

Day 1 : 21/10/2023

Backend technology -> Java and connecting database using JDBC.

## Java

Core Java

J2SE

J2EE

J2ME

JavaSE

JavaEE

JavaME

JSE

JEE

JME

Java Standard Edition

Java Enterprise Edition

Java Micro Edition

Desktop application

Servlet and JSP : web application

Console base application

We will teach how to create

Basic Java Programming

Rest API

OOPs Concept

those rest api we will

Object,

call in angular

Class,

maven build tools

Exception handling

Collection Framework with Data structure

JDBC topic to connect MySQL

Java8(Lambda Expression and Stream API)

and 11 and 15

Regex

Java 22

Intro to Mongo DB

No SQL Database

Using Java we can connect to database using

1. JDBC
2. ORM -> Hibernate and JPA.
3. Using Core Java, Servlet /JSP we can use JDBC or Hibernate or JPA.
4. Spring Framework we can use JDBC or Hibernate or JPA.
5. In Spring boot JDBC or JPA or Spring Data.

Phase 3 or Course 3

Spring framework and Spring boot

Junit 5 testing tool

Course 4 : Testing and deployment

Testing

TestNG, Selenium,

Docker, CI and CD using Jenkin, Overview of AWS

In AWS we will teach you how to deploy application in EC2 instance using Docker with CI and CD tool ie Jenkin

Frontend -> angular

Backend -> spring boot

Database -> MySQL

Day 2 : 22/10/2023

Java : Java is pure object oriented and platform independent programming language.

Java develop in nov 1995. The initial name of java is Oak.

Version of Java

Java 1.0,1.1,1.2	7, 8,9,11,15	22
------------------	--------------	----

Java is an open source.

From java 11 onward java is not open source.

Java was belong to sun micro system

Now it is part of oracle.

Oops

object : object is any real world entity. Example

property or state or variable -> have -> variable, fields etc

name, age, weight, height, colour etc.

int, float, char, double, string etc.

Person

Behaviour or function or methods -> do/does -> function or methods.

Teaching(), sleeping(), talking(), typing(), etc

Bank

Animal

Car

Customer

class : blue print of object or template of object or user defined data type which help to describe the object.

Syntax of class

```
class ClassName {  
    fields or variable declaration  
    method or function declaration  
}  
  
class Demo {  
    main method : pre defined methods.  
}
```

Class name must follow pascal naming

1. If class contains one word first letter upper case.
2. If class contains more than one word each word first letter upper case.

Syntax to write method in java

```
returnType methodName(parameterList) {  
    Method body;  
}
```

```
void display() {
```

```
}
```

Method name is display no passing parameter and no return type.

```
void add(int x, int y) {
```

```
}
```

Method name is add. We need to pass two parameter of type int but no return type.

```
String sayHello(String name) {
```

```
    return "Welcome user "+name;
```

```
}
```

Method name is sayHello. Need to pass one parameter of type string and return string value

In java all method must part of class.

```
class Test {
```

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

```
        System.out.println("Welcome to Java");
```

```
    }
```

```
}
```

Method name and variable name must be follow camel naming rules.

1. If variable name or method name one word. Then it must be in lower case. Like main(), sleeping(), display(), read() etc and id, name, salary etc.
2. If variable name or method name more than one word then from second word first letter upper case like displayInfo(), calSalary(), empId, etc.

**Data types** : Data type is a type of data which tells what data type data it can hold.

a=10;                      python

var b=20;                      js

let c:number =30;              ts

datatype variableName;

datatype variableName=value;

In java data types are divided into two types.

1. Primitive data types :it is use to store only value

8 types

a. byte              1 byte              -128 to 127

b. short              2 byte

c. int              4 byte

d. long              8 byte

without decimal point

e. float              4 byte

f. double              8 byte

with decimal point

g. char              2 byte      in single character

h. boolean              1 bit      true or false value

Operator :

1. arithmetic operator : +, -, \*, /, %
2. logical operator : &&, ||, !
3. conditional operator : >, <, >=, <=, ==, !=
4. assignment operator =
5. increment and decrement operator : ++, --

if statement

- a. simple if
- b. if else
- c. if else if
- d. switch statement

syntax of switch statement

```
int label =1;
```

```
switch(variableName) { variable type must be int, char or string in java.
```

```
case 1:block1
```

```
    break;
```

```
case 2:block12
```

```
    break;
```

```
case 3:block3
```

```
    break;
```

```
case 4:block4
```

```
    break;
```

```
default : default block ;
```

```
    break;
```

```
}
```

**Looping** : looping is use to execute the task or statement again and again repeatedly

Till the condition become false.

Initialization

Condition

Task

Increment and decrement.

