Java

1. Java is a platform independence
2. It was developed in ~~1985~~ 1995 (James Gosling).
3. Features
   - platform independence
   - scalability
   - portability
   - security etc
4. ~~JDK~~ JDK
   ↓
   JRE
   ↓
   JVM

1. What is Java?

Sol 1. Java is popular high-level, class-based objectoriented language
2. Java is platform independent because it is different from other language like C, C++
3. fast, secure & reliable
4. It is widely used for developing the Java Application

Java used for

Important Application

1, It is used for developing Android APPS
2, Help to create Enterprise software
3, wide range of Mobile Java Application
4, Scientific computing Application
5, use for Big Data Analytics
6, Java programming of Hardware devices.

In 1995, sun changed the name to "Java" and modified the language to take advantage

In 2009 oracle corporation Acquire

Java Application

Desktop Application such as 1. acrobat reader
2. Media player
3. antivirus etc

Web Application such as 1. Irctc
2. co.in etc.

* Enterprise Application such as Banking Application

1. Mobile
2. Embedded System 到 Celiro
3. Smart cards
4. Robotics
5. Games etc.

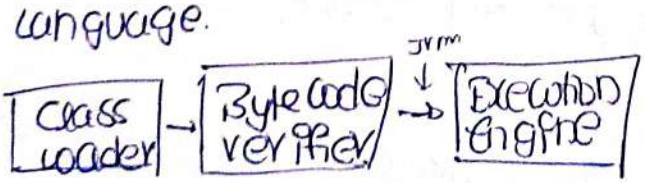
Features

Secured
Portable
Simple
Object-oriented
High-performance
Plat-independence
Robust

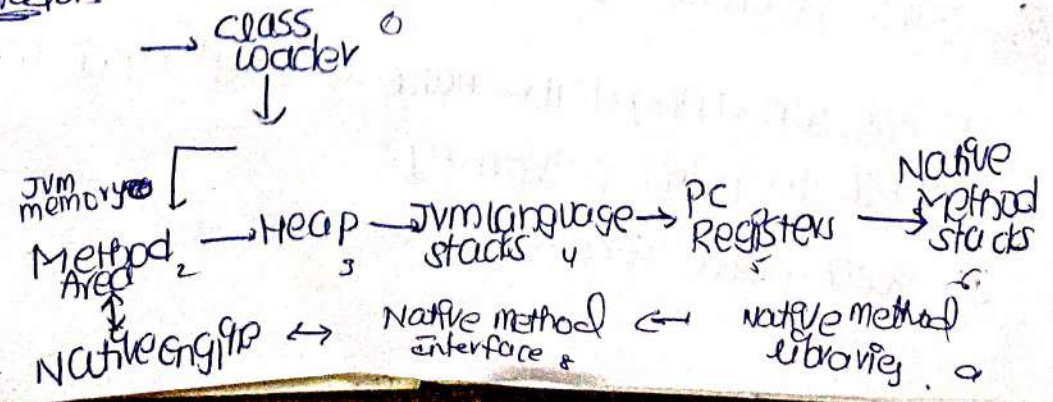
Architectural neutral, Interpreted, Multithreaded, Distributed
Dynamic.

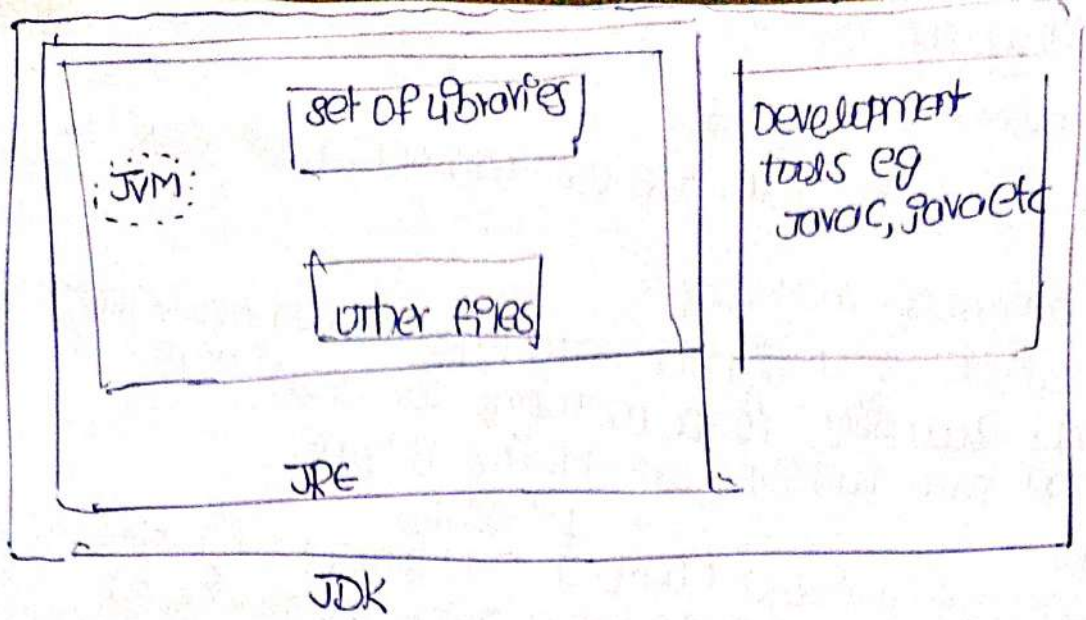
What is JVM?

It is used to converts Java Bytecode into machine language.



JVM Architecture






```
class Hello World
{
public static void main(String[] args)
  {
  System.out.println("Hello");
  }
}
```

class keyword is used to declare class in Java
public keyword is access modifier represents visibility visible to all
static is a keyword. If we declare any method as static is know as static method

i. we can override the method,
ii. we can't create object.
iii. we can't change the method
iv. it is unique path.

void is the return type of the method.
If we use int method then we need require "return"

- main represents the starting point of program
  String[] args or String args[] is used for command line argument.

System.out.println() is used to print statement.

println is comes "printstream class"

Java Installation

~~Path~~ Environmental set-up

for setting the path we use Environmental setup.

1. What is a variable in Java?

A. Every variable is assigned data type which designates the type and quantity of a memory location name of data. the Java variable are many 3 types.

they are:

1. Instance variable (Demo d = new Demo() (fn outside variable) ↓ Instance variable
2. local variable (function inside the variable)
3. static variable. (we can't create the object because is final)

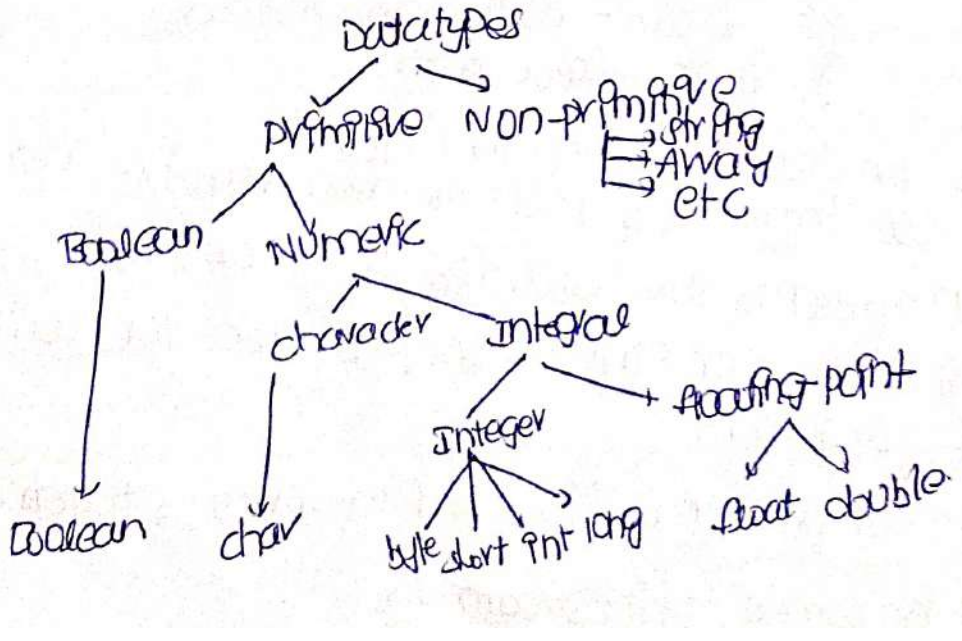
Eg:

```
class Demo
{
static int a = 1; //static variable
int data = 99; // Instance variable
void method()
{
int b = 90; // local variable
}
```

*Data type is defined for assigned the variables

they are

1. primitive Data type → Integer, character, Boolean, float
2. non-primitive Data type → class, array, Interface



Byte → 1 Byte
short → 2
Int → 4
long → 8
float → 4
double → 8

Byte $\xrightarrow{1}$ short $\xrightarrow{2}$ Int $\xrightarrow{4}$ float $\xrightarrow{4}$ long $\xrightarrow{8}$ double

Note

Pre-increment will be change

```
a=4
++a
```

O/P: 5

Post-Increment will not change

```
a=4
a++
```

O/P: a=4

```
a=5
print(a++);
print(a++);
```

O/P: 5
6.

Ex: a=25

```
print(a++)
print(++a)
print(a--)
```

O/P: 25
27
27

O/P: 25 Output = 25
27
27

a=3	O/P:	Output
(a++);	3	4
(a++);	4	5
(++a)	6	6
(--a)	5	5

Typecasting

converting one datatype to another Datatype

```
Doubled d=10;
int i;
i= (int) d
```

↑ Type cast operator.

formula

1, a<<b = a*2 Math(b)

2, a>>b = a/2 Math(b) ⇒ $\frac{a}{2b}$

shift operator formula
left shift = a<<b = a*2 powerb.

*without operator we can find Minimum and Maximum by using Ternary operator

Without using + operator to add two numbers.

```
int z= x-(-1-y)-1; (or) z= x-(~y)-1 (or) ~y= -1-y
sop(z);
```

Java control statements

1. Decision making statements
   if statements
2. Switch statements
   loop statement
   do-while loop
   while loop
   for loop
   for each loop.

Builtin fn

```
if(s.equals("biryani"))
{
}
else
{
}
```

2.
```
import java.util.*;
Public class Main
{
static void restaurant(string s)
{
if (s.equals("biryani"))
{
  sop("");
}
else
{
 sop()
}
Public static void main(string[] args)
{
scanner sc = new scanner(system.in);
string s = sc.next();
restaurant(s);
}
}.
```

O/P: biryani
available.

* Memory area allocated by JVM

1, Class Area
2, Heap
3, Stack
4, Program counter Register
5, Native Method stack

"==" (Note)

5+2 == 7 // False.
7 == 7 // True

(.equals(5+2) == 7) // True.

Switch case
syntax

```
String x = sc.nextLine();
switch (x){
    case "biriyani":
        sop("yes accented");
        break.
    case "water":
        sop(  )
    default:
        sop(  )
```

switch will always checks with "cases"

Ex:2

```
int s = sc.nextInt();
int k = sc.nextInt();
int h = s+k;
switch(h)
{
case 10:
    sop("")
    break
case 20:
    sop("")
    break
case 30:
    sop("")
    break
default:
    sop("")
}
```

```
public class WhileTest
{
Public static void main (String
  args[])
{
int i=5
while(i<=5)
  {
  sop( );
  i=i+2;
  }
}
```

do-while loop:

```
{
i=5
do
{
sop()
i=i+2
} while(i<=20)
}
```

For Each Loop

```
Public class forEachLoop
{
Public static void main
(String args[])
{
int s[]= {18,25, 28,29,30}
for (int i:s)
{
System.out.println(i)
}
}
}
```

What is OOPS?

1, Object

Object-Oriented Programming (OOPS) is a programming concept that works principle of
1. abstraction
2. encapsulation
3. inheritance
4. polymorphism

* object is a Instance of class. It contains the Address.
Ex: chair, pen, Table, bike etc.

It is curious to create object they want and create the method to handle of objects.

2, class

It is only a logical component and the physical entity "collecting of objects"

3, object

1, An object can be defined as an Instance of class.
2, It's real time entity
3, It has state and Behaviour.
(initial assign part ex variables) (Declaration part Ex: f^n)

3, Inheritance:

one object acquire the properties and Behaviour of the parent object. It is creating a parent child relationship Between Two classes.

2. code reusability

4, Polymorphism

object or function to take on multiple forms
one task can do many ways
many forms

i, Method overloading
ii, Method overriding.

5, Abstraction

Hiding the implementation shows functionality is knowns as Abstraction

Internal work will not be specified only o/p it's show result.

6, Encapsulation

private wrapping the data and code
Ex: passwords
It can only be accessing using the methods of their current class.

syntax

class_name object_name = new class_name();
object is created in main.

```
import java.util.*;
Public class Main
{
int empId;
String name;
int age
double salary;
Public static void main(String[] args)
{
Main d= new Main();
d.empId=5;
d.name="ajay";
d.age=20;
d.salary=50000;
System.out.println("empId is:"+d.empId+"\n"+"name is:"+d.name+"\n"+
"age is"+d.age+"\n"+"salary is"+d.salary);
}
}
```

Java allows 3 types of construction

1. Default construction (the display will default value) (construction will not be created)
2. No Argument construction
3. Parametrized construction

1. Default construction (It is created in system by Default) But we can't create.

constructor
It is used to initialize the object

```
class Student3
{
int Id;
String name;
void display()
{
system.out.println("      ")
}
Public static void main(String[] args)
{
student3 d= new student3()
d.display();
}
}
```

2. no argument constructor "construction is created By we not Pass any value"

```
class Demo
{
Public Demo()
{
system.out.println("this is no argument constructor");
}
}
Print()
{ new Demo()
```

this Keywords

Java this keyword in Java. this is a reference variable that "refers to the current object"

3. Parametrized constructor

```
Public class employee
{
int empId;
String empName;
* Parametrize
Logic Employee(int id, string name)
{ this.empid = id;
  this.empName = name;
}
void info()
{
system.out.println("Id:"+empId+"Name: "+empName);
}
Public static void main(string args[])
{
employee obj1 = new employee(1024, "Ajay");
employee obj2 = new Employee(9223, "Negan");
obj1.Info();
obj2.Info();
}
```

Parametrized constructor

```
public class Bank
{
int ac;
String name;
String bnkname;
public Bank (int ac,string name, string bnkname)
{
this.ac=ac;
this.name=name;
this.bnkname=bnkname;
}
public static void main(string[] args)
{
Bank d=new Bank (1, "ajay","SBI");
Bank d2=new Bank(2, "sk","ICICI");
System.out.println("ac is"+d.ac+"\n"+ "name is"+d.name+"\n"+
"bnkname is"+d.bnkname);
}}
```

O/P:

ac is 1
name is ajay
bnkname is SBI

Disadvantages of constructor

1. Not Returing any value
2. constructor is always public

Java Inheritance Example Programe.

```
class calculation
{
int z;
Public void addition(int x, int y)
{
z=x+y;
System.out.println("The sum of given number:"+z);
}
Public void subtraction(int x, int y)
{
z= x-y; sop(z);
}
```

Inheritance

child class Acquire the properties the parent class properties

Properties

Types (forms)

Single-Level "
Multi-Level "
} main
Hierarchical Inheritance
Multiple Inheritance
Hybrid Inheritance

Sample syntax for single Inheritance

```
class A
{
}
class B extends class A
{
}
```

Multiple-Level Inheritance (It like a chain process)

```
class A
{
}
class B extends class A
{
}
class C extends class B
{
}
class D extends class C
{}
```

Inheritance

Note: we can't use the public for class

```
class square
{
int z;
public void sqrt(int x)
{
z=x*x;
system.out.println(z);
}
}
class Rect extends square
{
int z;
public void rect( int x,int y)
{
z=x*x;
system.out.println(z);
}
}
public class Main'
{
public static void Main(String[] args)
{
Rect d = new Rect();
d.sqrt(10);
d.rect(3,4);
}}
```

O/P:

100

12.

Polymorphism

one Task can do in Many form

they are two types

1. compile polymorphism
2. Runtime polymorphism

compile polymorphism

It is also knows as static Polymorphism or Method overloading

occurs when the compile Determine which method to invoke

Based on number of argument passed to it

The Method with have same name and Different parameter.

Runtime polymorphism

Runtime polymorphism is a Dynamic polymorphism or method overriding occurs when the method to involved determine at runtime based on the actual type of object.

This Means that Method defined a superclass can be overridden in subclass to provide a specific implementation.

Methods

1. Method overloading → (Same name but different parameter) method
This is when a class has Multiple Methods with the same name, but different parameters.

2. Method overriding This is when a subclass provide its own implementation of a Method that is already defined its parent class

Polymorphism

1, Method overloading

```
class Example
{
void operator(int a, int b)
{
int z;
 z= a+b;
System.out.println(z);
}
void operator(string s1, string
{
string s3;
s3 = s1+s2;
System.out.println(s3)
}
public static void main(string[] args)
{
Example h = new Example();
   h.operator(3,5);
   h.operator("a", "ay");
}
}
```

O/P: 8
ajay.

