs from a top-level interface, which must either be declared as public or default. When a nested interface is used outside of its enclosing scope, it must be qualified by the name of the class or interface of which it is a member.

Variable in interface: Interface can be used to import shared constants into multiple classes by simply declaring an interface that contains variable that are initialized to desired values. When a class "implements" the interface, all of these variable names will be in scope as constants.

If an interface contains no methods, then any class that includes such an interface doesn't actually implementing anything. It is as if that class were importing the constant fields into the class name space as final variables.

- One interface can inherit another by using of the keyword extends. Syntax is same as inheriting classes.
- When a class implements an interface that inherits another interface it must provides implementation for all methods defined within the interface inheritance chain. An interface cannot implement another interface but it can inherit.

Realization Relationship:

- Interface shows realization relationship.
- It is rendered by a dashed line with a hollow triangle towards the specifier.

Generalization: an inheritance relationship.

- Inheritance (between classes)
- Realization (interface implementation)

Date

Abstract Class	Interfaces
A class can extend only one abstract class	A class can implement any number of interface.
In abstract class keyword 'abstract' is mandatory to declare a method as an abstract.	In an interface keyword 'abstract' is optional to declare a method as an abstract.
Abstract class can have protected & public abstract methods	Interface can have only public abstract methods.
No restrictions on variable	It can have only constant variables.
Methods that are not declared abstract can have its implementation	No methods can have implementation.
Constructor are invoked by subclass through constructor chaining	~~No constructors.~~ No constructors.
A strong is-a relationship can be modeled using abstract classes.	A weak is-a relationship can be modeled using interfaces.

Practise Question done on last page:

Exception Handling: Date

The exception handling in java is one of the powerful mechanism to handle the runtime errors so that the normal flow of the application can be maintained.
↳ such as: ClassNotFoundException, IOException, SQLException, RemoteException etc.

Advantage:

The core advantage of exception handling is to maintain the normal flow of the application. A exception normally disrupts the normal flow of the application; that is why we need to handle exceptions.

Exception describes errors caused by your program and external circumstances. These errors can be caught & handled by your program.
RuntimeException is caused by programming errors, such as bad casting, accessing an out-of-bounds array, and numeric errors.

There are mainly two types of exceptions: checked and unchecked. An error is considered as the unchecked exception. However, according to Oracle, there are three type of exceptions namely. 1) Checked 2) Unchecked 3) Error

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

Unchecked: The classes that inherits the RuntimeException are known as unchecked exception. For example ArithmeticException, NullPointerException etc. Unchecked exception are checked at runtime.

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

Date

Exception keywords:

Java provides five keywords that are used to handle exception.

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 to 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 used to throw the exception.

Throws: The "throws" keyword is used to declare exceptions. It specifies that there may occurs an exception in the method. It doesn't throw an exception. It is always used with method signature

Common Scenerios of Java Exceptions:

ArithmeticException: If we divide numbe by zero

NullPointerException: If we have null value in any variable

```
String s = null
Sout(s.length()); //NullPointerExc
```

NumberformatException: If the formating of any variable/number is mismatched. Suppose we have String variable convert into digit

```
String s = "abc"
int i = integer.parseInt(s); //NumberformatException.
```

ArrayIndexOutOfBoundsException: When array exceeds to it's size

```
int a[] = new int[5]
a[10] = 50;
```

Catch Multiple Exceptions:

For a try block we may give many catch blocks so whatever block exception related it will catch for example n = 30/0 we have 3 catch block 1) ArithmeticException 2) Array out of bound 3) Exception so it will catch ArithmeticException

Nested try block

In this in a try block we can make multiple try block with catch block.

* Continue.

Date

When any try block does not have a catch block for a particular exception, then the catch block of the outer (parent) try block are checked for that exception and if it matches the catch block of outer try block is executed - If none of the catch block block specified in the code is unable to handle the exception, then the java runtime system will handle the exception - Then it display the system generated message for that exception.

Finally block: is a block used to execute important code such as closing the connection, etc.

Java finally block is alway executed whether an exception is handled or not - Therefore, it contains all the necessary statements that need to be printed regardless of the exception occurs or not.

Common Scenarios of Java Exception:

Case 1: When an exception does not occur.

Case 2: When an exception occur but not handled by the catch block.

Case 3: When an exception occurs and is handled by the catch block.

throw Exception: In Java, exceptions allows us to write good quality codes where the errors are checked at the compile time instead of runtime and we can create custom exceptions making the code recovery & debugging easier.

Syntax: throw new exception_class ("error message")

Example throw new IOException ("Sorry device error")

```
Class new{
Validate( ){
if (age<18)
  throw new Arithmetic Exception ("Person is not eligible to vote") }
else{
  Sout ("Pers eligible to vote");
```

Date

throws keyword: The Java throws keyword is used to declare an exception. It gives an information to the programmer that they may occur 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.

Exception Handling is mainly used to handle the checked exceptions. If there occurs any unchecked exception such as NullPointerException, it is programmer's fault that he is not checking the code before it being used.

Java throws keyword is used in the method signature to declare an exception which might be thrown by the function while the execution of the code.

Syntax: return_type method_name() throws exception_class_name()

Java Custom Exception: In Java, we can create our own exceptions that are derived classes of Exception class. Creating our own Exception is known as custom exception or user-defined exception. Basically, Java custome exception are used to customize the exception according to user need.