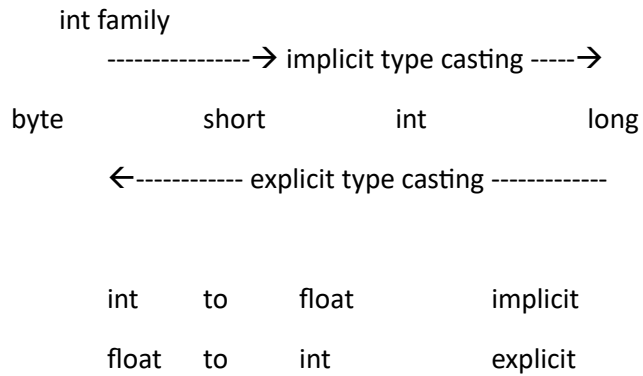
- a. While loop : entry loop
- b. Do while loop : exit loop
- c. For loop

## Type casting :

Converting from one data type to another data type is known as type casting.

Two type of type casting.

- a. Implicit type casting
- b. Explicit type casting



in java by default every decimal number double consider.

2. Non primitive or reference data types. : it is use to store value as well as reference of another data types.

4 types of non primitive

1. array :array is user defined or reference data type which help to store same type of values.

In java array is known as fixed in memory size.

datatype arrayname[];          array declaration

**int num[];**

int a;

**int num1[]={10,20,30,40,50};**          array declaration with initialization

array memory creation syntax

datatype arrayname[]=new datatype[size];

**int num2[]=new int[10];**

num2 can hold 10 value. Start from 0 to 9 position.

Taking the value through keyboard in java

Using Scanner class.

Scanner is a pre-defined class part of util package.

Package is a collection of classes as well as interfaces.

We need to create Scanner class object.

Syntax to create Scanner class object

```
Scanner obj = new Scanner(System.in);
```

While creating scanner class object we will get error. Because Scanner class part of util Package. We need to import it.

## 2. class

Scanner : pre defined class which help to scan the value through keyboard.

String : in Java String is a pre defined class. which help to store more than one Character in double quote.

String is pre defined class part of lang package.

By default java imported lang package.

## 3. interface

## 4. enum

in Ts or JS

```
class Customer {  
}
```

```
var obj1 = new Customer();
```

```
let obj2=new Customer();
```

```
Scanner obj1 = new Scanner(System.in);
```

```
obj.nextInt(); int value
```



obj.next()    string value

obj.nextFloat()    float value

Day 3 : 28/10/2023

### Java OOPs concept

object : any real world entity.

Property or state -> have -> variable or fields.

Person

Behaviour -> do/does -> functions or methods

Bank

Car

Animal

Customer

Employee

class : blue print of object or template of object or user defined data type which help to describe the object.

### Method or function syntax

```
return type methodName(parameterList) {
```

```
}
```

```
void info() {
```

```
    coding....
```

```
}
```

Method name is info. For this method we are not passing any parameter while calling

And this method doesn't return any value to caller method.

```
void add(int x, int y) {
```

```
}
```

Method name is add while calling this method we need to pass 2 parameter of type int and not return type.

```
String sayHello(String name) {
```

```
    Coding
```

```
    return "Welcome to my method user "+name;
```

```
}
```

Method name is sayHello. We need to pass one parameter of type string and it return string value to caller method.

Fields or variable.

In Java variable or fields are divided into 3 types.

1. Instance variable
  - a. The variable which declared inside a class but outside method including main method also is known as instance variable.
  - b. Instance variable hold default value base upon their data types.  
Int family → 0  
Float family → 0.0  
Char → space or white space  
Booean → false  
String → null
  - c. Instance variable we can use all method directly but method must be part of same class and it must be non static.
2. Local variable
  - a. The variable which declared inside a method is known as local variable.
  - b. Local variable doesn't hold default value we have to initialize.
  - c. The scope of the variable within that method where it declared.
3. Static variable

**Constructor :** constructor is a type of special method which help to create the object.

Pts.

1. Constructor have same name as class itself.
2. Constructor no return type not even void also.
3. Constructor get call automatically whenever we create the object that class.

```

class Employee {
    Employee() {}
    constructor(){}
}

```

in Java Constructor

in typescript or angular



When local or parameter variable and instance variable have same name then local variable or parameter variable hide the visibility of instance variable.

If we want to refer to instance variable this we need to use this keyword. this is a keyword which refer to current to object.

### Parameter constructor

**Encapsulation** : binding or wrapping data or variable and code or method/function is single unit is known as encapsulation.

Example : class.

