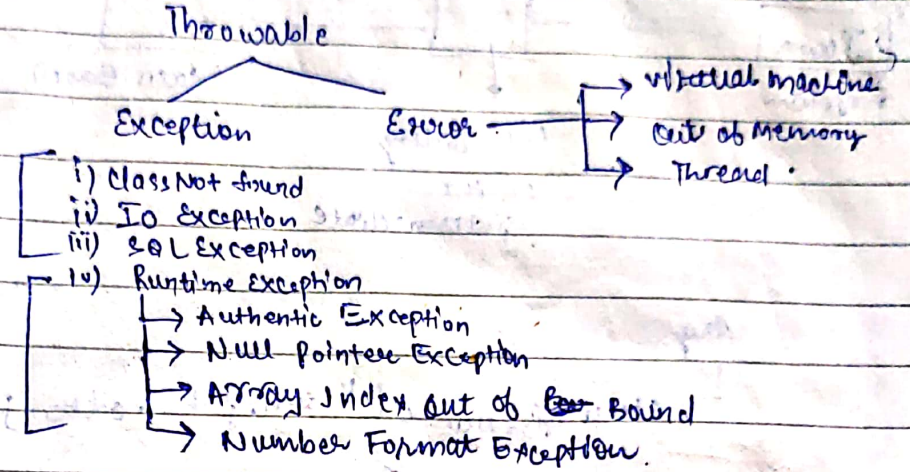
Exception

⇒ difference between Exception and Error?



* What is checked and unchecked exception?

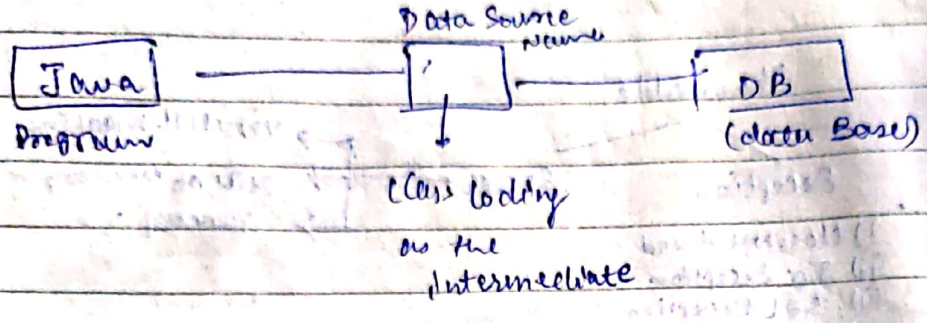
are those, that cannot be ignored when we write the code in our methods. All exceptions instantiated from Exception class or from the subclass of Exception other than runtime exception and it's subclass.

It occurs due to the mistake of programmer, so we must provide the handler for it. It checks at compile time.

- Unchecked exception is unknown to the compiler.

Constitute only RUN TIME EXCEPTION and Errors.

- No two exception cannot occur simultaneously.




```
try
{
    Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
    ---
    ---
}
```

Exception goes to JVM (catching it), to avoid this we have "catch" Block.

```
int x;
try
{
    S.o.p("Before Exception");
    x = 10/0;
    sop("After exception");
}
```

~~Linux to U~~

```
catch ( ArithmeticException i)
  {
      sop( i);
  }
```

One more solution. (global one)

```
catch ( Exception i)
  {
      sop( i);
  }
```