Polymorphism

- Compile time
  - Function overloading (or) Method overloading.
  - operator overloading
- Runtime
  - Method overriding

Method overriding :

** Different class same function same parameter.

overriding syntax

```
class HDFC
{
void ROI(int x) {
        SOP("the rate of first is" ))
}
} class ICICI
{
void ROI (int x) {
SOP()
}}
class SBI {
void ROI (int x)
{
SOP()
}
}
class Bank
{
PVM
HDFC h=new HDFC();
h.ROI();
ICICI I= new ICICI();
I.ROI();
SBI S= new SBI();
S.SBI();
}}
```

```
class HDFC
{
Public void roi(int x)
{ option
system.out.println(x/3);
}}
class ICICI
{
Public void roi(int x)
{ option
system.out.println(x/4);
}}
class sbi extends Icici
{
Public void roi (int x)
{ option
system.out.println(x/2);
}
Public static void main(String[] arg
Hdfc h=new Hdfc();
h.roI (9);
Icici i= new Icici;
i.roi (8);
Sbi s= new sbi();
s.roi (6);
}}
```

Ex: class Hdfc

```
{
void roi(int x)
{
System.out.println(x/3);
}}
class Icici extends Hdfc
{
void roi (int x)
{
System.out.println(x/4);
}}
class Sbi extends Icici
{
void roi(int x)
{
System.out.println(x/2);
}}
public class Main
{
public static void main(string[] args)
{
Hdfc h= new Hdfc();
h.roi(9);
Icici i= new Icici();
i.roi(8);
Sbi s= new Sbi();
s.roi(6);
}
}
```

Access Modifiers

1. Default
   Default Declaration are visible only within the package
   (package private)
2. Private
   private declaration are visible within the class only

3. Protected

Protected Declarations are visible "within the package or all subclasses"

4 Public

Public Declaration are visible everywhere.

Abstraction class

Abstraction - hides implementation shows functionality.

Abstraction class Explanation :

1. A class which contains the abstract keyword Pnit's Declaration is known as abstract class.

2. Abstract class May or May not contain abstract Method i.e, Methods without body (Put get();)

3. But if a class has at least one abstract Method, then it Must be declared as abstract.

4. We can't created the object for abstract class.

Important points

It can have abstract and non-abstract Methods

It can have constructors and static Method also

It can have final method which will force the subclass not to change the body of the Method.

Abstract class with abstract Method (subclass can do operation, hiding the implementation, it like a stave)

```
abstract class MotorBike
{ storage Purpose we can access it.
abstract void brake();
}
class SportsBike exteds MotorBike
{
public void brake()
{
System.out.Println "sports Bike Brake"
}}
class MountainBike extends MotorBike
{public void brake();
} sop()
}
```

```
class Main
{
public static void main(String[] arg
{
MountainBike a = new Mountain();
a.brake();
SportBike s1 = new SportBike();
s1.brake();
}
}
```

Interface

1. It is a Blue print of class.
2. Interface in java are a set of abstract and public Methods. We want to our class it.
3. Interfaces are used to achieve abstraction and implementation Multiple Inheritance
4. An Interface is a fully abstract classes.
5. We use the interface keyword to create an interface in java for example to use an interface other class Must implement.

Note: Interface is achieve multiple Inheritance.

Syntax:

```
interface language
{
public void getTypec();
Public void getversionc();
}

Interface A
{

}
Interface B
{

}
class c implements A, B
{
// abstract Members of A
// abstract members of B

}
```

6. they are two keys to Interface.

1. class c implement A, B } different
   class interface

2. 1. class extends class
   2. interface extends interface
   3. class implements interface
   4. interface implement class

Abstract class

1. Abstract class can have abstract and non-abstract Methods.

Interface

i, Interface can have only abstract methods. , It is also support multi-Inheritance

ii, Abstract class Does n't support multiple Inheritance.

iii Abstract class can have final, non-final, static and static variables.

iv, Interface has only static and final variables.

v, Abstract class can provide the implementation of Interface.

vi, Interface class can't provide the implementation of class.

Abstract class	Interface
1. Private protected	1. Public, default.
2. Multiple Inheritance is possible	2. It is possible.
3. Abstract class can't be implement	3. It is Implement
4. abstract keyword	4. Interface keyword.
5. Subclass extends abstract class.	5. Subclasses implement interface.
6. We can extend only one abstract class	6. We can implement Multiple Interface
7. fast	7. slow
8. All Access Modifiers.	8. Only Public Access Modifiers.
9. Static and non-static.	9. Static
10. To avoid Independence	10- For future enhancement

Interface Program

```
interface Polygon
{
Public void getArea();
}
Class Square implements Polygon
{
Public void getArea()
{
int s=10;
System.out.println(s*s);
}
}
Class Rectangle implements Polygon
{
Public void getArea()
{
int l=3
int b=4
int h=l*b;
System.out.println(h);
}
}
class Main
{
Public static void main(String[] args)
{
Square s=new Square();
s.getArea();
Rectangle r=new Rectangle();
r.getArea();
}
}
```

O/P: 100
12.

Encapsulation (It is used in Business Purpose)

It is a process of wrapping code and data together into a single unit,

Ex: a capsule which is mixed several medicines.

2. Encapsulation in java can be achieved by.

i. Declaring the variables of a class as private

ii. Providing public setter and getter methods to modify and view the variables value.

3. Provide public setter and getter methods to modify and view

```
Public class Encaptest
{
private string name;
private string id Num;
```

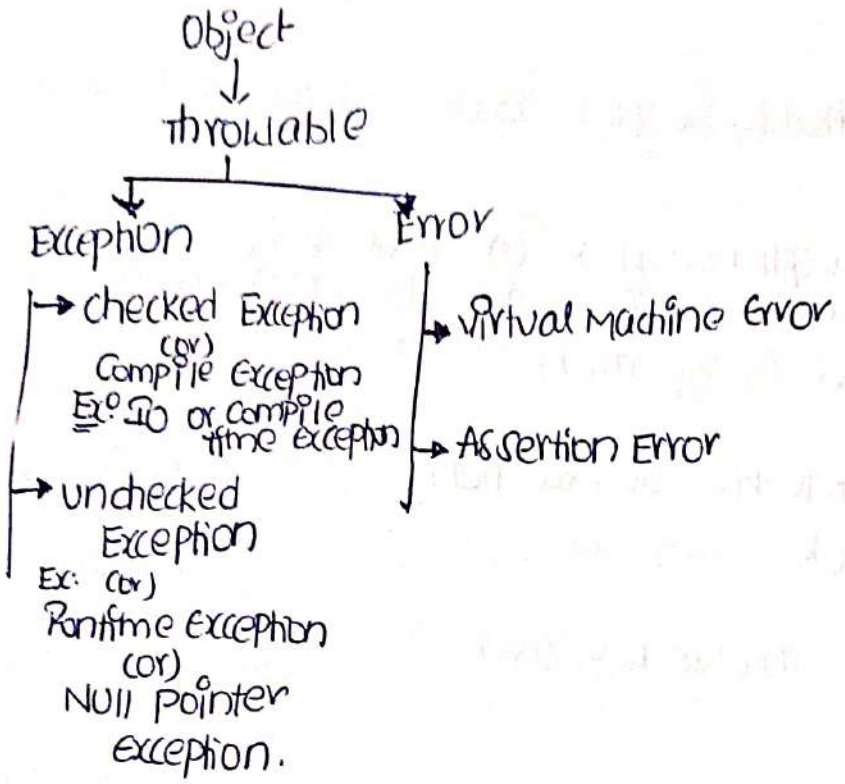
Program:

```
public class Carservice
{
private int carno;
private string carcolor;
private string companyname;
private int caryear;
private int billnumber
public int getcarNo()
{
return carno;
}
public string getCarColor()
{
return carcolor;
}
public string getCompanyName()
{
return companyname;
}
public int getcaryear()
{
return caryear;
}
public int getBillNumber()
{
return billnumber;
}
```

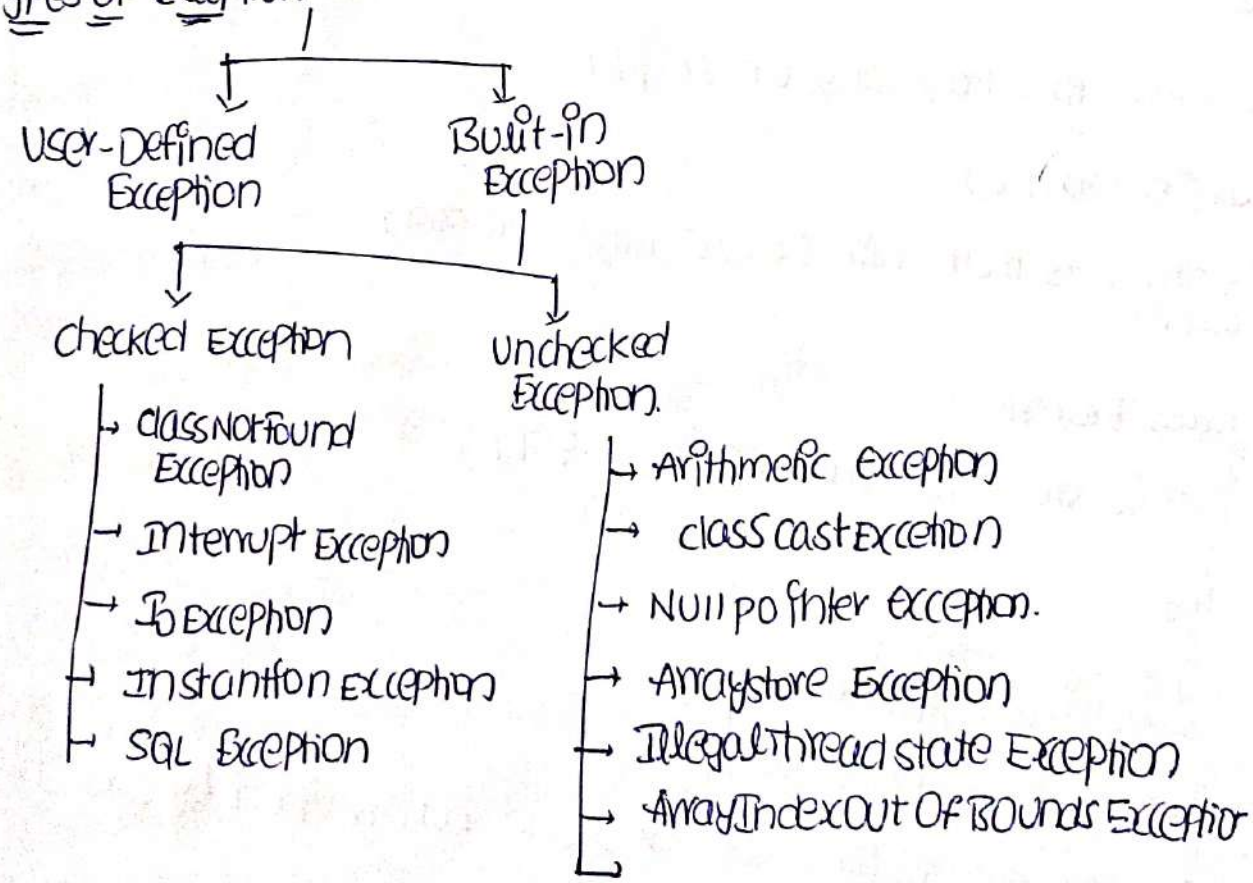
```
Public void setcarNO(int newcarNO)
{
carno= newcarNO;
}
Public void setcarcolor(int newCarColor)
{
carcolor= newCarcolor;
}
Public void setCompanyName(int new CompanyName)
{
companyname= newCompanyName;
}
Public void setcaryear(int new Caryear)
{
caryear= newcaryear;
}
Public void setBillNumber(int newBillNumber)
{
billnumber = newBillNumber;
}
}

class Main
{
Public static void main(string[] args)
{
carServices cs = new CarSewice();
cs. set carNO(888);
cs. get car color("black");
cs. get CompanyName("Ajaykawe");
cs. set caryear(2022);
cs. set Bill Number(1001);
system. out. Println(cs. getcarNo());
system. out. println(cs. getCompanyName());
system. out. println(cs getCarColor());
}
}
```

"Exception Handling"

1. An exception is a problem that arises during the execution of a program
2. When an Exception occurs the normal flow of the program is disrupted and the program/Application terminates abnormal.



Types of Exception



Checked Exception

Checked exception is a exception that is checked by the compilation-time
They are also called as compile-time Exceptions.

2. These exceptions cannot ignored, the programmer should take care of these Exceptions.

Unchecked exception

An unchecked exception is an exception that occurs at the time of

these are runtime Exceptions. These are include bugs such as logic improper use of API. Runtime Exception are ignored by time-of-compilation.

Different approach to Handle Exception

1. Try catch block (Multiple catch is possible)
2. final block
3. Throw and Throws keyword.

Syntax

```
try
{
//statement that may cause an exception
}
catch (Exception e)
{
// statements that will execute when exception
occur
}
```

Ex:1,

```
class Example
{
PUBLIC static void main(string args[])
{
try
{
int num= 121/0;
system. out. println(num);
}
catch (ArithmeticException e)
{
System. out. println( Number should not divided by zero:");
}
}
```

Ex:2, class Main Ex: multiple catch

```
{
public static void main(String[] args)
{
try
{
int n=5;
int ar[]= new int[n-1];
ar[5]= 45/5;
System.out.println("ok");
}
catch (ArithmeticException e)
{
System.out.println("number can not divisible by zero ");
}
catch(ArrayIndexOutOfBoundsException e)
{
System.out.println("warning ArrayIndex OutOf Bounds Exception");
}
} catch (Exception e)
{ System.out.println("warning");
}
}
```

O/P:
warning ArrayIndex OutOfBound Exception.

Ex:(Try inside Try)

Ex:3, class Main

```
{
public static void main(String[] args)
{
try
{
try
{
String s=null;
System.out.println(s.length());
}
catch (NullPointerException e)
{
System.out.println("enter the correct string");
}
int n=5;
int ar[]=new int[n-1];
ar[5] = 45/5;
}
catch (Arithmetic Exception e)
{
System.out.println("number can not divisible by 0:");
}
catch (ArrayIndexOutof BoundsException e)
{
System.out.println("warning ArrayIndexoutof BoundsException");
}
```

```
catch(Exception e)
{
System.out.println("exception");
}
}
}
```

O/P:

enter the correct string
warning ArrayIndexOutOfBoundsException.

Simple

```
try
{
try
{
arry a = new arry
SOP();
}
catch()
{
z = 8/0;
SOP
}
```

finallyBlock:

The block execute whether an exception occurs or not. you should place those statement is finally blocks, that must be execute whether exception occur or not.

" After catch block "

```
finally
{
system.out.println("the Block must be executed");
}
```

Note: 1

Without catch the final Block will be work

throw keyword

The throw keyword Java is used to throw an exception explicitly form a method or any Block

When an exception is thrown, the flow of program execution Transfer from the try block to Block. An exception can be either checked or unchecked.

Syntax:

```
public void method()
{
// an unexpected event occured
throw new customerException("Message");
}
```

Program:

```
public class thrownEx
{
static void vote (int age)
{
if (age<18)
{
throw new ArithmeticException("age is not valid");
}
else
{
System.out.println("age is valid");
}
}
public static void main(string[] args)
{
vote(13);   (or)   ThrownEx e= new ThrownEx();
                   e.vote(13);
}
}
```

O/P: age is not valid.

Note

We can access the method by created the object (or) call the method.

Ex:2

```
Public class ThrownEx
{
Static void vote(int age)
{
try
{
if(age<18)
{
throw new ArithmeticException("age is not valid");
}
else
{
System.out.Println ("age is valid");
}}
catch(ArithmeticException e)
{
System.out.println("exception");
}}
Public static void main(String[] args)
{
ThrowEx e=new ThrowEx();
e.vote(19);
}
}
```

O/P:
age is valid.

By using throw we get the o/p of some error to handle the error we are using try and catch.

2. For try exception handle by catch
3. For throw exception handle by throw area
4. for thrown exception handle by throws area.

Throws

throws is a keyword in Java that is used in the signature of method to indicate that this throw one of listed type exceptions.

The caller to these methods has to handle the exception try-catch block.

Syntax

type method_name (parameters) throws exception_list

Ex:

```
public class throwsExample
{
int division(int a, int b) throws ArithmeticException
{
int z= a/b;
return z;
}
}
```

Ex: 2, public class throwsEX

```
{
int division(int a, int b) throws Exception
{
int z= a/b;
return z;
}
public static void main(string[] args)
{
throwsEX e=new throwsEX();
int c= e.division(6,3);
system.out.println(c);
}
}
```

O/P: 3.

Ex: 3, public class throwsEX

```
{
int division(int a, int b) throws ArithmeticException
{
int z= a/b;
return z;
}
public static void main (string [] args)
{
throwsEX f= new throwsEX();
try
{
int c= f.division(6,0);
}
catch (ArithmeticException e)
{
system.out.println("exception");
}}}
```

O/P: exception

Note: try & catch Method used in inside the Main

Note

1, throw is used to throw an exception explicitly

2, throws to Declare one or more exceptions, separated by commas.

Ex: int foo name(int a, int b) throws Exception1, Exception2, ... Exceptionn

3, Throw can used in single exception.

4, Multiple can be thrown in throws.

5, throw keyword is followed by the instance variable

6, throws keyword is followed by the exception class.

7, throw is used inside the method.

8, throw is used with method signature.

9, Syntax of throw

```
int divide()
{
throw new ArithmeticException();
}
```

10, Syntax of throws

```
int divide() throws ArithmeticException()
```

**

Java String Methods:

string: (is a object in java)

strings, which are widely used in java programming are a sequence of characters. In java language, strings are treated as objects.

Creating strings

the most direct way to create a string is to write

```
string greeting = "Hello world!";
```

Ex:1, public class stringDemo

```
{
public static void main(string args[])
{
char[] helloArray = {'h', 'e', 'l', 'l', 'o'};
string helloArray2 = new helloArray2(helloArray);
system.out.println(helloArray2);
}}
```

There are 2 ways to create string object.