### JavaBean class

## Inheritance :

Inheritance is use to inherits or acquire the property and behaviour of old class to new class.

```
class OldClass {           super class or base class or parent class.
    property
    behaviour
}

class NewClass extends OldClass{       sub class or derived class or child class.
    property
    behaviour
}
```

### Types of inheritance

1. Single inheritance : one super class and one sub class  
class A {                      }  
class B extends A {           }
2. **Multilevel** inheritance : one super class and n number of sub class connected one by one.  
class A {                      }  
class B extends A {           }  
class C extends B {           }  
class D extends C {           }
3. Hierarchical inheritance : one super class and more than one sub class connected directly to super class.  
class A {                      }  
class B extends A {           }  
class C extends A {           }  
class D extends A {           }
4. Multiple inheritance : more than one super class and one sub class  
class A {                      }  
class B {                      }  
class C extends A,B {        } Java doesn't support multiple inheritance. Error  
this type of inheritance in java we can achieve using **interface**.

```

class Employee {
    id,name,salary
    readEmployee()
    displayEmployee();
}
class Manager extends Employee {
    numberOfEmp;
    readManager()
    displayManager();
}
class ProjectManager extends Manager{
    clientInfo
    readProjectManager()
    displayProjectManager();
}

class Developer extends Employee{
    techName;

    readDeveloper()

    disDeveloper()

}

```

#### Day 4 : 29/10/2023

Scanner is a pre defined class part of util package.

Outside a class we need to import java.util.Scanner;

```
Scanner sc = new Scanner(System.in);
```

```
System.out.println("Enter the id");
```

```
Int id = sc.nextInt();           to scan int value
```

```
String name = sc.next();         to scan string value but only one word.
```

```
String msg = sc.nextLine();      it take more than one word till hit enter key.
```

```
float salary = sc.nextFloat();
```

**Polymorphism** : one name many forms or many implementation.

2 types

1. Compile time or static binding or early binding

Example : Method overloading :

**In C++ Operator overloading is example of compile time polymorphism**

The method have same name but different parameter list ie type of parameter list or number of parameter list must be different.

2. Run time or dynamic binding or late binding

Example : Method overriding

The method have same name and same method signature (number of parameter list, type of parameter list and return type must be same).

To achieve method override we need inheritance.

Annotation : annotation is like a meta data . meta data is data about data.

Annotation is like a decorator in angular.

Java provided lot of pre defined annotation. All annotation start with pre fix @ followed by annotation name. few annotation we can use on class level or method level or property level.

@Override. This annotation we can use on method level of sub class. if sub class method

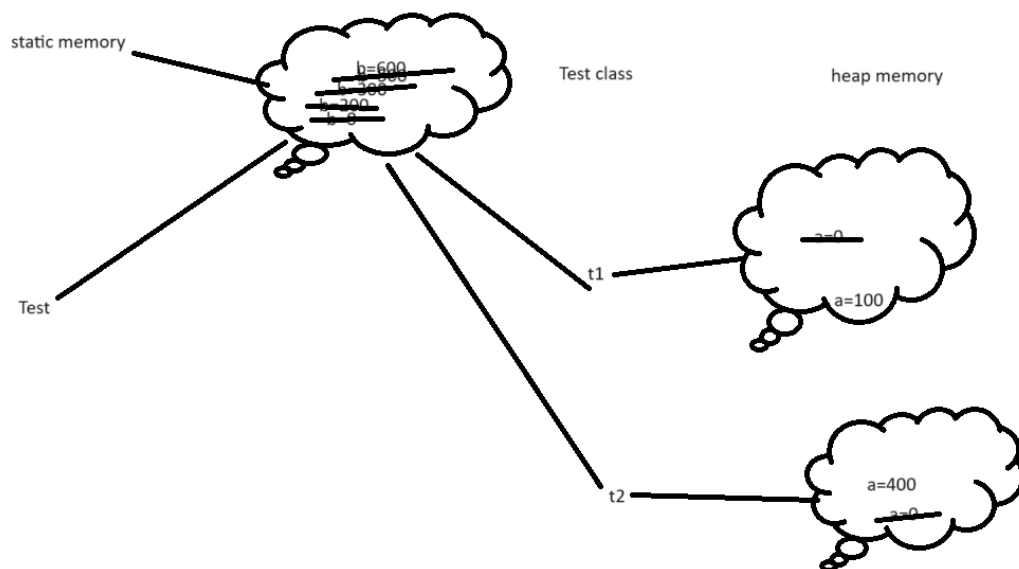
Overriding super class method then we doesn't get any error else we will get the error.

Non access specifiers keywords.

Java provide 3 non access specifiers keywords.

1. abstract :abstract is a keyword we can use with method and class but not with variable.
  - a. Abstract method : method without body or without curly braces or incomplete method is known as abstract method.  
abstract returnType methodName(parameterList);
  - b. If class contains one or more abstract method that class we need to declare as abstract class.  
abstract class classname{  
  
}  
}
  - c. Whichever class extends abstract class that class must be provide the body for all abstract method belong to that class mandatory.
  - d. Abstract class can contain normal as well as abstract method. Means it can contains zero or 1 or many abstract methods.
  - e. Abstract class we can't create object.