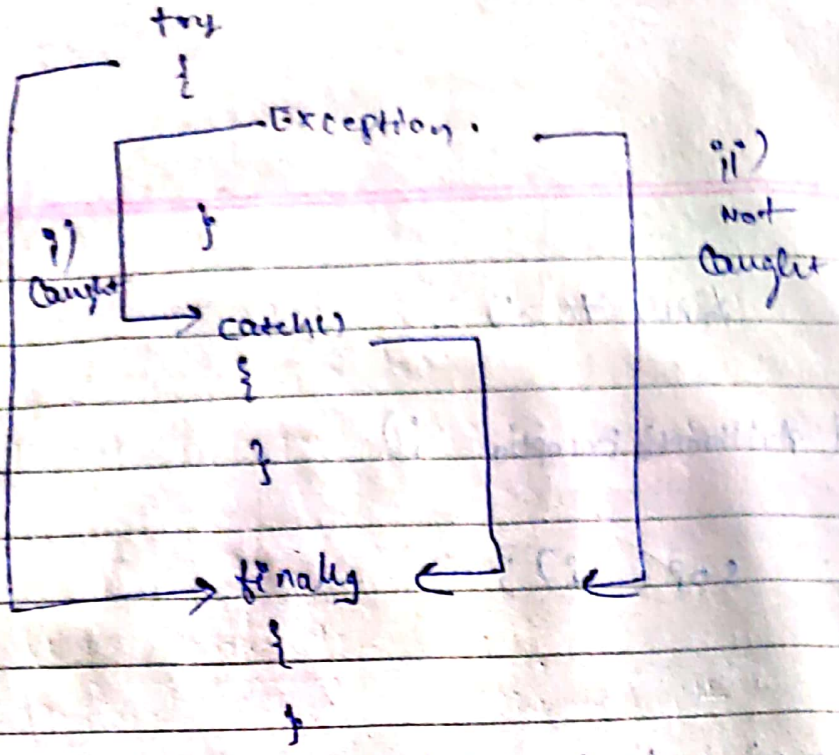
Q What is the diff. b/w final, finally, finalize?

→ finally is a Block that associates with try block.

- Whenever exception comes then following seq. follows
  1. Catch
  2. finally

or

1. finally.



* We can't give more than one final block with a single try block.
* finally always executed whether the exception occurs don't occur, caught, don't caught.
* It's priority is ~~high~~ last.
* If a statement `System.exit(0)` occurs before finally then finally will not execute.

exit(0) → Normal termination
exit(1) → AbNormal termination (forcefully)

```
int show()
{
  try
  {
    return 10;
  }
  catch (Exception i)
  { sop(i);
    return(20);}
  finally
  { return(30); }
}
```

```
int show()
{
  try
  {
    return 10;
  }
  catch (Exception i)
  {
    return 20;
  }
  finally { return 30;}
}
```

```
int show()
{
  int x;
  try
  { x=10/0;
    return 20;
  }
  finally
  { return 30; }
}
```

```
int show()
{
  try { }
  catch()
  {
    System.exit(0);
  }
  finally
  {
  }
}
```

* We should avoid return statement in finally
* finally with only try ___ No catch.
  we shld always avoid this stt., bcoz after finally prog. will terminate.
* finally always have greater priority except System.exit(0);

Unix

① Some unix projects (BSD and open solaris) are open source.

② Kernel is monolithic

Linux Commands -

① Internal Com. : are the in built command
for eg. i) Echo : the system won't look in it's path to locate it (for printing the message)
ii) type : for seeing the content of a file.

External Commands

The System has to find out where it is located
for eg ls : for listing the files

Q Find out the various opt. along with "ls" command

man - ls.

help command.

$man ls.

? where password

② echo "Computer" ⇒

for long comment use

" Navdeep . . .

>

"

?

? : denotes the continuation of the .

③ $ password ⇒ to change the password. (w

④ $ who ⇒ to know the login names (current users)
or
$ w

⑤ for merging two commands

$ password ; who

⑥ $ tty : to know the terminal number.

⑦ $ locle : used to lock the terminal.

⑧ $ stty : setting terminal settings.

[Q Attributes of stty commands]

⑨ $ clear : clearing the screen.

⑩ $ tput cup x y : moving the cursor.

⑪ $ uname : used to known the machine name.

↳ Q It's attribute

⑫ $ date : find current date.

⑬ Along with the date, we can find month

⑭ $ cal : for calender

Scanned with CamScanner

eg i) $ cal Aug
ii) $ cal 2010

(15) $ bc : fundamental calculation.
eg $ bc 2+3

(16) $ pwd : present working directory
display

. $ echo $ ~~bam~~ home → change to home directory.

(17) $ cd : Changing the directory
(18) $ mkdir : for making a directory.
(19) $ rmdir : to remove directory.
(20) $ copy file1 file2
[Attributes of copy ...?]
(21) $ rm * : remove all the

(22) $ mv : move a ~~file~~ from one place to another
→ it removes a file
→ moves a grp of file to a diff. directory

$ mv file name1 filename2

(23) $ cat : i) display the content of already existing file
ii) otherwise create a file
ctrl+d must press to terminate and saving.

(24) $ df : to find free disk space.
(it is used to display the amount of free space avail. on disk)

(1) Firewall
(2) Proxy Server → PAM, LDAP (5, 6)
(3) Pkg - Signed

Our friday.