1. String literal
2. By new keyword.

String literal

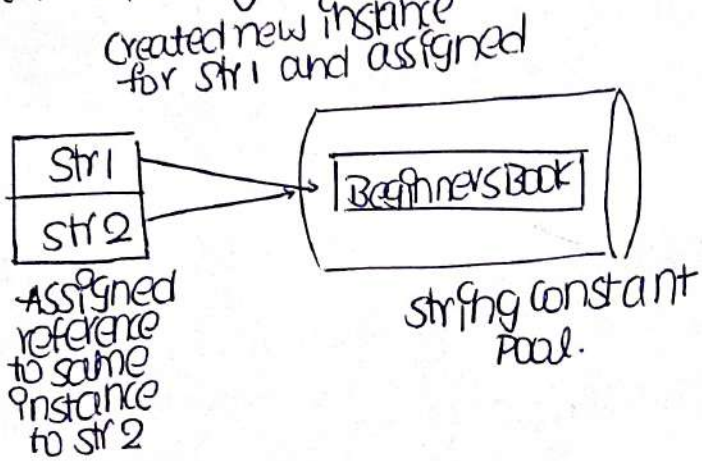
A string literal is a sequence of character enclosed in double quotation marks (" "). In java be created by assigning string literal to string instance.

```
String str1 = "Book";
String str2 = "Book";
```

} Problem Because Both are emory stored in same Memory.

To reduce the problem we are using the "new keyword". It is access Different Memory

Ex:1,



Note:

1. String is a immutable
2. It can has a special (or) separate. Memory area have like string pool, same object will not refer two separate variable.

2. Using New keyword

To create a new instance of a string, we use new keyword

Ex: string str3 = new string("Book");
string str4 = new string("Book");

Ex: string fn

1, compare()
2, concentrate()
3, split()
4, equals()
5, length()
6, replace()
7, compareTo()
8, intern()
9, substring()

These are 40+ built-in fn of string in Java

Ex:1,

```
Public class Main
{
Public static void main (string ry args)
{
String s1 = "ajay";
String s2 = "ajay";
String s3 = new string ("ajay");
String s4 = new string ("ajay");
if (s1==s2)
{
system.out.println ("Hello World");
}
else
{
system.out.println("Hi ");
}
if (s3==s4)
{
system.out.println("Hero ");
}
else
{
system.out.println("zero ");
}
}
}
```

O/P:

Hello World
zero.

Note:2 If we use the condition like
if(s1.equals(s2))
if(s3.equals(s4))
then the O/P is
Hello World
Hero
Because it is compare the string but not memory.

1. == is used to compare the memory
2. s3.equals(s4) is used to compare the string

Ex:3

```
class Main
{
public static void main(string[] args)
{
string s1 = "welcome to meritm";
int k = s1.length();
system.out.println(s1.charAt(0));
system.out.println(s1.charAt(k-1));
system.out.println(s1.charAt(4));
system.out.println(k);

}}
```

O/P:

w
m
o
17

Ex: 4,

```
class Main
{
public static void main(string[] args)
{
string s1="welcome to meritm";
string s2= "select your Branch";
system.out.println(s1.concat(" ").concat(s2));
}
}
```

O/P: welcome to meritm select your Branch.

```
string built-in-fn
class Main
{
public static void main (string [] args)
{
string s1= "welcome @to java";
string s2= "ajay";
system. out. println(s1.toupperCase());
system. out. println(s1. toLowerCase());
system. out. println(s1. trim());
System. out. println(s1.codePointAt(13));
system. out. println(s1. replace("j", "b"));
system. out. println(s1. contains("to"));
system. out. println(s2. indexOf("a"));
for (string k= s1.split("@"))
{
system.out.println(k);
}
system.out.println(s1. hashcode());
system. out. println(s1. toCharArray());
system. out. println(s1. compareTo (s2));
system. out. println(s2.substring(1));
}
}
```

o/p: WELCOME @TO JAVA
welcome @to java
welcome @ to java
118
abay
true
0
welcome
to java

-776050807

wa como @ to java

22.
ajay

Reverse the string

```
class Main
{
static string reverseString( String S)
{
if (S.isEmpty())
{
return S;
}
return reverseString(S.substring(1))+S.charAt(0);
}
Public static void main (String[] args)
{
String S= "welcome to java";
String k= reverseString(S);
System.out.println(k);
}
}
```

small

O/P:

avaj ot emoclew.

* we can do immutable value into mutable by using string buffer

String ~~Buffer~~ Buffer and string Builder:

1. Public class Test

```
{
public static void main(String args[])
{
StringBuffer b= new StringBuffer( new StringBuffer("test");
b.append("stringBuffer");
system.out.println(b);
}}
```

Note
insertion
Deletion
reverse
is possible.

2. String Builder

```
class Main
{
Public static void main (String[] args)
{
StringBuilder b= new String Builder("ak");
b.append("sk");
System.out.println(b);
}}
```

→ "same method is stored."

O/P: aksk

String Buffer

1. It is a thread safe and synchronized (we can't share resources)

2. It is used when multiple threads are working on same string.

3. It is ~~faster~~ slow

String Builder

String Builder operations are not thread safe are not synchronized.

2. String Builder is used for single-threaded environment.

3. It is ~~slow~~ faster

Ex: class Main
{
public static void main(String[] args)
{
StringBuilder b= new StringBuilder("ajay");
b.reverse();
}
}

O/P:
yaja.

Realtime Examples

1. Constructor - It is used to create the instance of variable (before the Main)

2.

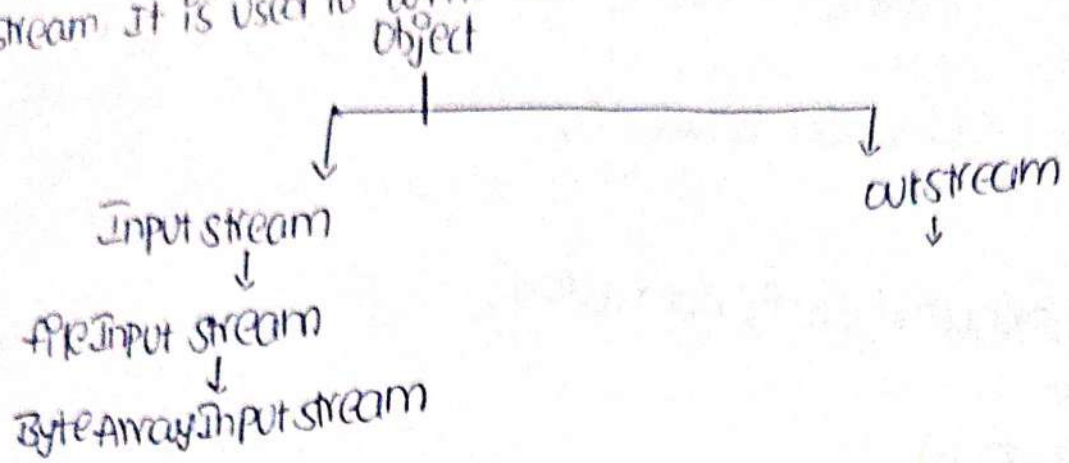
File Input/output:

Java io package contains nearly everyclass you might ever need to perform input and output. All these streams represent an input source and an o/p destination. the Java io package support many data as primitive object, localized character etc.

Stream: It is a sequence of Data transfer.

I/P: It is used to Read the Data

O/P stream: It is used to write Data



Java file operation Methods:

operation	Method	package
To create file	createNewFile()	Java.io.file
To read file	read	Java.io.FileReader
To write file	write()	Java.io.FileWriter
To Delete file	Delete()	Java.io.File.

Ex: import java.io.*;

```
public class Main
{
File f= new File("sample.txt");
package
try
{
boolean v= f.createNewFile();
if (v)
{
system.out.println("file is created");
}
else
{
system.out.println("file is not created");
}}
catch (Exception e)
{
system.out.println("exception");
}
//e.printStackTrace
}
}
```

it is a tool to handle the exceptions and error.

O/P:

file is created.

Writer program:

```
import java.io.*;
public class Main
{
public static void main(String[] args)
{
String d= "this is the data in o/p";
try
{
FileWriter f=new FileWriter("sample.txt");
f.write(d);
system.out.println("copy into file");
f.close();
}
catch (Exception e)
{
system.out.println("Error");
}}}
```

already present file.

O/P: copy into file.

Ex:
```
import java io FileOutputStream;
Public class Main
{
Public static void main(String[] args)
{
FileOutputStream f= new FileOutputStream("sample.txt");
String s= "Welcome to meritm";
byte b[]= s.getBytes(); // convert string into byte array
f.write(b);
f.close();
system.out.println("success...");
}
catch (Exception e)
{
system.out.println("error");
}
}
}
```

O/P:

sample.txt → "It changes to" → "Welcome to meritm."

~~copy data file~~

"This is data in O/P"

Reader

```
import java io.Reader;
Public class Main
{
Public static void main(String[] args)
{
~~FileReader f= new FileReader("sample.txt");~~
char a[]= new char[20];
try
{
FileReader f= new FileReader("sample.txt");
f.read(a);
system.out.println(a);
f.close();
}
catch (exception e)
{
system.out.println("Error");
}}}
```

O/P: welcome to meritm

Collections: (store and manipulate the multiple object).

A Java collection framework provides an architecture to store and manipulate a group of objects. A Java collection framework include the following

1. Interface
2. Classes
3. Algorithm

1. Interface

Interface in Java refers to the abstract data types. they allows Java collections to be manipulated independently from the details of their representation. Also they form a Hierarchy in object-oriented programming languages.

2. classes

classes in Java are the implementation of the collection Interface. It basically refers to the Data structure that are used again and again.

3. Algorithms

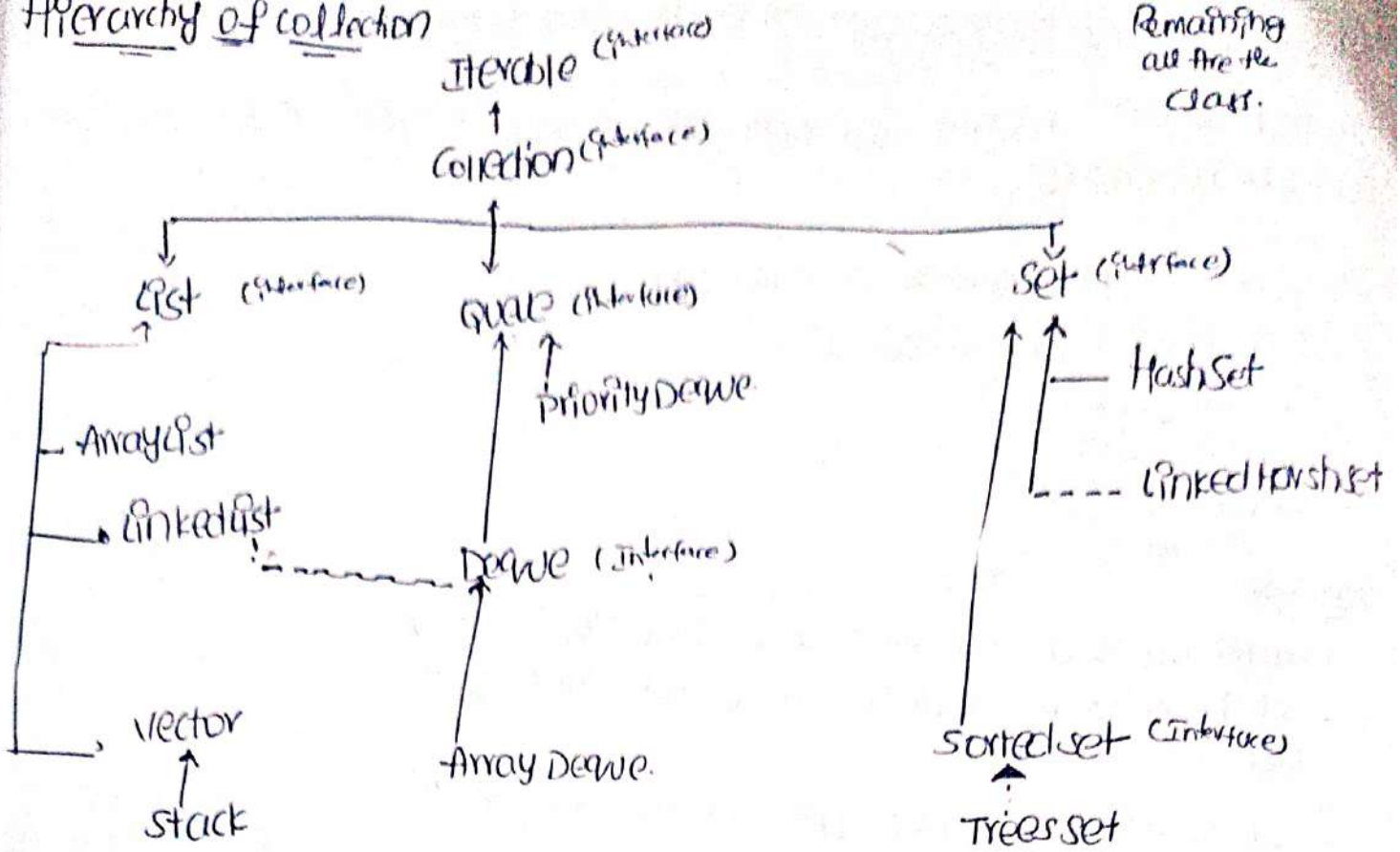
Algorithm refers to the method which are used to perform operations such as searching and sorting, on objects that implement collection interfaces.

Used

1. It is save time
2. It is improve performance.

Hierarchy of collection

Remaining all are the class.



Array	Collection Framework
1. Arrays are fixed in size	1. Collection is not fixed size.
2. Arrays can hold only homogeneous elements.	2. Collection is store both homogeneous and heterogeneous elements.
3. Array memory management is poor.	3. Collection memory is more effective.
4. For any requirement there is no ready method available	4. For every requirement ready made method support is available.
5. faster	5. slower

Ordered List:

A List is an ordered collection of elements which May contain Duplicates.

It is an child interface of collection.
It is a further classified into
1. ArrayList
2. LinkedList
3 vectors.

ArrayList

1. ArrayList is a class present in java.util package.
2. It provides a Dynamic array for storing the element.
3 It is an array but there is no size limit
4. We can add or remove elements easily.
5. It is More flexible than a Traditional Array.

Ways to reading elements from any list

1. for loop
2. For...Each loop
3. Iterator (line by line)

Note
values will be iterated in using hasNext()

Program

```
import java.util.*;
class Main
{
public static void main(String[] args)
{
ArrayList al = new ArrayList();
al.add("ak");
al.add("sk");
al.add("lk");
al.add("jk");
al.add("mk");
Iterator i = al.iterator();
while (i.hasNext())
{
system.out.println(i.next());
}
}
}
```

O/P:1
ak
sk
lk
jk
mk.

O/P2:
7
7.0
lk
a
7.53
It is accept all datatype.

** ArrayList can accept all the Datatypes

Linked List: Linked List is a sequence of links which contains items. Each link contains a connection to another link.

Syntax
Linkedlist object = new Linkedlist();

Note className and linkedlist Doesn't same other error.

```
import java.util.*;
class LinkedlistExample
{
public static void main(String args[])
{
LinkedList<String> al = new LinkedList<String>(); //creating linked list
al.add("ak");
al.add("sk");
al.add("gk");
al.add("bk");
Iterator<String> i = al.iterator();
while (i.hasNext())
{
system.out.println(i.next());
}
}
}
```

O/P:
ak
sk
gk
bk

** Genrtic inside using is collection like Datatype

Ex: LinkedList<String>
Generic

Ex:3

```
import java.util.*;
public class LinkedList
{
public static void main(String[] args)
LinkedList l= new LinkedList<Integer>();
l.add(5);
l.add(7);
l.add(2,6);
l.add(0,10);
l.remove();
for(int k: l)
{
System.out.Println(k);
}}
}
```

Inter → l.add(7)

→ It will remove first element

O/P:

5
7
6
7

"10 will remove because it is first"

Note

LinkedList is allow the duplications.

Ex:3

```
import java.util.*;
public class LinkedList{
{
Public static void main(String[] args)
{
LinkedList<Integer> al= new LinkedList<Integer>();
al.add(5);
al.add(7);
al.add(2,6);
al.add(0,10);
al.add(7);
al.remove();
for(int k=0; k< al.size(); k++)
{
System.out.Println(al.get(k));
}}
}
```

O/P: 5
7
6

4)
```
import java.util.*;
public class Example
{
LinkedList<Integer> list = new LinkedList<Integer>();
list.add(123);
list.add(12);
list.add(11);
list.add(1134);
system.out.println(list);
object[] a = list.toArray();
System.out.print("After converted");
for(object element: a)
{
system.out.println(element+" ");
}}
}.
```

O/P: [123, 12, 11, 1134]
After converted 123 12 11 1134.

Note
object Array is used to store the multiple value in a single variable.

Apply to all data types.
- list.removeFirst() → It will remove first element
- list.removeLast() → It will remove last element
- list.addFirst() → add first element
- list.addLast() → add last element

ArrayList	LinkedList
1. ArrayList is slow	1. LinkedList is fast.
2. ArrayList implemented by only list	2. LinkedList implement list acts as as a queue as well.
3. ArrayList is faster in storing and accessing data	3. LinkedList is faster in manipulation.
4. ArrayList internally uses a dynamic array to store its element.	4. LinkedList uses doubly list. underlying the data structure.

Set:

Java collections sets refers to collection that cannot contains duplication. It is a interface.

Note: 1 HashSet is a unordered.

We can't use forloop because get() is cannot be used for Interface.

Program

```
import java.util.*;
public class Hashset
{
HashSet<Integer> al = new HashSet<Integer>();
al.add(5);
al.add(7);
al.add(10);
al.add(7);
al.add(8);
for(int k : al)
{
system.out.println(k);
}
}
}
```

O/P:

```
5
7
8
10
```

Treeset:

Treeset is one of the most important implementations of the sorted set interface in java that uses a tree of storage

Set <--extends-- sorted set <--Extends-- Navigableset <--Implements-- Treeset

It is follows the ordered.