2. final : final keyword we can use with variable, method and class.
- a. final variable : to declare constant value in java we use final keyword.  
final int A=10;  
A=20; error
  - b. final method : if method is final we can't override that method. But we can use that method.
  - c. Final class : if class is final we can't inherit that class or extends that class.
3. static : static keyword we can use with variable and method but not with class.
- a. static variable : if variable is static we can access or assign the value of variable using class name.
  - b. static method : if method is static we can call that method with help of class name.
  - c. we can access static variable using object also as well as we can call static method with help of object also.
  - d. Inside non static method we can access both the type of variable directly but inside static method we can access only static variable directly.
  - e.



**interface** : interface is a type of reference data type. It is also known as

100% pure abstract class.

Syntax for interface

```
interface interfaceName {  
    variables;  
    methods;  
}
```

**By default** all variable inside a interface are public ,static and final.

By default all methods in interface are public and abstract.

```
interface Abc {  
    int A=10;  
    void dis1();  
}  
  
interface Xyz {  
    int B=20;  
    void dis2();  
}  
  
interface Mno extends Abc,Xyz{  
    int C=30;  
    void dis3();  
}  
  
class Test implements Abc,Xyz {  
    dis1() and dis2() methods.  
}
```

Like a class one interface can extends another interface as well as interface can extends more than one interface but class can't.

Class always implements interface. Class can implements more than one interface.

Which ever class implements any interface that class must be provide the body for all those methods belong that interface.



## Interface Vs Abstract class.

1. Abstract class can contains normal as well as final variable but interface contains only final variable.
2. Abstract class can contains normal as well as abstract method but interface contains only abstract method.
3. Normal class can extends only one abstract class but normal class implements more than one interface.
4. Abstract class can contains default constructor as well as we can write empty or parameter constructor but interface doesn't contains any constructor.
5. Using abstract class we can achieve partial abstract but using interface we can achieve 100% abstraction.

Common point we can't create object of interface as well as abstract class.

Class extends class only one

Interface extends interface more than one

Class implements interface more than one

Interface can't extends or implements to class.

```
abstract class Bank {  
    abstract void withdraw();  
    abstract void deposit();  
    void rateOfInterest() {  
    }  
}
```

Or

```
interface Bank {  
    void withdraw();  
    void deposit();  
    void rateOfInterest();  
}
```

## Package and access specifiers

Package is a collection of classes and interface which have same name but different purpose.

In java package are divided into 2 types.

1. User defined package.
2. Pre defined package.

To create the package we need to use

```
package packagename;
```

school

Attendance

School student

Package is like a directory or folder

college

Attendance

College student

## Access specifiers

Using access specifiers we can provide the visibility of variable, method and class part of same package or other package.

Java provided 4 types of access specifiers

1. private: we can use this access specifier with what  
we can use private with instance variable, static variable, non static method, static method, constructor but not with local variable as well as class.  
scope : within a same class.
2. default or nothing : we can use this access specifier with what  
we can use with all  
scope : within a same package
3. protected : we can use this access specifier with what  
we can use private with instance variable, static variable, non static method, static method, constructor but not with local variable as well as class.  
scope : within a same class as well as other package if it is sub class
4. public : we can use this access specifier with what  
we can use private with instance variable, static variable, non static method, static method, constructor, class but not with local variable.  
scope : same package well as other package.

Day 5 : 04/11/2023

Java bean class

1. class must be public
2. all variable must be private
3. for each variable you need to provide setter and getter method.
4. setter set the value and getter get the value.
5. Setter you can write with conditions if you need .
6. If you want provide empty as well as parameter constructor rather than setting value through setter methods.

```
public class Customer {  
  
    private int cid;  
  
    private String cname;  
  
    public void setId(int cid) {  
        this.cid = cid;  
    }  
  
    public int getCid() {  
        return this.cid;  
    }  
}
```

Pre defined packages.

```
java                                javax    root package
```

lang

sql

io

swing

util

servlet

sql

ejb

net

jms

awt

net

etc

etc

by default every java program imported lang package.

So we can use all the classes and interface part of lang package without imported

### Explicitly

String

System

By default every java program it may be pre defined or user defined internally extends

Object class.

Super API (application programming interface).

## Exception handling

Exception is an object or memory which generate when unexpected or abnormal thing happened during the execution of a program.

Using some technique we need to handle generate exception that is known as exception handling.