(4) IDS (Intraction detection)
: Karvenos

---

(25) $ du : diskusage
(used to find the consumption of memory)

(26) $ chmod : (manage mode)
Category: u : user o : others
g : group a : all

Permission.
r : read x : execute.
w : write

Operation.
+ : assign permission
- : remove
= :

i) $ chmod a+x f1.
(means we are assigning all user to execute the file f1)
ii) $ chmod ugo+w f1
iii) $ chmod o -rwx f1

read : 1 0 : (zero) for Access Denied.
write : 2
execute : 4

iv) $ chmod 777 f1 (user group other)
(1+2+4)

$ chmod g=rx f1

(29) $ touch [filename] : to create a file.

FILTER.

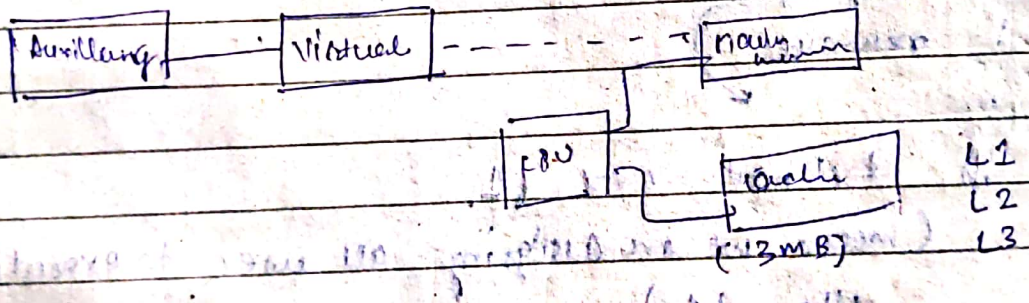
(28) $ more : display the content of file page wise.

$ wc : 1) word count
2) characters
3) No. of ~~fffr~~ lines

Computer Architecture

1. Length of Instruction
2. clock Rate

Memory Organization Hierarchy
- Main memory
- Auxiliary memory
- Virtual
- Cache



Where does virtual memory resides?

Multiprocessor & Multicomputer

What are Buses?

120 lines used as Bus

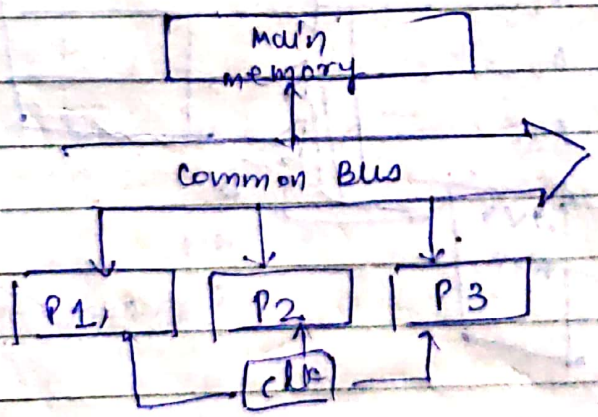
- Power supply
- Data lines → Bidirectional → (multiple of 8)
- Control lines
- Address lines → unidirectional

⇒ Design multiprocessor system?

Requirement

1. Two or More processor
2. Communication (Interprocessors Comm.)
3. Master-slave identification
4. Async/ sync → Benefical.

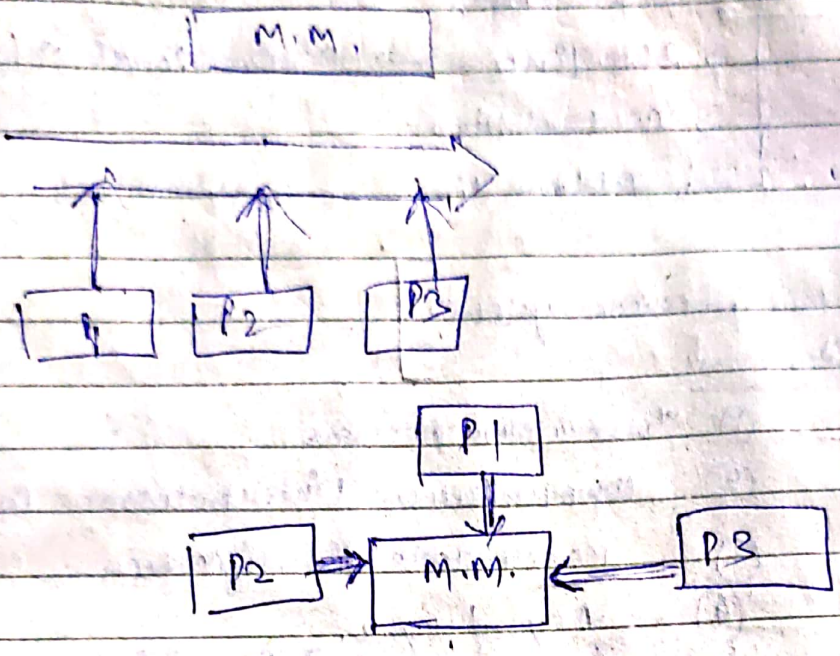
If this, then the o/p sometime desired. or either undesired.