Ex:
```
import java.util.*;
public class Treeset
{
public static void main(String[] args)
{
TreeSet<Integer> al= new TreeSet<Integer>();
al.add(5);
al.add(7);
al.add(7);
al.add(11);
al.add(13);
al.remove(11);
al.add(8);
for(int k:al)
{
System.out.println(k);
}
}
}
```

O/P:
5
7
8
13

Hashcode()

The Hashcode() Method must return the same value for a particular object every time, given that we are not changing the

the Hashcode() can return the same value for two objects that are not equal according to the equals() method.

Each value can have the hashvalue.

Ex: a=1, b=2, ... z=26.

Ex:
```
import.java.util.*;
class Hashcode
{
public static void main(String[] args)
{
TreeSet<String> set = new Tree set<String>();
set.add("abc");
set.add("bcd");
set.add("acd");
for (String k: al)
{
System.out.println(k);
}
}
}
```

O/P: Because
abc // 1+2+3=6
acd // 1+3+4=8
bcd // 2+3+4=9.

5, Example

```
import java.util.*;
Public class HashSet
{
Public static void main(String[] args)
{
HashSet <String> hs = new HashSet<String>();
hs.add("A");
hs.add("B");
hs.add("C");
hs.add("d");
hs.remove("d");
for (String k : hs)
{
system.out.println(k);
}
}
}
```

O/P: A
B
C

Map Interface

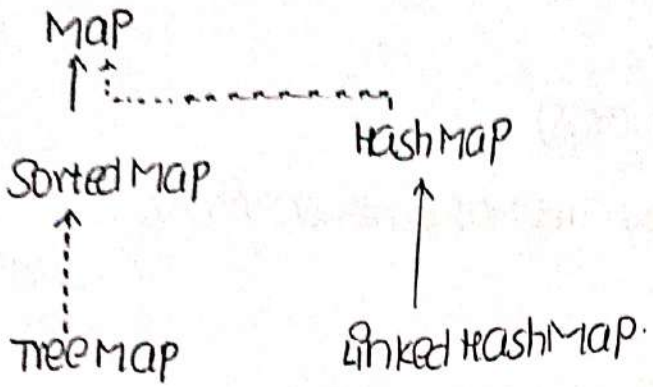
Map contain value on the basis of key, ie key and value pair
·Each key and value pair is known entry. A Map contains unique key.

Java Map Hierarchy:

There are two interface for implementing Map in Java Map are sorted Map, and three classes HashMap, LinkedHashMap and TreeMap.

1. key are unique doesn't contain Duplication
2. values contains the Duplication.
3. use put() function to add elements.
4. values getting from Map using getkey(). getvalue() fn.

Ex:



5. A Map can't be traversed, so you need to convert in to set using setkey() and getkey().

Ex:1

```
import java.util.*;
public class Main
{
public static void main(String[] args)
{
Map<Integer,String> hm = new HashMap<Integer,String>();
hm.put(1, "ajay");
hm.put(2, "sk");
hm.put(3, "jk");
hm.put(4, "gk");
hm.put(5, "bk");
for(Map.Entry m : hm.entryset())
{
System.out.println(m.getvalue());
}
}
}
```

O/P:

ajay
sk
jk
gk
bk.

TreeHash It is used to sorted ordered.

Ex:
```
import java.util.*;
 Public class Main
 {
 Public static void main(String[] args)
 {
 TreeMap<Integer,String> tm= new TreeMap<Integer,String>();
 tm.put(1, "ajay");
 tm.put(6, "ak");
 tm.put(2, "jk");
 tm.put(4, "gk");
 tm.put(3, "bk");
 for(Map.Entry m:hm.entryset())
 {
 System.out.println(m.getKey());
}
}
}
```

O/P
1
2
3
4
6

Java Hashtable

1. Hashtable is a synthorized
2. Hashtable class contains unique elements
3. Hashtable class Doesn't allow null key or value.
4. The initial defaut capacity of Hashtable class is 11 whereas loadfactor is 0.75

```
import java.util.*
Public class Main
{
public static void main(String[] args)
{
Hashtable<Integer,String> hm = new Hashtable<Integer,String>();
 hm.put(1, "ajay");
 hm.put(6, "sk");
 hm.put(2, "jk");
 hm.put(4, "gk");
 hm.put(3, "bk");
 for(Map.Entry m: hm.entryset())
 {system.out.println(m.getkey());
}
```

O/P:

6
4
3
2
1

Ex:2,

```
import java.util.*;
public class Main
{
public static void main(String[] args)
{
Hashtable<Integer, String> ht= new Hashtable<Integer, String>();
ht.put(7, "ajay");
ht.put(6, "sk");
ht.put(2, "jk");
ht.put(4, "gk");
ht.put(3, "bk");
ht.remove(3); (or) ht.remove(3, "bk");
ht.putIfAbsent(1, "sk");
for(Map.Entry m: hm.entrySet())
{
System.out.println(m.getKey() + " " + m.getValue());
}
}
}
```

O/P:

7 ajay
6 sk
4 gk
2 jk
1 sk.

Unique Duplication are allowed.

Ex:4,
```
import java.util.*;
public class Main
{
public static void main(String[] args)
{
Hashtable<Integer, String> ht = new Hashtable<Integer, String>();
ht.put(7, "ajay");
ht.put(6, "sk");
ht.put(2, "jk");
ht.put(4, "gk");
ht.put(3, "bk");
ht.PutIfAbsent(1, "sk");
System.out.println(hm);
}}
```

O/P:

{7=ajay, 6=sk, 4=gk, 2=jk, 1=sk}

Array List	Linked List
1. It interally working by dynamic Array	1. It is interally working by doubly linked list
2. insertion order is follow	2. It is follow insertion order
3. allows duplicates.	3. It is allow duplicates
4. Manipulation is slow	4. Manipulation is fast.
5. storing data, and Accessing data is fast.	5. storing data and accessing data is slow.

*** Hash table

Hash table is a Thread safe as compare to Hash map

List

1. It will allow Duplicates
2. It will follow insertion order.

Set

1. It will not allow Duplicates
2. It will not follow insertion order.

Treeset

1. It's maintaining ascending order.

Map

1. It's store element key and value pair.
2. It's is allow Duplicates in key and allow Duplicates value.
3. We cant implement traverse (iterator).
4. Map is using entryset() function to setvalues.
5. to get the value getvalue()
   to get the key getkey()

List and set

1. either single value or single key we can use
2. using iterator we can traverse elements
   Using hasNext(), next to print values.

HashMap

1. Hash map contain values Based on key
2. Hash map contain unique key
3. Hash map may have one null key and multiple null values.
4. Hash map is non synchronize
   not thread safe.
5. It maintain no order.
6. The initial default capacity of java Hashmap class is 16 and load factor is 0.75

Hashtable

Hashtable is an array of a list. Each list is known as a bucket.

The position of bucket is identified by calling the hashcode() method.

Hashtable contains unique elements.

Hashtable doesn't allow null key or value

Hashtable class is synchronized

The initial default capacity of hashtable class is 11 whereas loadfactor is 0.75.

Data Structures in Java

"They are 2 types of DS"
1. Linear
2. non Linear

Linked List DS
i, Single Linked List
ii, doubly Linked List
iii, Circular Linked List

2. Stack DS
3. Queue DS
4. Tree DS
5. Graph DS
6. Heap DS
7. Searching
8. Linear search
9. binary search
10. Hashing

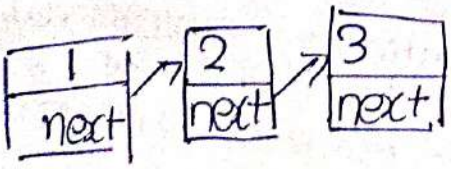
Any 5 Sorting Techniques.

"Linkedlist is mainly" used for.

1. variable size
2. Non-contiguous Memory
3. Insertion is O(1)
4. search in O(n).

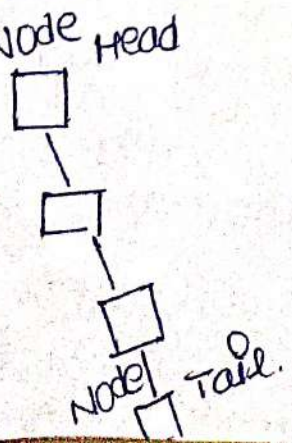
Node

Ex:

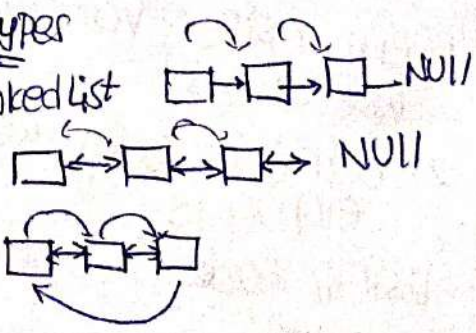


next is used for Reference

Ex:2



They are 3 types
1. Single Linked List
2. Double
3. Circle



throws:

```
import java.io.*;
class Main
{
static void method() throws IOException
{
throw new IOException("device error");
}
public static void main(String args[])
{
try
{
method();
}
catch(Exception e)
{
system.out.println("ok");
system.out.println("normal form...");
}
}
}
```

O/P:
ok
normal form.

MCQ

1. Which f^n among H
2. In a class, encapsulating an object of another class is
   A: Composition
3. The last node of linkedlist.
   A: Head → starting node.
   Tail → last node
   node ⇒ Data + Address.
4. Linear Datastructures
   Element are Arrange in orderwise. Ex: 1. Array 2. linked list 3. stack 4. queue.
5. Non linear DS
   It can't follows Ascending order & Descending order.
   Ex: 1. Tree 2. Graph 3. Heap. 4. Tries.

Q) Purpose of doing DS

A: Manipulation of Data

Ex: 1. Searching
2. Sorting
3. Insertion
4. Updation
5. Deletion

1. Write a program to Read the file by using InputStream ?

```
import java.io.*
Public class Main
{
Public static void main(String[] args)
{
String d = "Welcome to merifm";
try
{
FileWriter FW = new FileWriter("ak.txt");
fw.write(d);
System.out.Println("copy");
fw.close();
}
Catch (Exception e)
{
System.out.Println("not copy");
}
try
{
FileInputStream f = new FileInputStream("ak.txt");
int i = f.read();
while(i != -1)
{
system.out.print((char) i);
i = f.read();
}
f.close();
}
catch(Exception e)
{
system.out.Println("Error");
}}
}
```

O/P: copy
Welcome to merifm.

Array

Array is a container object hold fixed number of value in single type

ArrayList:

i, the elements are stored in dynamic away

Ex: | 1 | 3 | 4 | 5 | 6 | 7 | ...

If you want to insert the elements in Middle "2"

arr = | 1 | 2 | 3 | 4 | 5 | 6 | 7

* Insert time complexity: O(n)

ii, If we access the element in arr

arr[0] = 1

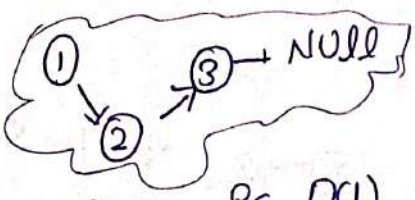
search complexity: O(1).

LinkedList:

① ③

If you want to insert "2" in middle

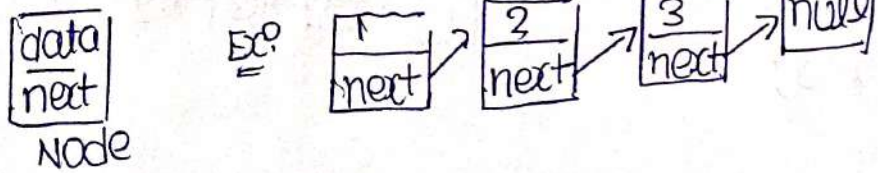
Ex:



It is a Non-continguous Memory.

Insertion T.C is O(1)

search T.C is O(n).

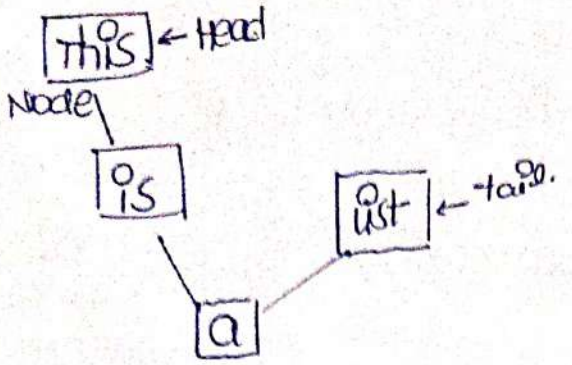


next is a acts as reference

Single LinkedList

1. In a singleLinkedList Node has Data and pointer to nextnode
2. It does not have pointer to previous node
3. Last node's next pointer to null.

Ex:1,



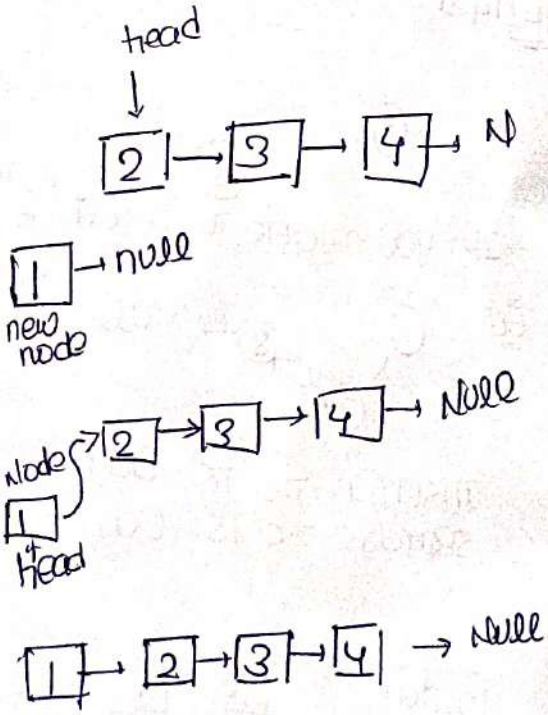
Program

```
class LL
{
class Node
{
String data;
Node next;
contructor → Node(String data)
{
this.data=data;
this.next = null;
} sizeH;
}

// add-first,last
public void addFirst(String data)
{
Node newNode = new Node(data);
if (head == null)
{
head = new Node;
return;
}
newNode.next = head;
head= new Node
}
```

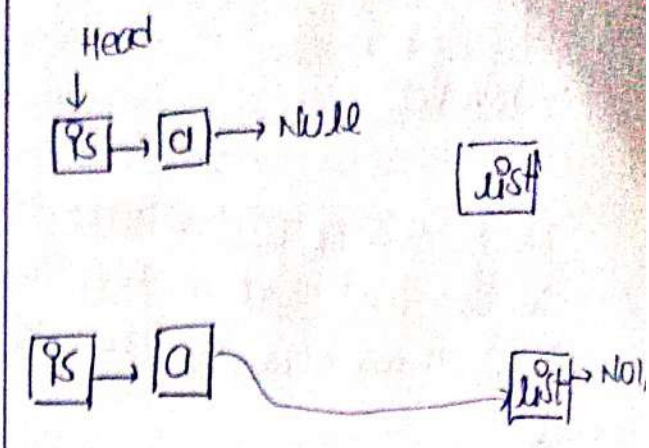
Node head;
private int size;
LL () → contructor.
{
this.size=0;
}

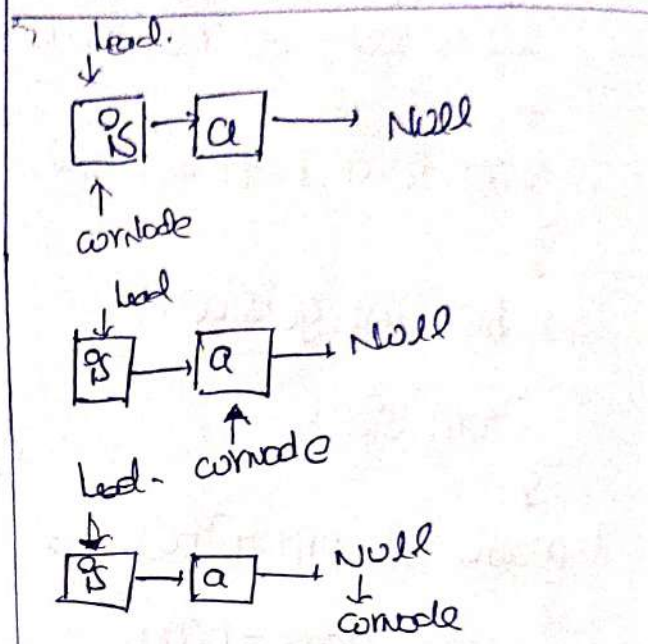
List is not exist head is NULL

list is exited head is not NULL

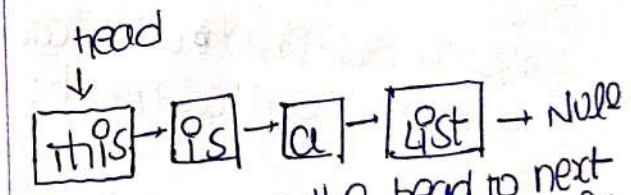


```
public void addLast (String data)
{
  Node newNode = new Node (data);
  if (head == null)
  {
   head = newNode;
   return;
  }

  Node curNode = head;
  while (curNode.next != null)
  {
   curNode = curNode.next;
  }
   curNode.next = newNode;
}
```

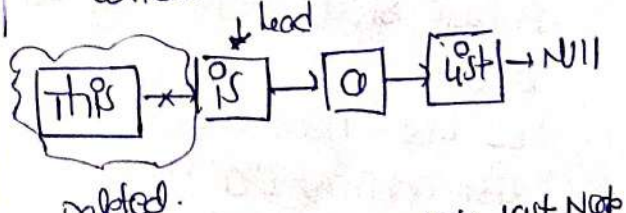




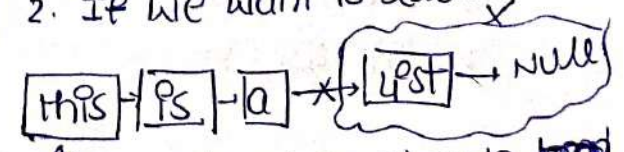

```
// Delete-first, Last.

public void deleteFirst()
{
  if (head==null)
  {
  system.out.println("list is Empty");
  return;
  }
  size--;
  head = head.next;
}
public void deleteLast()
{
  if (head == null)
  {
  system.out.println("list is Empty");
  return;
  }
  size--;
```



If we move the head to next node the previous data will be removed by Garbage collector.