## Java program

Compile program

`javac Demo.java`

Run the program

`java Demo`

compile time error

run time error

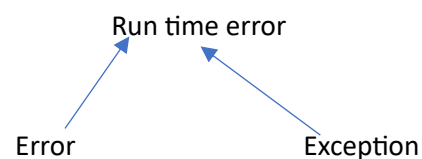
syntax error

or typo error

after compile program

successfully it generate .class file.

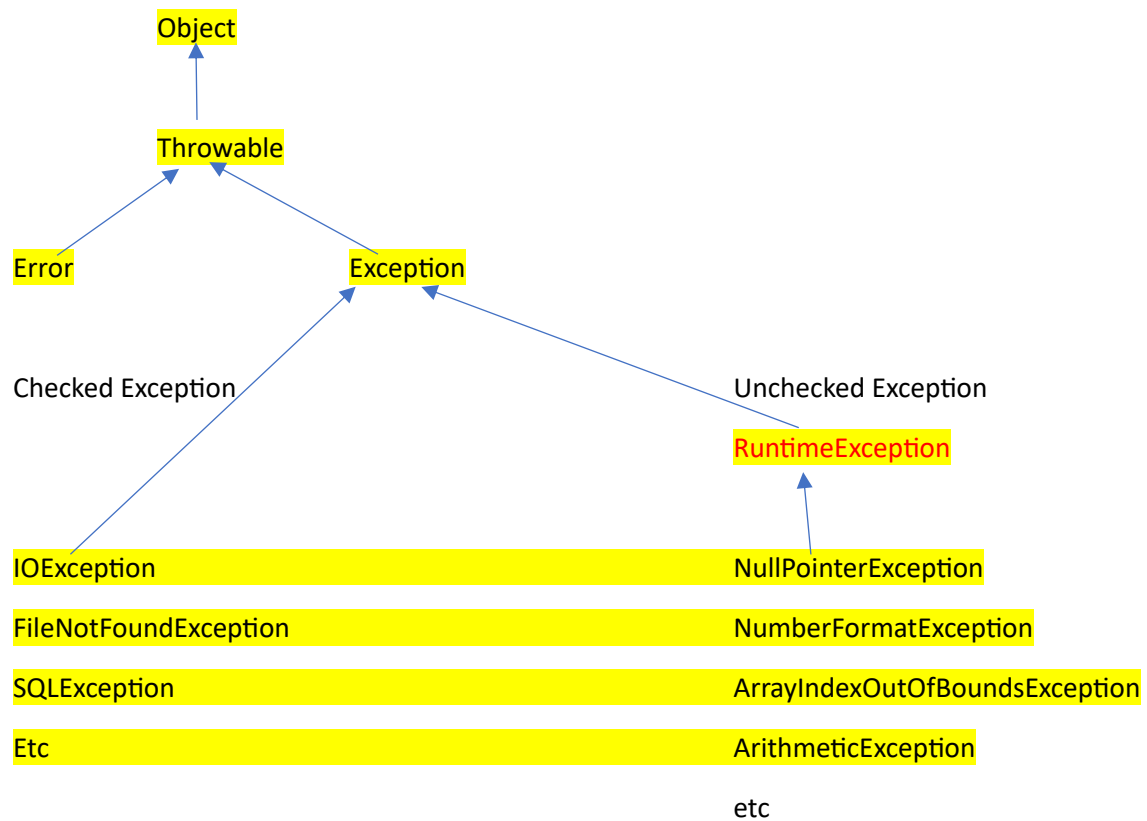
Which contains byte code.



In Java Error and Exception both are pre defined classes part of lang package.

**Error** : The error which generate at rum time which we can't handle it. JVM crash, software or hardware issue, out of memory.

**Exception** : The error which generate at rum time which we can handle it. Divided by Zero.



To handle both the type of exception java provided 5 keyword.

1. try
2. catch
3. finally
4. throw
5. throws

we will handle un checked exception using try catch block

syntax

```
try {
```

```
}catch(Exception e) {
```

```
}
```

## Try with multiple catch block

1. **try with single catch block** : if any exception generate we want to perform common task. Then try with single catch block advisable. In this option is good if we don't which code generate which type of exception.
2. **try with multiple catch block** : base upon type of exception we want to execute set of code then we need to use try with multiple catch block. In this option you must be know which code generate what type of exception.

```
try {  
  
  
} catch (ArithmeticException e) {  
  
  
} catch (ArrayIndexOutOfBoundsException e) {  
  
  
}
```

finally keyword or block.

**try block** : the code which may generate the exception we need to keep in try block.

It one line code or more than one line code.