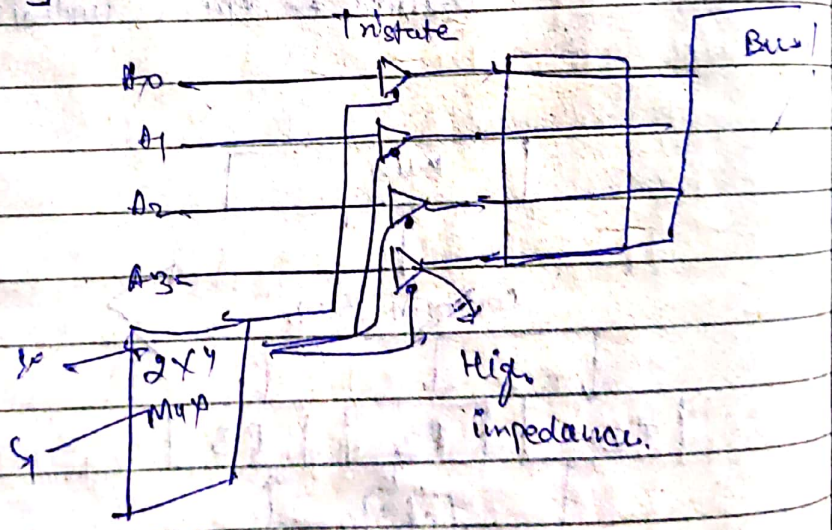
Synchronization can be done by i) semaphores
ii) Local clock.
Global clock.

- Benefit of Multiprocessor Sys. is → 1. Redudancy
  2. If one is bail, other can perform.

⇒ How a Multiprocessing can be done in parallel way?



• What is Tristate?



→ Independent of Physical D.B.

Physical DB (coding part)

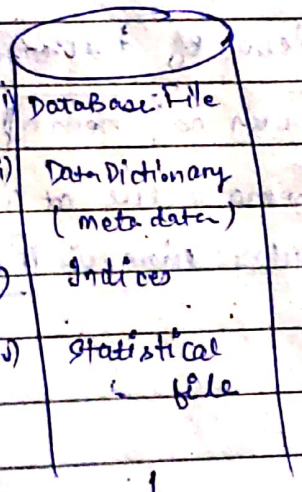
1 Naive user
2 App. Programmer (deal with CODING)
3 Sophisticated user
4 Specialised user

Data Definition Language :, for creating the data Base. (Create)

DML → Insert, update.
(modification)