2. If we want to Delete last Node



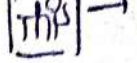
put last second node. next == NULL

```
if(head.next == null)
{
 head = null;
 return;
}
Node secondLast = head;
Node lastNode = head.next;
while(lastNode.next != null)
{
 lastNode = lastNode.next;
 secondLast = secondLast.next;
}
secondLast.next = null;
}
public int getSize()
{
 return size;
}
public void printList()
{
Node curNode = head;
while (curNode.next != null)
{
System.out.print(curNode.data + "->");
 curNode = curNode.next;
}
System.out.println("NULL");
}.
public static void main(String.args[])
{
 LL list = new LL();
 list.addFirst("a");
 list.addFirst("is");
 list.printList();
```

"edge case"

```
if(head == null)
{
System.out.println("list is Empty");
}
```

head



→ //head.next = null → lastNode = null

null.next ⇒ Error occur.

→ "Debug"

Note

Null Doesn't count by size.

```
public boolean search
 (String key)
 {
 Node currentNode = head
while(currentNode != null)
{
if(currentNode.data == key)
 {
 return true;
 }
current Node = currentNode.next
}
return false;
 }
}
```

```
list.addLast("list");
list.printList();
list.addFirst("this");
list.printList();
list.deleteFirst();
list.printList();
list.deleteLast();
list.printList();

system.out.println(list.getSize());
}
```

O/P:

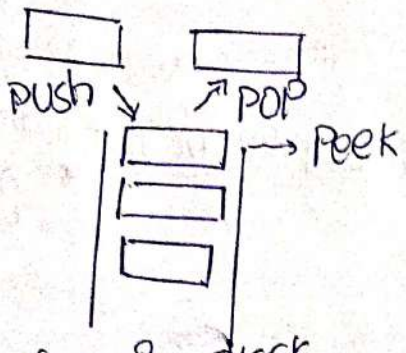
is → a → NULL
is → a → list → NULL
this → is → a → list → NULL
is → a → NULL

2.
is

```
string searchkey = "is";
if(list.search(searchkey))
{
system.out.println(searchkey);
}
else
{
system.out.println("not found");
}
}
```

Stack

A stack is a linear Data structures that follows the principle (Last-in-first-out) LIFO



Operations in stack

Push(): add the element → O(1)
Pop(): remove the element → O(1)
Peek(): TOP of the element → O(1)
isEmpty(): It is used to check whether stack is empty or not
peek(): It returns the element at the given position
isFull(): It is used to check whether the stack is full or not.

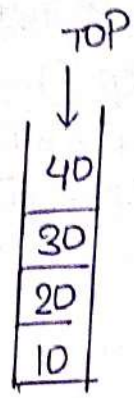
Example

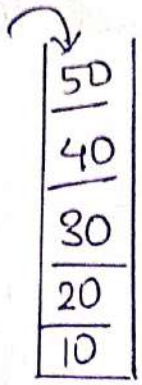
```
public class stack
{
static int max=100;
int top;
int ar y= new int[max];

boolean isEmpty()
{
return (top<0);
}
stack()
{
top=-1;
}
boolean push(int x)
{
if (top>(max))
{
system.out.println("overflow");
return false;
}
```

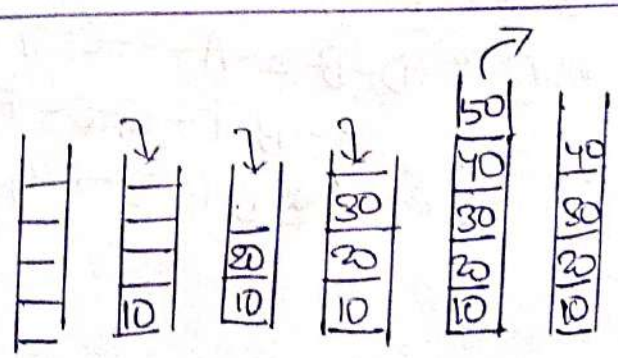
```
Eg:1,
class stack
{
int maxsize;
int top
int[] stackarray;
public stack (int size)
{
maxsize = size;
top = -1;
stackarray= new int[maxsize];
}
public void push(int value)
{
if (isFull())
{
system.out.println("overflow");
return;
}
stackarray[++top]=value;
}
public int pop()
{
if(isEmpty())
{
system.out.println("underflow");
return-1;
}
return stackarray[top--];
}
public boolean isEmpty()
{
return (top==-1)
}
public boolean isFull()
{
return (top==maxsize-1);
}
public int peek()
{
if(isEmpty())
{
system.out.println("underflow");
return-1;
}
return stackarray[top];
}
public static void main(String[] ar(s)
{
Stack s = new Stack(5);
system.out.println( s.isEmpty());
s.push(10)
s.push(20)
s.push(30)
s.push(40)
s.push(50)
s.pop();
s.pop();
system.out.println(s.isEmpty());
system.out.println( s.isFull());
system.out.println(s.peek());
}
```

O/P: true
false
false
40.

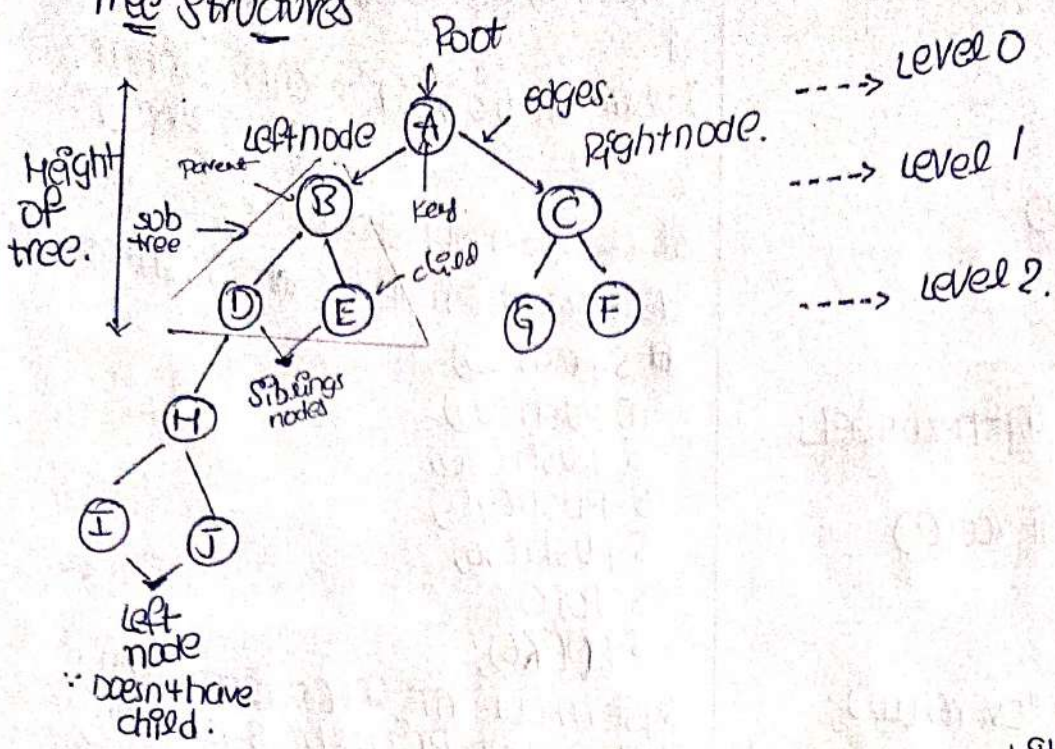
Ex:







Tree Structures

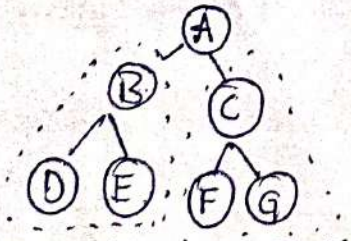


Types

1. Binary Tree (Maximum 2 child (or) 1 child (or) 0 child)
2. BST (Binary Search Tree) Ex: Left child is less than Root
   Right child is more than Root (or) equal
3. Available Tree → Ex: Left child and Right child ht is same.
   B = L - R
4. Red Black Tree

Ex: Tree Traversal (Binary, BST, Avl)

1. Inorder (Left - root - right)
2. Pre-order (root - left - right)
3. Post-order (Left - Right - root)



Inorder: D-B-E-A-F-C-G

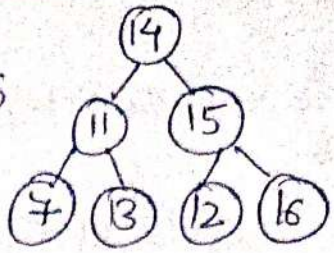
Pre-order: A-B-D-E-C-F-G

Post-order: D-B-E-F-G-C-A

In-order: 7-11-13-14-12-15-16

Pre-order: 14-11-13-7-12-15-12-16

Post-order: 7-11-13-12-15-16-14

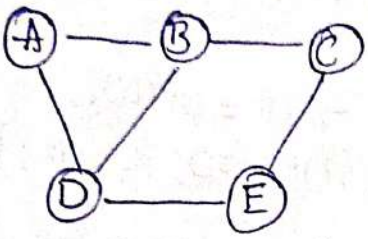


Graph:

A Graph can be defined as ordered set G(V, E) where V(G) represents the set of verticles of E(G) represents the set of edges which are used to connect these verticles.

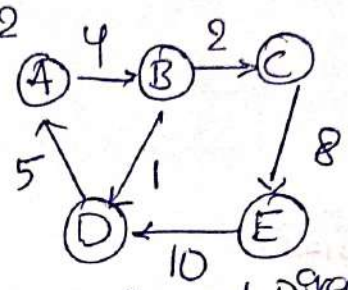
A Graph G(V,E) with 5 vertices (A, B, C, D, E) and six edges (A, B), (B, C) (C, E) (E, D) (D, B) (D, A) is shown in the following figures.

Ex:



Un Directed Graph

Ex:2

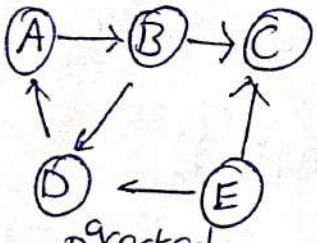


$$\begin{array}{c|ccccc} & A & B & C & D & E \\ \hline A & 0 & 4 & 0 & 0 & 0 \\ B & 0 & 0 & 2 & 1 & 0 \\ C & 0 & 0 & 0 & 0 & 8 \\ D & 5 & 0 & 0 & 0 & 0 \\ E & 0 & 0 & 0 & 10 & 0 \end{array}$$

Weighted Directed Graph.

Directed Graph:

In Directed Graph, edges from an ordered pair, Edges.



Directed Graph.

$$\begin{array}{c|ccccc} & A & B & C & D & E \\ \hline A & 0 & 1 & 0 & 0 & 0 \\ B & 0 & 0 & 1 & 1 & 0 \\ C & 0 & 0 & 0 & 0 & 1 \\ D & 1 & 0 & 0 & 0 & 0 \\ E & 0 & 0 & 0 & 1 & 0 \end{array}$$

Adjacent Matrix

queue

1. queue is follows FIFO

Operations

1. Enqueue : add the element
2. dequeue Remove the element
3. isEmpty : check whether the queue is Empty or not
4. isFull : It is used to check whether queue is Full or not
5. Peek : It is return the top element

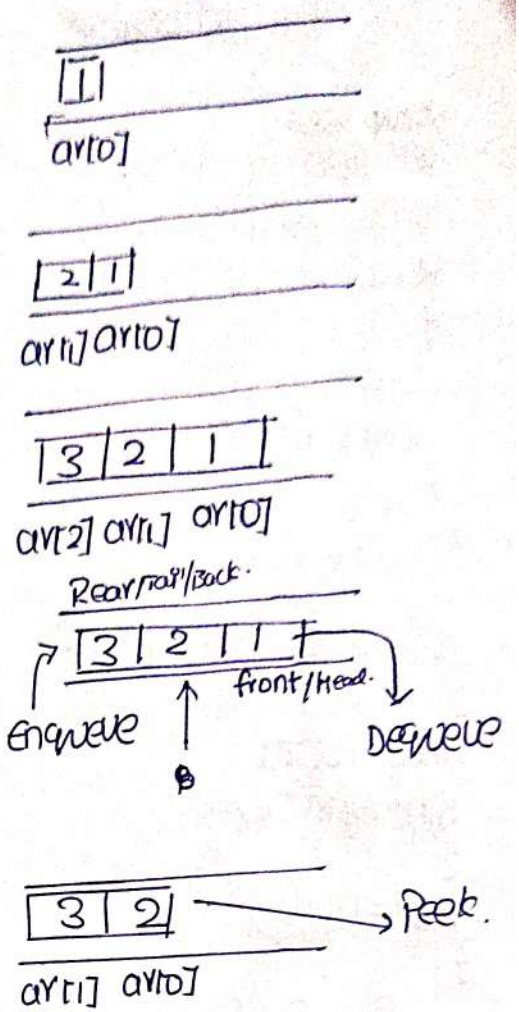
Program

```
Public class Queue3
{
static class queue
{
int arr[];
int size;
int rear;
Queue(int size)
{
this.size= size;
arr= new int[size];
rear = -1;
}
public boolean isEmpty()
{
return rear ==-1;
}
public boolean isFull()
{
return rear==size-1;
}
public void Enqueue (int data)
{
if(isFull())
{
system.out.println ("overflow");
return;
}
arr[++rear]=data;
}
public void dequeue()
{
if(isEmpty())
{
system.out.println("underflow
return -1
}
int front = arr[0];
for(int i=0; i<rear; i++)
{
arr[i]= arr[i+1];
}
rear--;
return front;
}
public void peek()
{
if(isEmpty())
{
system.out.println("underflow");
return -1;
}
return arr[0];
}
}
```

```
public static void main(String args[])
{
Queue q = new Queue(5);
q.enque(1);
q.enque(2);
q.Enque(3);
System.out.Println(q.dequeue());
System.out.Println(q.Peek());
}
}
```

O/P:

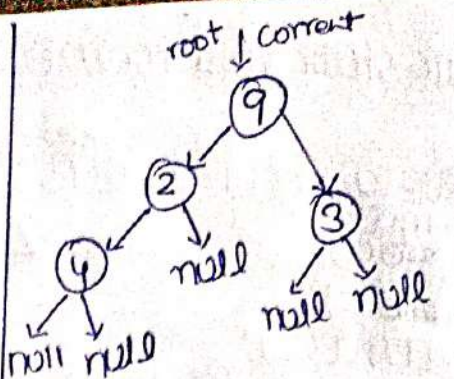
1
2

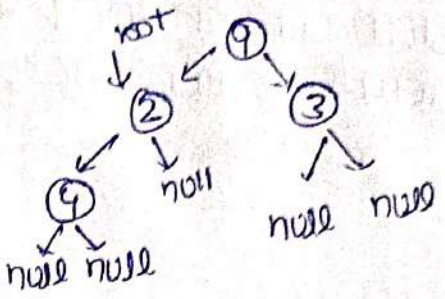


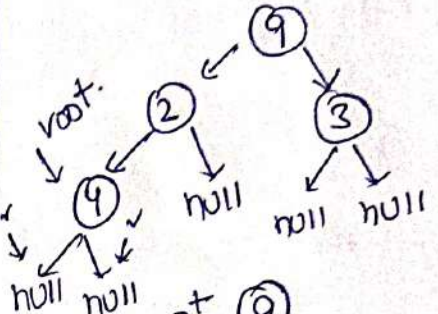
Tree traversal?