**catch block** : this block execute if any exception generate. No exception no catch block.

**finally block** : it is a type of block which will execute 100% sure if any exception generate of not.

```
try  
  
catch      catch      catch      catch      finally  
  
           catch      finally      catch  
  
           catch      catch  
  
                        finally
```

file handling program

database connectivity program

```
try {  
    open the file  
    read and write operation  
  
}catch(Exception e) {  
  
}finally {  
    Close my resources ie file handling or database connection.  
}
```

**throw:** throw keyword is use to raise or generate any pre defined or user defined (customer exception) depending upon the conditions.

Syntax

```
throw new Exception();
```

or

```
throw new ExceptionSubClass();
```

by default every sub class constructor contains **super()** parameter.

super() parameter is use to call super class empty constructor.

**throws :** throws keyword we use with method signature to throw the exception to caller method.

```
void display() throws Exception, ExceptionSubClass {  
  
}
```



Sleep() method throw checked exception.

If method throw checked exception we need to handle it mandatory otherwise we can't compile program. Checked exception we can't avoid we need to handle using try catch or throws mandatory. Unchecked exception can avoid some extends.

So checked exception check twice compile time as well as run time.

## Multi threading

Program : set of instruction to perform specific task.

Processor : Processor is responsible to execute the code.

Process : time taken to execute the code or program in execution.

Thread : small execution of a process. Thread also known as light weighed process.

It takes less resource or memory of our machine. Process known as heavy weighted.

By default in java inside a main method default thread execute.

To check default thread details.

```
Thread t = Thread.currentThread();
```

Thread is a pre defined class part of lang package which contains currentThread static method which provide default or current thread details.

```
System.out.println(t);          Thread[main,5,main];
```

Main → name of the thread

5 → priority of the thread

Main → group of thread

```
t.setName("My Thread");
```

```
t.setPriority(1);          min 1 and max 10
```

we can set between 1 to 10 we can't set less than 1 or more than 10.

## Multi tasking :



Task1 ---→ welcome to java traininasfasdfasffs

Task 2 -→

Task 3 -→

Copy and paste

C drive

D drive

D drive

E drive

E drive

F drive

Using process base

Using thread base

Multi tasking using thread base is faster than multi tasking is process base.

In java we can create more than one thread using lot of ways.

### 1. Using extends Thread class

- We need to create normal java class and that class must be extends Thread class.
- We need to create Thread class reference or object.
- With help of reference we need to call start() method. start() method is a pre defined method part of thread class. which help to start the thread.
- Start method internally call run() method. run method present in thread class.
- Run method part of thread class which contains empty body.
- If we want to do some custom logic we need to override run method and write the custom logic.

### 2. Using implements Runnable interface.

- We need to create normal java class and that class must be implements Runnable interface.
- Runnable interface contains one method ie run() method and this method is abstract.
- To start thread we need to start method.

- d. But when our class implements Runnable interface we can't get start() method
- e. So we need create Thread class object and pass the Runnable interface reference. And using thread class reference we can call start() method.
- f. Start method internally call run method

Day 6 : 05/11/2023

Life cycle of thread

create ----> runnable state → running state ----> stop

obj1	obj1.start()	run	i = 10
t1	t1.start()	run	i = 10

isAlive() : check the status of thread

sleep() : to pause the flow the thread

wait() : to suspend the thread

notify() : resume the thread

notifyAll() resume more than one thread.

Task :

Pay the bill

Book the ticket

Transfer the amount etc

**Synchronization** : it is concept which help to block or lock the thread.

It is use to allow for only thread to use all resources at time.

To use synchronization concept java provided synchronized keyword.

This keyword we can use with method or inside a method we can use more than  
Synchronized block.

**wait(), notify() and notifyAll()**

these three method use to do inner thread communication.

wait() method is make the thread to wait or suspend.

notify() method is use to resume or call back waited thread.

Pts

More than one thread created in same memory and method must be synchronized.



Task t1 = new Task();

Task t2 = new Task();

3 thread we will add

1 thread

Data -> of type number

Consumer                      need to consume data of any type.

Producer                      need to produce the data.

Resource class

Resource object we will share to consumer and producer.

Consumer

Producer

Main class

## Collection Framework (Data Structure )

Variable

```
int a=10;
```

```
a=20;
```

array

```
int abc[]={10,20,30};
```

**structure** : it is a type of user defined data type which help to store more than one value of different types.

```
class Employee {
```

```
    int id
```

```
    String name
```

```
    float salary;
```

```
}
```

```
Employee emp = new Employee();
```

```
emp.id=100;
```

```
emp.name="Ravi";
```

```
emp.salary = 12000.50f;
```

**array object**

```
Employee employees[]=new Employee[100];
```

```
employees[0]=new Employee();
```

```
employees[1]=new Employee();
```

```
employees[99]=new Employee();
```

```
int abcd[]=new int[100];
```