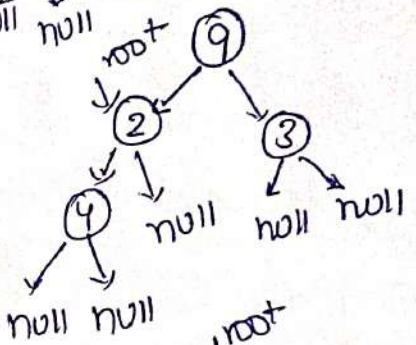
Example

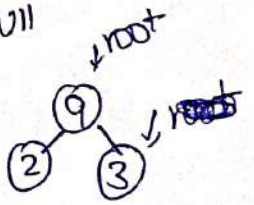
```
Public class Node
{
  int item;
  Node Left, Right;
  Public Node (int key)
  {
    item=key;
    left = right=null;
  }
}

Public class Binary tree
{
Node root;
Binary tree()
{
root=null;
}
1. Public void inorder (Node node)
2.  {
2.   if(node==null)      // Base case
3.   {
3.    return;
4.   }
5.   inorder(node.left);
6.   System.out.Println(node.item+ "->");
7.   in order(node.Right);
8.  }
```

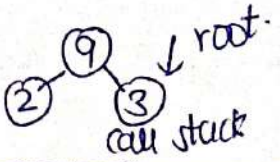










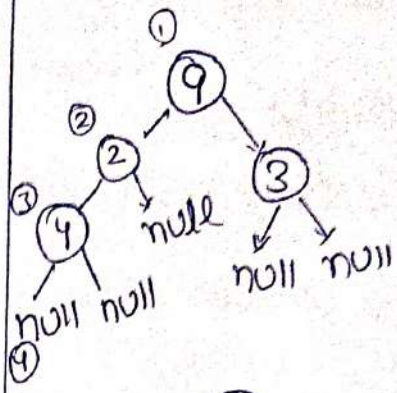


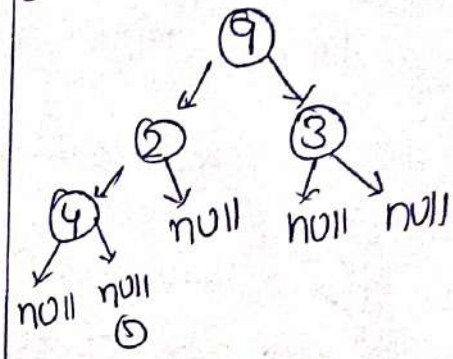
call stack

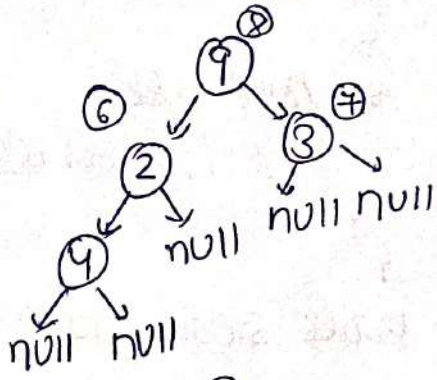
	Method	line number	root
	Inorder	5	9
	Inorder	5	2
	Inorder	~~5~~ 7	4
×	Inorder	Basecase.	null
×	Inorder		null
	Inorder	7	9
	Inorder		3

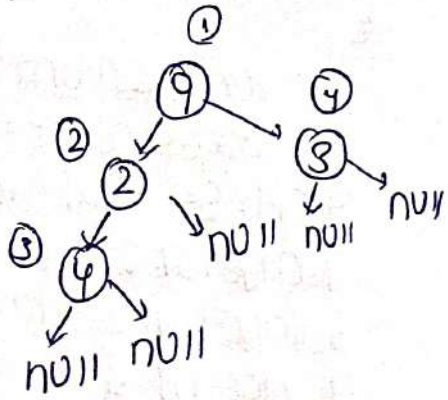
```
public void Post order(Node node)
{
 if(node==null)
 {
 return;
 }
 Postorder(node.left);
 Post order(node.right);
 system.out.println(node.item+"=>");
}.


public void Pre order(Node node)
 {
  if(node==null)
  {
  return;
  }
  system.out.print(node.item+"=>");
   preorder(node.left);
   Pre order(node.right);
 }
public static void main(string args[])
 {
  Binary tree  b= new Binary tree();
  bt.root = new Node(9);
  bt.root.left = new Node(2);
  bt.root.right= new Node(3);
  bt.root.left.left = new Node(4);
  bt.inorder(bt.root);
  bt.preorder(bt.root);
  bt.postorder(bt.root);
 }
}
```

O/P: 4-2 9 -3









O/P: 4-2-9-3
4-2-3-9
9-2-3-4

Graph

Program

```
Public class Graph
{
class Edge
{
int source, destination
}
int vertices, Edges
Edge[] edge;
Graph(int vertices, int edges)
{
this.vertices = vertices
this.edges = edges
edge = new Edge[edges];
for(int i=0; i<edge; i++)
{
edge[i] = new Edge();
}
}
public static void main(string args[])
{
int verticescount = 5;
int edges_count = 4.
Graph g = new Graph(verticescount, edgecount);
g.edge[0].source = 1
g.edge[0].destination = 2
g.edge[1].source = 2;
g.edge[1].destination = 3;
g.edge[2].source = 3;
g.edge[2].destination = 4;
g.edge[3].source = 4;
g.edge[3].destination = 5;
for(int i=0; i< edgecount; i++)
{
System.out.println(g.edge[i].source+"->"g.edge[i].destination);
}}
```

Stack (by using ArrayList)

```
import java.util.*;
public class StackAL
{
Static class Stack
{
ArrayList<Integer> list = new ArrayList<Integer>();
public void push(int data)
{
   list.add(data);
}
public boolean isEmpty()
{
return list.size()==0;
}
public void pop()
{
if (isEmpty())
{
return;
}
int top = list.remove(list.size()-1);
}
public int peek()
{
if(isEmpty())
{
return -1;
}
return list.get(list.size()-1);
}
}
```

```
public static void main(String args[]
{
Stack s=new Stack();
s.push(1);
s.push(2);
s.push(3);
s.push(4);
s.pop();
s.pop();
while(!s.isEmpty())
{
System.out.Println(s.peek());
s.pop();
}
}
}
```

O/P: 2
1

1, Explain about Java buzzwords?

A: Java is the most popular object-oriented programming language Java has many advance features.

2, A list of key feature is known as Java Buzzwords.

3, The Java team has listed the following terms as Java buzzwords.

1. simple
2. secure
3. Portable
4. object-oriented
5. Robust
6. Architecture neutral (or) platform independent
7. Multi-threaded
8. Interpreted
9. High performance
10. Distributed
11. Dynamic.

simple

Java programming language is very simple and easy to learn. understand and code. Most of the syntax in Java follow by C language. and objected-oriented programming concept are similar to C++.

Many complicated feature like pointer, operator, overloading, structure, union etc. has been removed. one of the most useful feature is the Garbage collector it make Java more simple.

secure

Java is said to be more secure programming language because it does not have pointers concept, Java provides a features "applet" which can embedded into a web application. The applet in Java does not allow to access to other parts of computer. It keep away from harmful programs like viruses and unauthorized access.

3. Portable:

Portability is one of the core feature of java which enable the java program to run on any computer or operating system.

Ex: An applet developed using java runs on a wide variety of CPU's operating systems and browser connected to Internet.

4. Object-oriented:

Java is said to be a pure object-oriented Programming language. In java, everything is an object. It support all the feature of the object-oriented programming paradigm.

Primitive data type java also implemented as object using wrapper classes, but still, it allows primitive data types to archieve high performance.

5. Robust:

Java is more robust because the java code can be executed on a variety environments. Java has a strong memory management mechanism.

Java is a strictly typed language, it has a strong set of Exception handling mechanism and many more.

6. Architecture-neutral (or) Platform Independent:

Java has invented to archieve "Write once; run anywhere, any time, forever". The java provide JVM to archieve architectural-neutral or platform-independent. The JVM allows the java program created using operating system can be executed on any other operating system.

7. Multi-threaded:

Java support Multi-threading programming which allows us to write program that do multiple operations simultaneously

8. Interpreted:

Java enables the creation of cross-platform programs by compiling into an intermediate representation called bytecode. The bytecode is interpreted to any machine code so that it runs on native machine

9. High Performance

Java Provides high performance with the help of features like JVM. interpretation and its simplicity.

10. Distributed:

Java programming language supports TCP/IP protocols. with enable the java to support the distributed environment of the Internet.

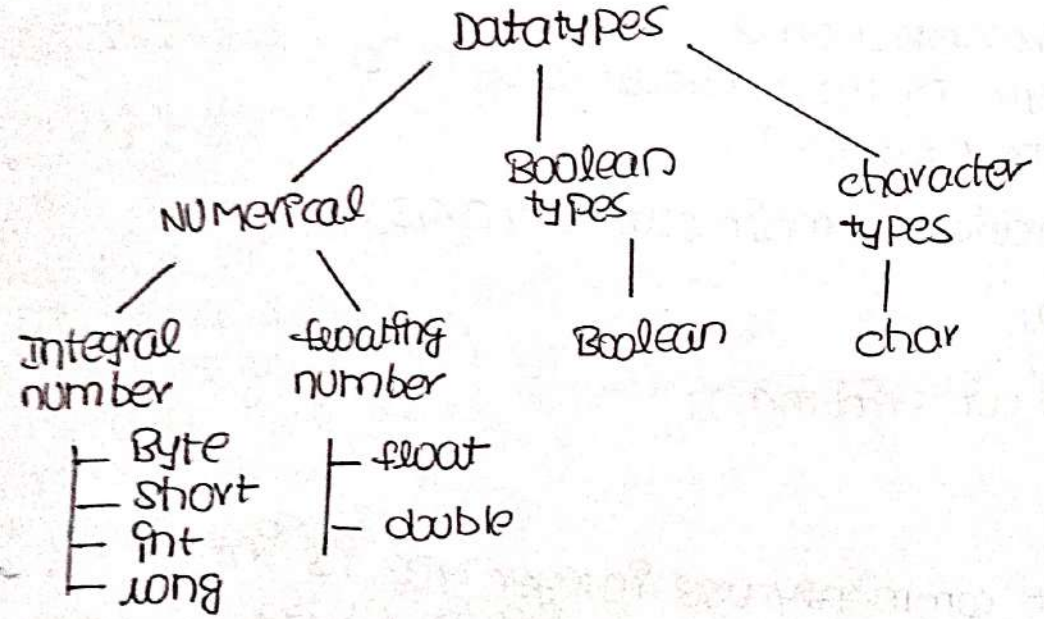
Java also supports Remote Method Invocation (RMI). this feature enable a program to invoke methods across network

11. Dynamic?

Java is said to be dynamic because the java byte code may be dynamically updated on a running system and it has a dynamic memory allocation and deallocation.

3, Explain about java data types?

Ans



Byte

1, the smallest integer type is byte.
2, the size of byte is 1 byte
3, this is a signed 8-bit type range is -2^7 to 2^7-1
4, Byte variable are declared with byte keyword.
5, the default type is int & Default value is 0

Ex: byte b, c;

Ex:
```
class SimpleExample
{
Public static void Main(String[] args)
{
 byte n;
 n=113
 System.out.Println(n);
}}
```

O/P:

113

Short:

1, Short is a signed 16-bit type & size is 2byte
2, Short is a range of -2^{15} to $2^{15}-1$
3, It is probably that least used Java type.
4, Declaration of short Datatype is

short variable_name.

5, default type is int & default value is "0"

Ex:
```
class SimpleExample
{
Public static void main(String[] args)
{
 short n;
 n=150;
 System.out.Println(n);
}
}
```

int:

1, The Most commonly use integer type is int.
2, It is signed 32-bit type & size is 4 byte
3, The range of int is -2^{31} to $2^{31}-1$
4, int is often the best choice when integer is needed
5. The default type is int & default value is "0"

Integral literals

Ex:
```
class SimpleExample
{
Public static void main(String[] args)
{
 int n;
 n=150;
 System.out.Println(n);
}
}
```

O/P: 150

long:

1. long is a signed 64-bit type & size is 8 byte.
2. the range of long is quite large.
3. the range of long is -2^{63} to $2^{63}-1$
4. Declaration of long data type

```
long variable_name;
```

5. default type is int & default value is "0"

Ex:

```
class SimpleExample
{
public static void main(String[] args)
{
long n=-12332252 6261;
system.out.println(n);
}
}
```

O/P:

-12332252 6261.

floating-point numbers:

float:

1. the size of float is 4 bytes.
2. It is a single precision.
3. It is 5 to 6 decimal number.
4. default type is double. & default value is 0.0.
5. range is -2^{31} to $2^{31}-1$

Ex:

```
class SimpleExample
{
public static void main(String[] args)
{float n = 19.85f;
system.out.println(n);
}
}
```

O/P: 19.98.

double:

1. The size of double is 8 byte
2. It is the double precision.
3. 14 to 15 decimal number.
4. default type is double & default value is 0.0
5. range is -2^{63} to $2^{63}-1$

Ex: class simpleExample
{
public static void main(string[] args)
{
double n= -42937737.9d;
System.out.println(n);
}
}

O/P:
-4.2937737 9E7.

Boolean

1. In Boolean data types there is no range.
2. Boolean data types only two value they are true or false
3. true or false should be lower case.
4. default value is false
5. Declaration of Boolean data types is Boolean variable_name=v

Ex: class simpleExample
{
public static void main(string[] args)
{
boolean b= false;
system.out.println(b);
}
}

O/P:
false.

character

1. The character data type can be represented by ' ' quotes.
2. The size of character is 2 byte.
3. The range is -2^{15} to $2^{15}-1$
4. default value is "0"

Ex: class simpleExample
{
public static void main(string[] args)
{
char ch= 'z';
System.out.println(ch);
}
}

O/P:
z

conversation

1 byte byte → short (2 byte) → int (4 byte) → long (8 byte) → float (4 byte) → double (8 byte)

char → int

3. Explain the types of variables?

A: Based on the type of value represented by a variable all variable are divided into 2 types. They are.

1. Primitive variable
2. Reference variable

2. Based on the behaviour position of declaration all variable are divided into 3 types.

1. Instance variable
2. static variable
3. local variable.

Instance variable:

1. If the value of variable is varied from object to object such type of variable are called instance variable
2. For every object a separate copy of instance variable will be created.
3. Instance variable will be created at the time of object creation and destroyed at the time of object destruction hence the scope of instance variable is exactly same as scope of object.
4. Instance variable will be stored on heap as part of object.
5. Instance variable should be declared with in the class directly but outside of any method or block or constructor.
6. Instance variable can be accessed directly from instance area. But cannot be accessed directly from static area.
7. But by using object reference we can access instance variable from static area.
8. Instance variable are also known as object level variable or attributes

Ex: class Test

```
{
boolean b;
Public static void main(String[] args)
{
Test t=new Test();
system.out.print-ln(t.b);
}}
```

O/P: false.

Static variable:

If the value of variable is not varied from object to object such type of variable is not recommended to declare as instance variable. We have to declare such type of variable as class level by using static modifier.

2. In this case of instance variable for every object a separate copy will be created but in the case of static variable for entire class only one copy will be created and shared by every object of the class.

3. Static variable will be created at the time of class loading and destroyed at the time of class unloading hence the scope of static variable is exactly same as the class file.

4. Static variable will be stored in method area. Static variable should be declared with class directly outside of any method or block or constructor.

5. static variable can be accessed from both instance and static area directly.

6. We can access the static variable either by class name or by object reference but usage of class name is recommended.

7. But within the same class it is not required to use class name we can access directly.

Ex: class Test

```
{
Static int i=10;
Public static void main(String[] args)
{
Test t=new Test();
system.out.print-ln(t.i);
}}
```

O/P: 10

Local variable

1. Local variable are declared in methods, constructor or blocks.
2. Local variables are created when the method, constructor or block and the variable will be destroyed once it exists method, constructor or block
3. Access modifiers cannot be used for local variabe
4. Local variable are visible only within the declared method, constructor, or block.
5. Local variable are implemented at stack level internally.
6. There is no default value for local variable, so local variable should be declared and an initial value should be assigned before the first use.

Ex:
```
int area()
  {
  int l=10;
  int b=5
  int r= l*b;
  return r;
  }
```

O/P: 50.

4. Explain the Array Declaration, construction, & initialization?

Ans
1. Every array in java is an object
2. An array is a group of like-typed variable that referred to the common name.
3. Arrays of any type can be created and may have one or more dimensions.
4. Array is a group of similar statement.
5. there is no inbuilt data/function to perform any operation in java.
6. Array in fixed size
7. Array can store the homogeneous element.
8. It is an indexed collection of fixed number of language element

Array Declaration:

We can declare the array datatype arrayname[];

i, one-Dimensional array : int ar[];
ii, Two-Dimensional array: int ar[][];
iii, Three-Dimensional array: int ar[][][];

Array creation:

We can create an array

datatype arrayname[]= new datatype[size];

Ex: int ar[]= new int[10];
int ar[][]= new int[5][2]

In two Dimensional array we save the memory we use ragged array

Array Initialization:

We can initialize the array element directly

ar[1]=3
ar[2][3]=5

Program:

```
Class A
{
Public static void Main(String[] args)
{
int ar[]= new int[5];
a[0]=3;
ar[1]=2;
a[2]=5;
ar[3]=7;
ar[4]=11;
for(int i=0; i<a.length; i++);
System.out.println(ar[i]);
}
}
```

6) Explain Different types of operators in Java

A: Java language support the following types of operation.

They are

1. Arithmetic operator
2. Relation operator
3. Assignment operator
4. Logical operator
5. Bitwise operator
6. Unary operator
7. Ternary operator.

1. Arithmetic operator:

Arithmetic operator are used to perform the Mathematical operations like Addition, subtraction, Multiplication etc.

Ex: a+b, a-b, a*b, a/b, a%b.

Program

```
class Main
{
Public static void main(string[] args)
{
int a=12, b=3;
System.out.println("a+b= ",(a+b));
}
}
```

O/P:

a+b=15

2. Relation operator:

comparison operator are used to compare values. It return either true or false A/c to condition.

Ex: a<b; a==b, a!=b, a>b;

Program:

```
class Main
{
Public static void main(string[] args){
int a=7, b=11;
System.out. println(a!=b);
}
}
```

O/P:

Assignment operator:

Assignment operators are used in Java to assign the values to variable.

Ex: a=b; a+=b; a-=b; a*=b; a/=b; a%=b;

Problem:

```
class Main
{
Public static void main(String args[])
{
int a;
int var;
a=10;
var=a;
System.out.println("var using= :"+var);
}
}
```

O/P:

var using= 10.

Logical operator:

Logical operator are used to check whether an expression is true or false.

Ex $e_1 \&\& e_2$, $e_1 || e_2$

Program:

```
class A
{
Public static void main(String[] args)
{
System.out.println((5>3) && (8>3));
}}
```

O/P: True.

Bitwise operator:

Bitwise operator in Java are used to perform operation on individual bits.

Ex: <<, >>, ~, ^, &, >>> etc.

∴ Bitwise complement operation of 35

35 = 00100011 (Binary)

~ 00100011

11011100 = 200 (decimal)

Unary operators

Unary operator are used with only operand.
Ex: ++ is an unary operand that increase the value of a by 1. Ex: ++5 will return 6

Programe

```
class A
{
public static void main (String[] args)
{
int a=12
int result
result = ++a;
System.out.Println("After increment:" + result);
}
}
```

O/P:

After increment : 13

Ternary operator

Ternary operator is shorthand for if-then-else statement.

Ex: variable= Expression ? expression 1: expression 2.

Problem

```
class A
{
public static void main(String[] args)
{
int feb=29
String result;
result = (feb==28) ? "Not a leap year:" "leap year";
System.out.Println(result);
}
}
```

O/P: leap year.

7, Explain control statement indetail ?

A: control statement:
The statement which control the flow of execution is called control statement.

1. Selection statement (or) Decision Making statement (or) condition statement
2. Iteration (or) looping (or) repetition statement
3. Jumping (or) Branching statement.

1. Selection statement:

i, Simple if : It is used when they is only one statement.

```
class A
{
public static void main(String[] args)
{
int x=0
if (x>0)
{
System.out.println("positive");
}
}
```

O/P: Positive.

ii, if else: If else is used if they is two condition

```
class A
{
public static void main(String[] args)
{
int a,b;
if (a<b)
a=0;
else
b=0;
System.out.println(a);
System.out.println(b);
}
}
```

iii, if-else-if-ladder: if there is 300 or more condition it is used.

Ex: import java.util.*;
Class A
{
public static void main(String[] args)

```
}
Scanner Sc= new Scanner(System in);
int n=Sc.nextInt();
if(n>0)
System out Println("positive");
else if(n<0)
System out.Println("Negative");
else
System out.Println("zero");
}}
```

nested if: A nested if is an if statement that target of another if or else. Nested if mean inside the if statement

Ex: class A

```
{
Public static void main(String[] args)
{
int i= Sc.nextInt();
if (i==10)
{
if (j<20)
a=b;
if(k>100)
c=d;
else
a=c
}
else a=d;
}
}
```

Switch: When we have the no. of choice or option then we use switch statement

Ex: class A

```
{
public static void main(String[] args)
{
int x=2, y=3;
Switch (option)
{
case1: system.out.Println(x+"y);
    break;
case2: system.out.Println("x"-"y);
    break;
default: system.out.Println("Invalid option");
}}}
    break;
```

Iteration statement:

If the condition is true then enter into the

While loop

```
class A
{
public static void main(string[] args)
{
int x=7;
while (x>0)
{
System.out.println("positive integer");
}
}
}
```

For loop: loop will be repeated until the statement is false

```
class A
{
public static void main(string[] args)
{
for (int i=0;i<=10; i++)
{
if (i%2==0)
{
System.out.println(i);
}
}
}
```

Do-while It is used to iterate a part of program repeatedly until the specified condition is true.

```
class A
{
public static void main(string[] args)
{
int i=1,sum=0;
do
{
sum=sum+i;
i++;
}
while(i<10);
System.out.println(sum);
}
}
```

Jumping statement:

Break: Break statement is optional. If you want to Break the execution. then use the Break statement between them. It is used in switch case, while loop

Ex: class A

```
{
public static void main (string[] args)
{
for (int i=0; i<=10; i++)
  { if (i==7)
      break
    system.out.println(i);
  }
}
}
```

continue:

The continue keyword is used to end the iteration in the loop

```
class A
{
public static void main(string[] args)
{
for (int i=1; i<=10; i++)
{
 if (i==5)
   continue
system.out.println(i);
}
}
```

return: It is used for the explicit return the value.

Ex: return 0

```
class A
{ int display()
  {
  return 10
  }
  public static void main(string argr[])
   {
   A a= new A()
   system.out.println(a.display());
   }
}
```

10) Explain about the Garbage collector?

A: Garbage collector is a process of reclaiming the runtime unused memory automatically

2. In other words, it is a way to destroy the unused object.

3. In Java we don't have delete operation so we can use the Garbage collector.

4. It Make More efficiency.

5. It is automatically done by the Garbage collector, so we don't need to make extra efforts

6. the finalize() Method is invoked each time before the object is Garbage collected.

7. this method are used to perform cleanup processing.

Programe

```
Public class Test Garbage1
{
Public void finalize()
{
System.out.Println("object is garbage collected");
Public static void main(String args[])
{
  Test Garbage1 S1= new Test Garbage1();
  Test Garbage1 S2= new Test Garbage2();
  S1=null;
  S2=null;
  System.gc()
}
}
```

O/P:

object is Garbage collected
object is Garbage collected.

!! Explain about this keyword in java?

A: 1, this keyword can be used to refer current class of instance variable.

2, this can be used to invoke current class method
3, this can be used to invoke current class constructor.
4, this can be passed as an argument in method call.
5, this can be passed as argument to constructor call.
6, the most common use of this keyword is to eliminate the confusion between class attribute and parameter with same name.

Program

```
class Test
{
int a;
int b;
Test(int a, int b)
{
 this.a=a;
 this.b=b;
}
void display()
{
System.out.println("a= "+a+" b= "+b);
}
Public static void main(String[] args)
{
 Test object= new Test(10,20);
  object.display();
}
}
```

O/P
a=10 b=20

1) Explain Inheritance in Java?

A: 1, Inheritance in Java is a Mechanism in which object can acquire all the properties and behaviour of parent object

2, It is an important part of OOPS.

3, The idea behind inheritance in Java is that you can create new classes that are built upon the existing classes

4, Inheritance is used for Method overloading.

5, The extend class indicate that you making the new classes that Dervies from existing class.

6, The Meaning of extends "increase in functionally"

7, They are three types of inheritance

1. single Inheritance
2. multilevel Inheritance
3. hierarchical Inheritance

7, When a call inherits to another class, it is call single Inheritance.

8, When they is an chain inheritance, it is known as Multilevel inheritance.

9, Two or more class inherits in a single class it is known as hierarchial Inheritance

10, It is used for code Reusability.

Program:

```
class Animal
{
public void eat()
{
system.out.println("I can eat");
}
}
class Dog extend Animal
{
public void eat()
{
system.out.print("I eat dog food");
}}
```

```
Public void bark()
{
System.out.println("I can bark");
}
class Main
{
Public static void Main(String[] args)
{
Dog x= new Dog();
x.eat();
x.bark();
}
}
```

2, Explain Packages in Java?

A: 1, A package is a group of similar types of classes, interface and sub-package

2, It organize classes into a single unit.

3, In Java already many predefined packages are available.

Ex: Java.lang, Java.io, Java.util etc.

4, They are two types of packages are available in Java.

i, Built-in Package

∵ Package are already defined Java API

ii, Used defined Package

∵ We can create according to our need is called defined package.

5, The "import" key word is provide to access the other Package class and interface of current package.

6, The import key word is used in-built and user defined Package in Java.

7. We can create our own Package by creating own class and interface together.

8, The package name should be declared at the beginning of program.

9, Package provide code reasability. Before Package group of class

10, Access limit can be applied with in the Package.

Program:

```
Package P1
class Demo
{
Public void m1()
{
System.out.Println("method m1 ..");
}
```

S, Explain about Interface in Java?

A: 1. Interface is similar to class, but it contain only abstract Method.

2. By default variable are declared in interface public, static and final

3. Interface is a Mechanism to achieve full abstraction.

4. An Interface does not contain any constructor.

5. Syntax:

```
Interface Interface_Name
{
Public void Method1();
Public void method2();
<type> variable_name=value;
}
```

6. An interface extend another interface as it happen in class. It cannot Implement Interface.

Ex: Interface Base

```
{
Public void display();
}
Interface Derive extend Base()
{
Public void show();
}
```

7. The implement key word is used by class to implement an interface

8. A class can implement an interface After that class can perform the specific behaviour on an interface

8. Difference between Method overloading and Method overriding?

Method overloading	Method overriding.
1. Parameter Must be Different	1. Parameter Must be same
2. It is an example of compile time polymorphism	2. It is an example of runtime polymorphism.
3. return type May or May not be same	3. return type Must be same
4. It is also known as static binding or early binding.	4. It is also known as Dynamic Binding are late binding.
5. static variable can be overloaded.	5. the method Must be non-volatile or static Method for overriding.
6. It is used to increase readability of code	6. It is provide the specific implementation of Method that is already provide its Super class.

1. Difference between Multitasking and Multithreading.

A:

Multitasking	Multithreading
1. In Multitasking users are allowed to perform many tasks by CPU	1. While in Multithreading, Many threads are created from a process through which computer power is increased
2. Multitasking is slow compared to multithreading	2. While Multithreading is faster
3. In Multitasking termination process take more time	3. While in Multithreading termination of thread takes less time.
4. Component involves Multiprocessing	4. Component does not involve Multiprocessing
5. The processes share separate memory	5. The process allocated same memory.
6. Users are allowed to perform many tasks by CPU.	6. Many threads are created from a process through with computer power is increased
7. CPU is provided to execute many task at a time.	7. CPU is provided in order to execute many threads from process at a time.

2. Explain the types of Exception in Java with Examples.

A: Java defines several type of exceptions that relate to its various class libraries. Java also allows users to define their own exception

Java Exception

- Built-in- Exception
- User-defined Exception

Built-in-Exception:

Built-in-exceptions are the exceptions that are available in Java libraries

2. these exceptions are suitable to explain certain error situation
3. Below the list of important built-in-exception in Java.

they are

1. Arithmetic Exception
2. ArrayIndexOut of Bound Exception
3. ClassNotFound Exception
4. File Not found Exception
5. IO Exception
6. Interrupted Exception
7. NoSuchField Exception
8. No such Method Exception
9. NULL Pointer Exception
10. NumberFormat Exception
11. Runtime Exception
12. String IndexOutOfBound Exception
13. Illegal-Argument Exception
14. Illegal state Exception.

User-defined Exception:

Sometimes, the built-in-Exception in Java are not able to describe a certain situation.

2. In such cases, user can also create exception which are called 'user defined Exception'.

Ex: class MyException extends Exception.

3. We can write a default constructor in his own exception class

```
MyException(){}
```

4. We can also created a Parameterized constructor with a string as a parameter.

```
MyException (String str)
{
super(str);
}
```

first thread priority : 5
Second thread priority : 1
thread running.... thread-1

Q. What are different ways to create a thread

A: There are two ways to create a thread

1. By extending thread class
2. By implementing Runnable interface.

thread class:

thread class provide constructors and methods to create and perform operation on a thread. thread class extends object class and implements Runnable interface.

Commonly used constructors of thread class.

1. thread()
2. thread(string name)
3. thread(Runnable r)
4. thread(Runnable r, string name)

the start() method of thread class is used to start a newly created thread. It perform following tasks.

1. A new thread start
2. The thread moves New state to Runnable state
3. When the thread get a chance to execute, it's target run() method will run.

Program:

```
class Multi extends thread
{
public void run()
{
system.out.println("thread is
                running....");
}
public static void main(string[] args)
{
multi t1=new multi();
t1.start();
}
}
```

o/p: Thread is running

Program:2

```
class Multi implement Runnable
{
public void run()
{
system.out.println("thread is
                running.");
}
public static void main(string[] args
{
Multi m1= new multi();
Thread t1= new thread(m1);
t1. start();
}
}
}
```

o/p
thread is Running...

12, Explain about synchronization in java?

A: 1, Synchronization in java is the capability to control the access of multiple thread to shared access.

2, Synchronization is better option where we want to allow only one thread to access the shared resources.

3. synchronization is mainly two types.
   i, To prevent thread interface
   ii, To prevent consistency problem.

4. they are two types of synchronization.
   i, Process synchronization
   ii, thread synchronization

5. The main purpose of synchronization to avoid thread interface.

6. synchronization is a process of handling resource accessibility multiple thread request.

7. synchorization is a keyword in java to create a block of code referred to as critical section

8. Syntax

```
synchronized (obj identifier)
{
 //
}
```

9. they are two types of thread synchronization mutual exclusive and inter-thread communication.

   1. Mutual Exclusive
      i, synchronized Method
      ii, synchronized block
      iii, static synchronized.

   2. co-operation.

10. If you declare any method as synchronized, it is known as synchronized Method.

11, Synchronized Method is used to lock an object for any shared code.

12, When a thread invokes a synchronized method. It is automatically acquires the lock for that object and release it when the thread complete its task.

Programe

```
class Solution
{
synchronized public void display (string msg)
{
System.out.println("[" +msg);
try
{
thread.sleep (1000);
}
catch (InterruptedException e)
{
e.printstackTrace();
}
System.out.println("]");
}
}
class second extends thread
{
string msg;
first fobj;
second (first fp, string str)
{
fobj = fp;
msg = str;
start();
}
public void run()
{
fobj.display(msg);
}}
```

```
public class mythread
{
public static void main (string[] args)
{
first fnew = new first();
second ss = new second (fnew, "welcome");
second s1 = new second (fnew, "new");
second s2 = new second (fnew, "programmer");
}
}
```

o/p:

[welcome]

[programmer]

[new].

2) Explain about Inter-thread communication in java.

Ans Inter-thread communication (or) co-operation is all about allowing synchronized threads to communicate with each other.

2. Co-operation is a mechanism in which a thread is paused running in it's critical section and another thread is allowed sumecritical section to be executed

3. It is implemented by the following method of object class

1. wait()
2. notify()
3. notifyall()

4. the wait() method causes current thread to release the lock wait until either another thread invokes the notify() method or notify All() method for object.

5. The current thread must own this object monitor. So, it must called synchronized method only otherwise thrown exception.

6. i, public void wait() throws Interrupt Exception

7. ii, public final void wait(long time out) throws Interrupted Exception.

1. Explain in detail about Event, Event source, Event listeners

1. Event:
changing the state of an object is known as an event.
Ex: click on button
Dragging mouse etc.

2. The java.awt.event package provides many event classes and listener interfaces for event handling.

3. Event Source:
Events are generated from the event source

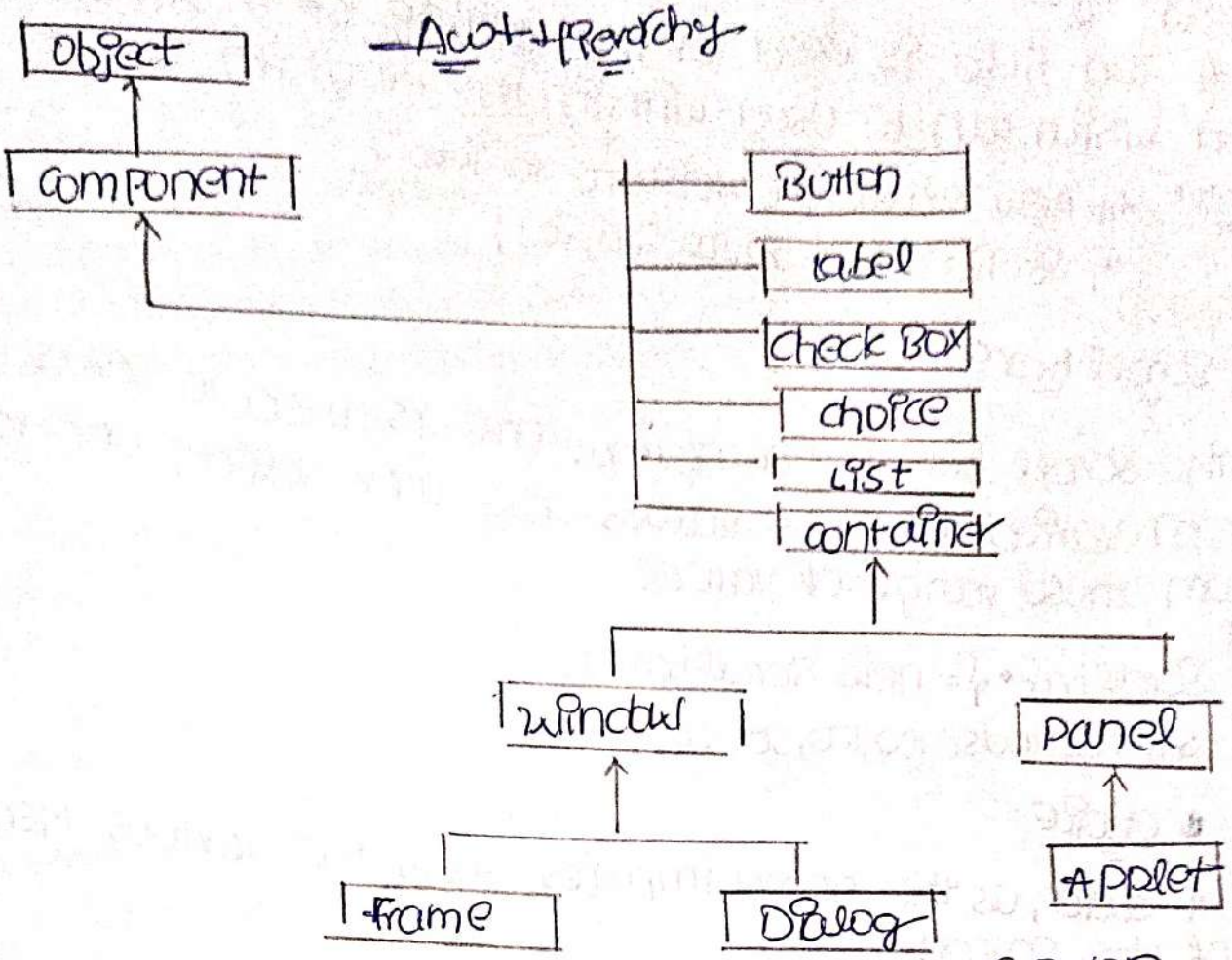
4. Event source are
1. Button
2. checked box
3. choice
4. list
5. menu item
6. Scroll bar
7. Text component
8. window etc.