Employee java bean class : id,name,salary, designation

Receive more than one employee details

And base upon designation update salary

If desg is manager 5000 bonus

If desg is developer 3000 bonus

Else

1500

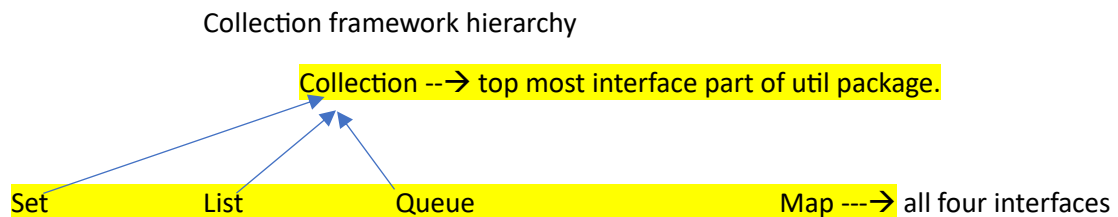
Draw back of array object.

Array is known as fixed in memory size.

If we want to add or remove any data from array it is very complex.

Collection framework provided set of collection of classes which help to store collection of object of any type like int, float , char, double, string or user defined object.

It provided lot of pre defined method which help to add, remove, search, iterate very easily.



Set, List, Queue internally extends Collection but map doesn't extends.

Set : it doesn't allow duplicate. Set doesn't provide index position to access the value.

**HashSet,**

**LinkedHashSet**

**TreeSet**

**These top classes internally implements Set interface.**

List : it allow duplicate. List allow index position

**Stack**

**ArrayList**

**LinkedList**

**Vector**

**These top classes internally implements List interface.**

Queue : by default first in first out feature provides

**LinkedList**

**PriorityQueue**

Map : it will allow to store information in key-value pairs. Key is unique and value can be duplicate. Using key we can get the value.

**HashMap**

**LinkedHashMap**

**TreeMap**

**HashTable**

Day 7 : 18/11/2023

HashSet : Unorder

LinkedHashSet : Order

TreeSet : sorting by default asc TreeSet internally implements SortedSet interface that interface provided logic do to sorting.

List API

Stack : Stack is a type of data structure which provide the features push, pop, peek etc.

ArrayList Vs Normal Array

1. Normal array allow to store same type of values but array list by nature we can store any types of values.
2. Normal array fixed in memory size. ArrayList dynamic memory.
3. Normal array doesn't provide any method to do some operation. ArrayList provide lot of pre defined method to add, remove, search etc.

## LinkedList

LinkedList is a type of List API which internally use Node to store the data.

Type of linked list

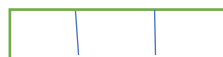
1. Singular linked list
2. Double linked list
3. Circular linked list

10      null



Value ref

null   10   null



pref   value   nref

By default java LinkedList consider as double linked.

1      2                                  to                                  100

0 pos   200 ---element in 1 post

LinkedList is good for insertion and deletion

ArrayList is good for retrieve operation

Vector : Vector is a type of legacy class. in Vector by default all method are synchronized.

Thread safe but slow performance.

Map : key – value : key can be unique and value can be duplicate.

HashMap	unordered
LinkedHashMap	order
TreeMap	asc order as key
Hashtable	by default method are synchronized.

Hashtable , HashMap

Retrieving the value from collection classes one by one

1. For each loop
2. Iterator – interface
3. ListIterator – interface
4. Enumeration – interface

Collection framework with generics

```
//CollectionClass<type> object = new CollectionClass<Type>();
```

Type can be Integer, Float, Character, Double, or any user defined objects. Like Employee, Customer, Manager etc.



Collection Framework with user defined objects.

Employee, Customer, Manager, Account etc.

Product -> bean class variable, setter, getter, constructor etc

ProductService -> service class which contains business logic

Add, remove, retrieve, update etc.

Service layer not to interact with input device. Don't take the value in service layer using scanner class. In simple don't use system.out.println("IN Service layer ");

Day 8 : 25/11/2023

JDBC

In VM

Open terminal

Write as

sudo mysql -u root -p

Simplilearn

**JDBC** : Java Database Connectivity : JDBC provided lot of pre defined API (application programming interface) which contains lot of classes and interfaces which help to connect database ie mysql or oracle o db2 etc using java technologies to do CRUD Operation ie create, insert, update, delete and retrieve etc.

Steps to connect the database.

1. We need to import java.sql or javax.sql package.
2. Jdbc always throw checked exception so we need to handle the exception using try-catch or throws mandatory.
3. Load the Driver. Driver is pre defined class provided by vendor whose database we are planning to connect.

4 types of drivers.

a. Type 1 or jdbc odbc bridge driver

b. Type 2 or jdbc native api driver      database dependent driver. Performance fast but database dependent.

c. Type 3 or jdbc net protocol driver : we need server to connect the database. ie data source

d. Type 4 or jdbc thin driver or pure driver : we need to download jar files provided by vendor.

From java 8 onward type 1 deprecated.

`Class.forName("drivername");`

Class is a pre defined class name itself is Class part of lang package. Which contains forName static method. which help to load the class or driver

MySQL 5.x version    `com.mysql.jdbc.Driver`

MySQL 8.x version    `com.mysql.cj.jdbc.Driver`

4. Connect the database.

`Connection con = DriverManager.getConnection(url,username,password);`

Connection is interface which provide set of method which help to give the reference of Statement or PreparedStatement.

DriverManager is a pre defined class part of sql package. Which contains getConnection static method which takes 3 parameter

1<sup>st</sup> url

2<sup>nd</sup> username of database

3<sup>rd</sup> password of database

getConnection method return type is Connection interface reference.

5. Now we create Statement interface reference.

`Statement stmt = con.createStatement();`

Statement interface provided set of methods which help to do some operation.

a. DML Operation : insert, delete and update

`int result = stmt.executeUpdate("DML Operation");`

if record inserted or deleted or updated result hold value > 0. Else for insert exception for delete or update zero.

b. DRL or DQL : select query

```
ResultSet rs = stmt.executeQuery("select clause")  
while(rs.next()) {  
  
}
```

Statement : Statement is use to execute static query

Using Statement if we execute any query. Query compile on java side send to database side and execute and get acknowledgement.

PreparedStatement : PreparedStatement is use to execute dynamic query. It support parameterized query concept.

Query compiles only once in java side and execute n number of time in database depending upon value.

```
Statement stmt = con.createStatement();  
stmt.executeUpdate("DML") or stmt.executeQuery("DRL");
```

```
PreparedStatement pstmt = con.prepareStatement("DML or DQL");
```

Limitation

1. Adding jar files manually in project
2. Every IDE follow their own project structure.
3. Creating documentation

**Build tool** : build tool is responsible to build the application. Compile the program, run program, creating jar, war , ear etc. adding jar file or dependencies as well as maintaining the dependencies. Creating the documentation.

ANT

**Maven**

**Gradle**

Maven build tool.

Spring and Hibernate

So we can create maven project using command prompt as well as we can create using any IDE.

Group id : collection of more than one project

Artifact id : project name

Maven build tool use pom.xml file (project object model) this file contains all configuration details to build the application.

#### Maven goal

mvn clean	clean the project
mvn compile	compile all files
mvn test	testing
mvn package	creating jar or war or ear by default jar consider
mvn deploy	deploy the project in server

**DAO layer** : Data Access Object : this class contains more than one method which is responsible to write the JDBC code.

By default Every Java program extends Objects class.

Day 9 : 26/11/2023

Creating ProductManagementSystem app using maven tool with bean layer, service layer, dao layer, resource layer and main class.

In Database ie table

**Product** table

Pid, pname, price etc column

Programming language ie Java

**Product** → java bean

pid, pname, price, setter and getter method

Constructor ie empty as well parameter etc

**ProductDao**

**Data Access Object**

Which contains more than one method which help to do some operation using jdbc on product table. storeProduct, deleteProduct, updateProduct, retrieveProduct etc.

**ProductService**

This class contains more than business method

To storeProduct, deleteProduct, updateProduct, retrieveProduct etc.

**Main class** which contains main method interact with input device ie keyword.

If we display any user defined class reference in println it internally call Object class toString method.

That method return object in string format as [packageName.className@code](#)

If we want meaningful object then we need to override toString method in our user defined class.

**resource layer**

this layer contains resource details

like database connection, security, file handling details etc

## Java 8 features

1. From java 8 onward interface can contains method with body.  
But we need to use the keyword as **default or static**.  
Whichever class implements any java8 interface that class must be provide the body for only abstract method not for default or static.  
If we want we can override default method but not static method.  
Default method as well as override default method we can call with the help of object of that class. The class which implements that interface.  
Static method part of interface we need to call through interfaceName.methodName.  
We can add more than one abstract , static as well as default.

Bank interface

Set of methods

rateOfInterest : default implementation 8%      or      static 8%

HDFC implements Bank

HSBC implements Bank

Collection -> interface added any methods default methods  
Set      List      Queue

2. **Functional interface** : the interface must be contains only one abstract method it can contains more than one static as well as default but only one abstract that type of interface is known as functional interface.
3. **Functional interface is use to achieve lambda expression.**
4. **Lambda expression Lambda expression is like anonymous function or method. using lambda expression we can achieve functional programming language features in java.**

## Syntax

**InterfaceName objectRef = (parameterList)->statement**

## Java 11 features