5. An event listener in java is designed process some kind of event it "listens" for an event, such as a user's mouse click or key press, and then it responds according

6. An event listener must be connected to event obj that define event.

7. Different types of Event listener
ActionListener
Item Listener
Key Listener
Mouse Listener
Mouse Motion Listener
Window Listener
focus Listener
Caret Listener

Explain about Java AWT Hierarchy



Explain the following with example: i, Label ii, Button iii, Scroll bar iv, Textfield

1. Label:

A Label is user for placing text inside the container

2, A Label is used only for inputting text.

3, The Label does not imply that text can be altered or it can used as a button which can futher worked upon

Ex: Label n = new Label ("Name:", Label. CENTER);

2. Button:

1. The command generates a button in the user interface.

2. clicking on the button would move the command to another page or another web server which is used to show several other o/p to user interface page

a_1 = new Button ("submit");

a_2 = new Button ("cancel");

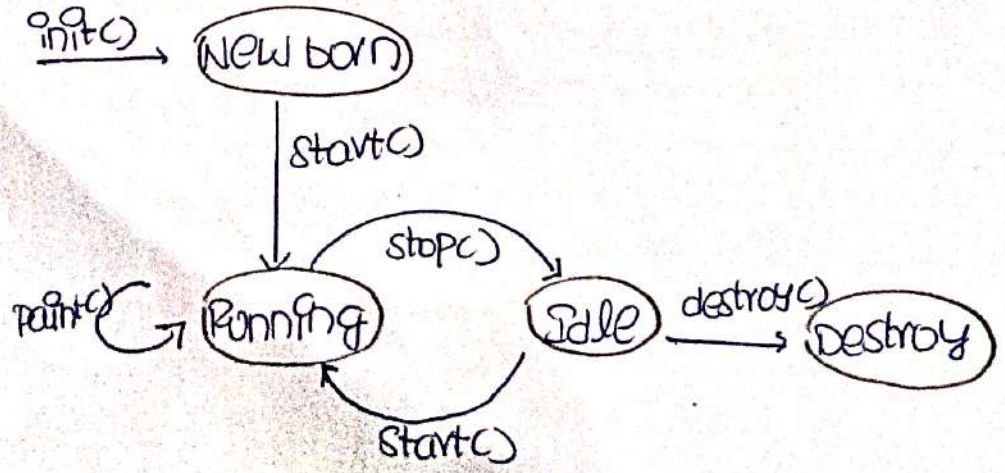
1) Write the Different between Java Application & Applet?

A:

Java application	Applet.
1. Java application contain Main Method.	1. An applet does not contain Main Method
2. Does not require Internet connection to execute.	2. Require the Internet connection to execute.
3. It stand alone application.	3. It is part of web page.
4. can be run without a Browser.	4. Require Java compatible Browser.
5. Uses stream I/O classes.	5. Use GUI interface provide by AWT or swings.
6. Entry points in Main Method.	6. Entry point is init method.
7. Generally used for Console program	7. Generally used for GUI Interface.
8. A large program.	8. A small program.

2. Write the life cycle of an Applet.

A:



10, Write the Difference between Java AWT & Java Swing.

A:

Java AWT	Java Swing.
1. AWT components are platform dependent.	1. Java swing components are platform independent
2. AWT components are heavy weight	2. Swing components are light weight.
3. AWT doesn't support pluggable look and feel.	3. Swing support pluggable look and feel.
4. AWT provides less components than swing.	4. Swing provides more powerful components such as Table, lists scroll panes color chooser, tabbed pane etc.
5. AWT doesn't follows MVC	5. Swing follows MVC.

11, Explain about MVC architecture in detail.

A: Model view controller or MVC as it is a popularly called. It a software design pattern for developing web applications.

2. A Model view controller pattern is made up of following 3 parts.

3. They are
   Model
   View
   controller.

4. Model: The lowest level of the pattern which is responsible for maintaining data.

5. View: This is responsible for displaying all or a portion of the data to the user.

6. Controller: Software code that controls the interactions between Model and view

1, Explain garbage collection in java?

A: The main objective of garbage collector is to free heap memory by destroying unreachable objects.

~~2, What is the necessity of oop's concept in java?~~

~~A: The main objective of garbage collector is to free heap memory by destroying unreachable objects)~~

3, What is the necessity of oop's concept in java?

A: The main aim of object-oriented programming is to implement real-world entities,

Ex: object, classes, abstraction, inheritance, polymorphism etc

3, What is class and object?

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

2, An object is an instance of a class.

4, What is package?

A: Package in java is a collection of classes, subpackage and interface.

Syntax:

Package name of package;

5, List the access modifiers in java.

A: 1, Default
2, Public
3, Protected
4, Private

6, Define the interface in java?

A: The interface in java is a mechanism to achieve abstraction, and multiple inheritance in java

It contain only abstract methods.

7) List the few checked Exceptions?

A: 1, IO Exception
2, SQL Exception
3, ClassNotFound Exception
4, IllegalAccessException.

8, Define checked and unchecked Exceptions:

A: checked Exceptions: The classes that extend Throwable class known as checked Exception
These are checked at compile time.

Unchecked Exception: the classes that extend Runtime Exception are known as unchecked Exception. these are checked at Runtime.

9. List out the OOP's concept.

A: 1, Abstraction.
2, Encapsulation
3, Inheritance
4, Polymorphism.

10, What is meant by constructor?

A: A constructor in Java is a special method that is used to initialize objects.
The constructor is called when an object of a class is created.

11, How to declare a variable and use of the variable?

A: Syntax:
type variableName_value;
variable in Java is a data container that saves the data value during Java program execution.

12, What is inheritance?

A: It is a mechanism in which one object acquires all the properties and behaviour of a parent object.

13, List the advantage of interface?

A: 1. We can implement multiple inheritance.
2. Interface function to break up the complex design and clear the dependencies between object.

14, What is an exception and when it occurs?

A: An exception is a problem that arises during the execution of a program.

It occurs when user entered an invalid data. A file that needs to be opened cannot be found.

15 How to handle Exception in java?

A: We can handle Exception by using keywords

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

16. What is method overloading?

A: If a class has multiple methods have same name but different in parameter, it is known as method overloading.

17, Write about the Access Modifiers in java?

A: Access Modifier are keywords that can be used to control the visibility of fields, methods, and constructors in a class.

18, What is data Abstraction?

A: Data abstraction is the process of hiding implementation details and showing only functionality to the user.

19, Define superclass and subclass?

A: subclass: A class that is derived from another class is called as subclass.

Superclass: The class from which the subclass is derived is called a superclass.

20, Types of inheritance in java?

A: 1. Single inheritance.
2. Multiple Inheritance.
3. Multi-level Inheritance.
4. Hierarchial Inheritance
5. Hybrid Inheritance

21, How to import a package?

A: import package name.

22, compare throw and throws keyword

A: The throw keyword is used to explicity throw an exception within a method or block of code, while the throws keyword is used to declare which exceptions can be thrown from a method.

23, What is the use of finally keyword?

A: The finally keyword is used to execute code no matter if there is an exception or not.

24, What is the use of Recursion in a program?

A: Recursion provide a way to break complicated problems down into simple problems which are easier to solve.

25, What is a method and its advantage?

A: A method is a block of code which only runs when it is called.

Method can be called several times in the same program.

26, What is a synchronized thread?

A: Synchronization is a process of handling resource accessibility by multiple thread request

The main purpose of synchronization is to avoid thread interface.

27, What is Daemon thread and how it can set dameon thread?

A: A Daemon thread is a background service thread which runs as a low priority thread and performs background operations like garbage collection.

The setDaemon() method of thread class is used to mark the thread either daemon thread or a user thread.

28, What is an Event?

A: An event can be defined changing the state of an object

Ex: clicking of button.

4, What are the uses of using synchronized Method in Java?

A: Synchronized Methods enable a simple strategy for preventing thread interface and memory consistency error.

5, What is start(), run(), sleep() Methods in Multithreading

A: 1, start(): It is used to start the execution of the thread.
2, run(): It is used to do an action for a thread.
3, sleep(): It sleeps a thread for specified amount of time.

6, What is Adapter class in AWT?

A: Java Adapter class provide the default implementation of listener interface.

If you inherit the adapter class, you will not be forced to provide the implementation of all methods of listener interface so. It save code.

7, Write about Buttons in AWT?

A: A button is Basically control component with Label that generate an event when pushed.

8, Write about, use Textfied in Event?

A: The textfield component allows the user to edit single line of text. When the user type a key in the textfield the event is sent to the textfield.

The key event may be pressed, key released or key pressed, or key typed. The key event is passed to register keylisten

9. List the textfield component?

A:
1. Place holder
2. clear Button
3. Text Alignment
4. prefix and suffix
5. Small variant
6. pattern.

10 List out available classes in swing container.

Ans The Javax.swing package provide classes for Java swing API such as JButton, JTextfield, JTextArea, JRadioButton, JCheckbox, JMenu, JColorchooser etc

11 What is JComponent?

Ans The JComponent class provides the base class for both standard and custom components that use the swing architecture.

12 How to Declare JTextfield class?

Ans Public class JTextfield extends JTextcomponent implements swing constants.

13 List out the Layout Manger Types

Ans
1. Border Layout
2. Box Layout
3. card Layout
4. Flow Layout
5. Grid Bag Layout
6. Grid Layout
7. Group Layout
8. Spring Layout.

14 List a few Mouse Listeners Methods.

Ans
Public abstract void MouseClicked (MouseEvent e);
Public abstract void mouseEntered (MouseEvent e);
Public abstract void mouseExcited (MouseEvent e);
Public abstract void mouse Exited (MouseEvent e);
Public abstract void mouse Released (MouseEvent e);

15 What are MouseMotion Methods.

Ans Public abstract void mouseDragged (MouseEvent e);
Public abstract void mouseMoved (MouseEvent e);

16 What is Inter-thread communication and write its Methods?

Ans Inter-thread communication is the process in which two threads communicate with each other by using wait(), notify(), and notifyAll() Methods.

17, What is the thread group and how to create the thread group?

A: 1, ThreadGroup creates a group of threads.
2, It offers a convenient way to manage group of threads as a unit.

```
ThreadGroup tg1=new ThreadGroup("GroupA");
Thread t1 = new Thread(tg1, new MyRunnable(), "one");
Thread t2 = new Thread(tg2, new MyRunnable(), "Two");
```

18, List out the Event source?

A: The event source, is an object that is created when an event occurs.
1. Java provides several types of event occurs.

1. ActionEvent
2. ContainerEvent
3. Key Event
4. Window Event
5. Mouse Event.

19, List out Event classes and Listener Interfaces.

A: Event classes.
1. Action Event
2. AdjustmentEvent
3. ComponentEvent
4. ContainerEvent
5. FocusEvent
6. InputEvent.

Listener Interfaces.
1. ~~the~~ ActionListerInterface
2. Adjustment Listener
3. component Listener
4. ContainerListener
5. FocusListener
6. InputListener.

20, Write the Design Goals and Delegation of Model?

A: 1, It is easy to learn and implement.
2, It supports a clean separation between application and GUI code.

21, List out Methods of KeyListener Interfaces.

A: 1, void keyPressed (KeyEvent e)
2, void keyReleased (KeyEvent e)
3, void keyTyped (KeyEvent e)

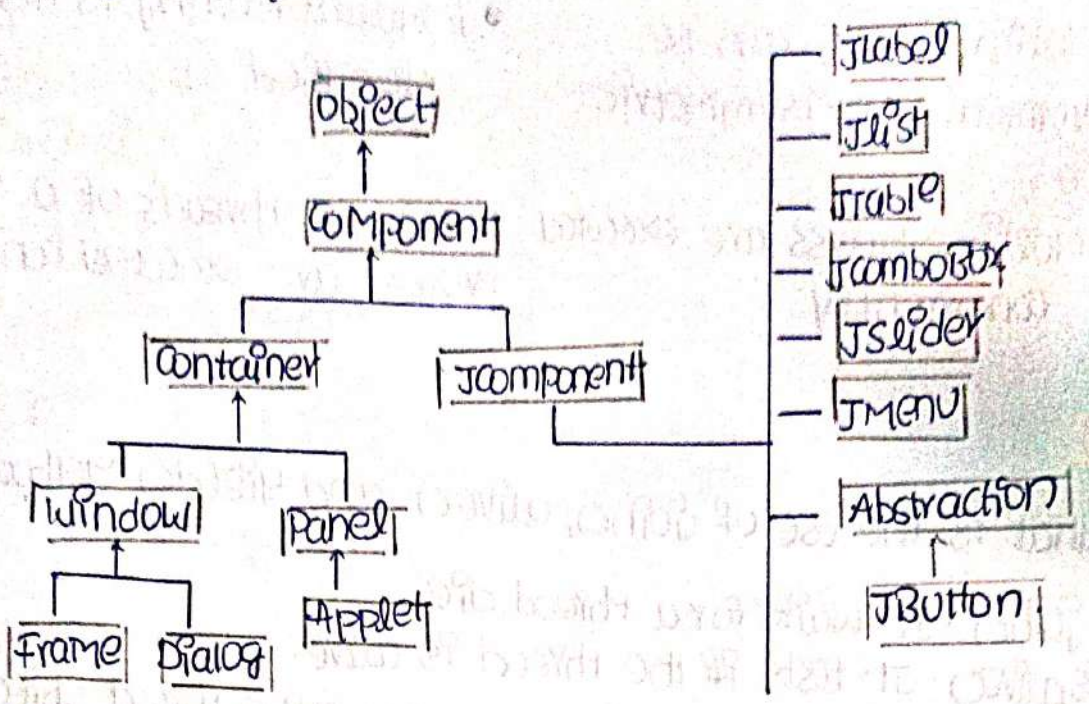
22. List out the Different Between Applet and Swing.

A:

AWT	Swing.
1. AWT component are heavyweight component	1. Swing component are light weight component
2. AWT doesn't follow MVC	2. Swing follows MVC

23. Draw the swing Hierarchy?

A: Swing Hierarchy:



24. What is MVC in Java.

A: MVC Pattern stands for Model-View-Controller Pattern.
i. This pattern is used to separate application concern.
ii. MVC is a pattern in software design commonly used to implement user interface, data and controlling logic.

25. What is JLabel and write its syntax?

A: JLabel(): Create a blank label with no text or image in it.

Public class JLabel extends JComponent implements Swing Constants Accessible.

26. How we can use Buttons in AWT?

A: The Button class is used to create a labeled button that has platform independent implementation.

27. What is an Applet?

A: 1. It is a small software program that supports a large application program.
2. Java applet are used to provide interactive features to web application can be executed by Browser for many platform

28. List out Difference Between Multiprocessing & Multithreading?

Ans

Multiprocessing.	Multiprocessing
1. Multiprocessing can be symmetric or asymmetric	1. Multithreading is not classified
2. Multiple process are executed concurrently.	2. Multiple threads of a single process are executed concurrently.

29. What is the use of join(), alive(), and yield() Method.

Ans join() : It waits for a thread die.
isAlive() : It tests if the thread is alive.
yield() : It caused the currently executing thread object to pause and allow other threads to execute temporarily.

30. What is Layout Manager and list its types?

Ans A Layout Manager is an object that controls the size and position of components in the container.

31. List out the Listener Interface?

Ans
1. Event Listener Interfaces.
2. The ActionListener Interface
3. The AdjustmentListener Interface.
4. The ComponentListener Interface
5. The ContainerListener Interface.
6. The FocusListener Interface.
7. The ItemListener Interface. etc.

32. What is Event Listener and its classes.

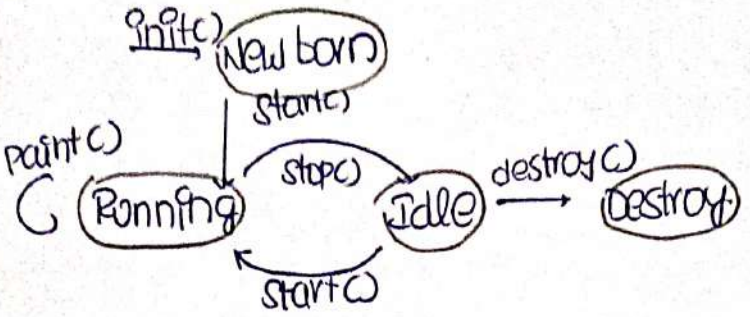
Ans The Event Listener represents the interface responsible to hand events.

Every Method of event listener Method has a single argume as a object. which is subclass of EventObject class

33. List out the Event sources.

Ans
1. ActionEvent
2. ContainerEvent
3. KeyEvent
4. WindowEvent
5. MouseEvent

34. Draw the life cycle of Applet.



35, Difference Between Java Application and Java Applet?

Ans

Java Application	Applet.
1. Java application contain a Main Method.	1. A applet does not contain a Main Method.
2. Is stand alone application.	2. Is a part of webpage.
3. Generally used for consale program.	3. Generally used for GUI interfaces.

36) What is swing and how is Better than the AWT.

Ans Java Swing tutorial is a part of Java foundation classes (JFC) that is used to create window-based Applications.

AWT uses Native Library call to draw all of it's components.

∵ Swing uses the Graphics2D API to draw all it's components

37) What is GUI its advantage.

Ans Graphical user Interface in Java is an easy-to-use visual experience builder for Java application.

Advantages

1. Easy Use.
2. Easy understanding
3. Attractiveness
4. shortcuts
5. Multitasking

Q. What contains the container class in swings?

Ans JPanel, JFrame and JWindow.