DQL → selection queries.
Query

DCL → provides authorisation and rights
Control
⇒ eg save Point, check point,



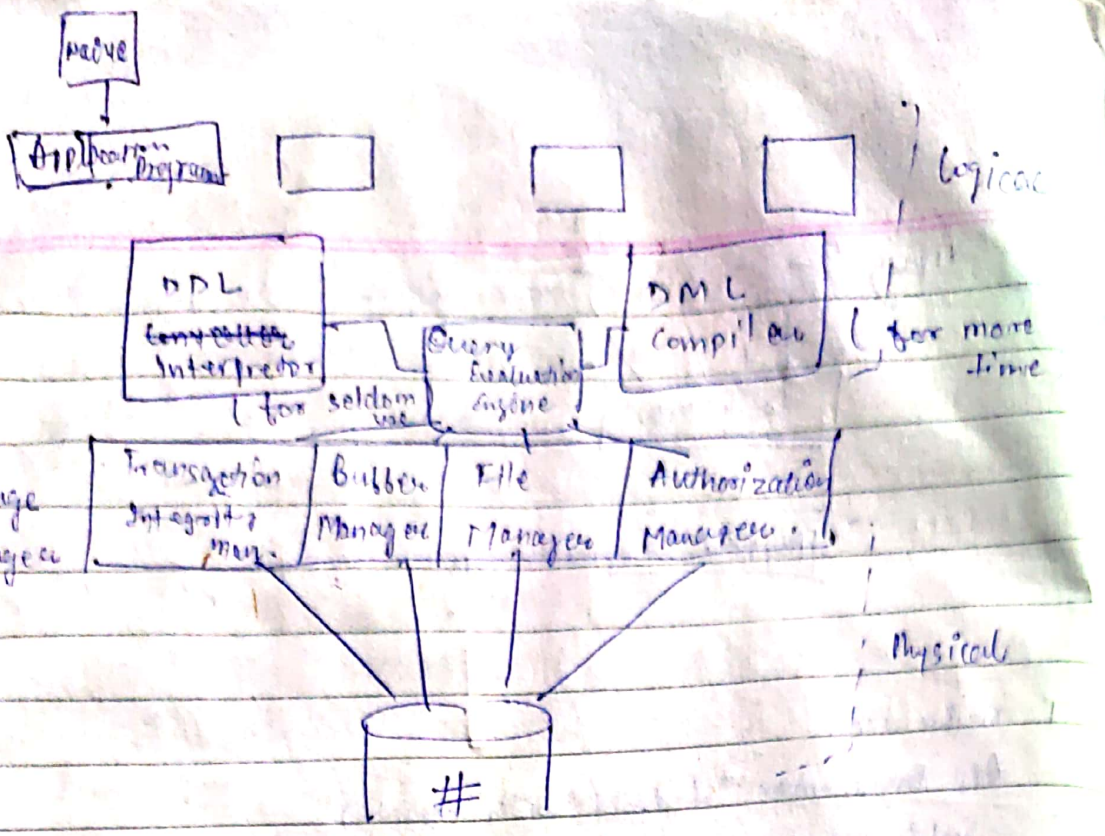
Properties. (ACID) of transaction

1) Atomicity
(all or none not half)

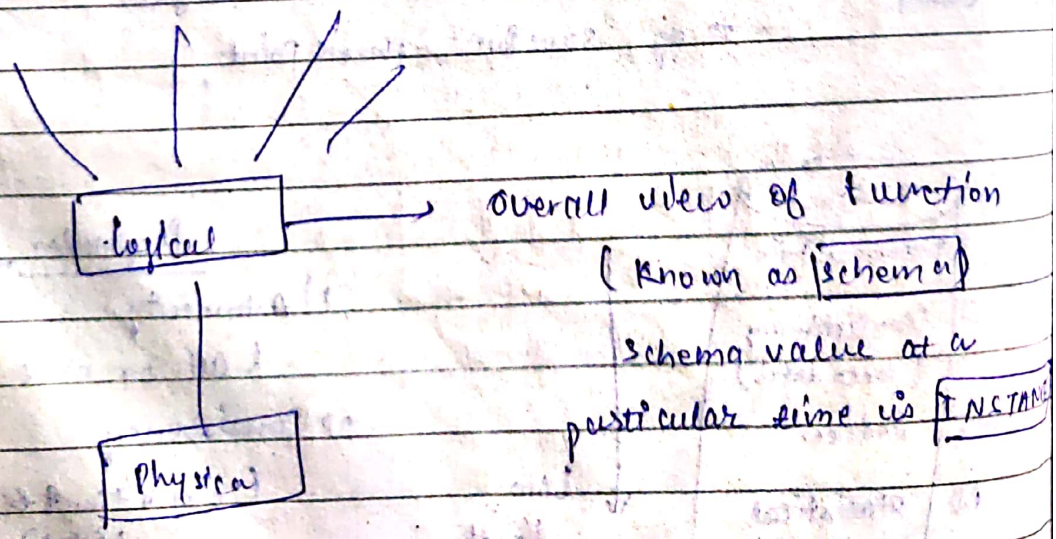
2) Consistency
(amt. A + amt B = amt A (After) + B (Before))

3) Integrity or Isolation
(adding the constraints (conditions))

4) Durable (Persistent)



• DDL & DML Analogy — mixing (1) Petrol + water
2) Hindi + English)



DBA → DataBase Administrator (Most Powerful man)

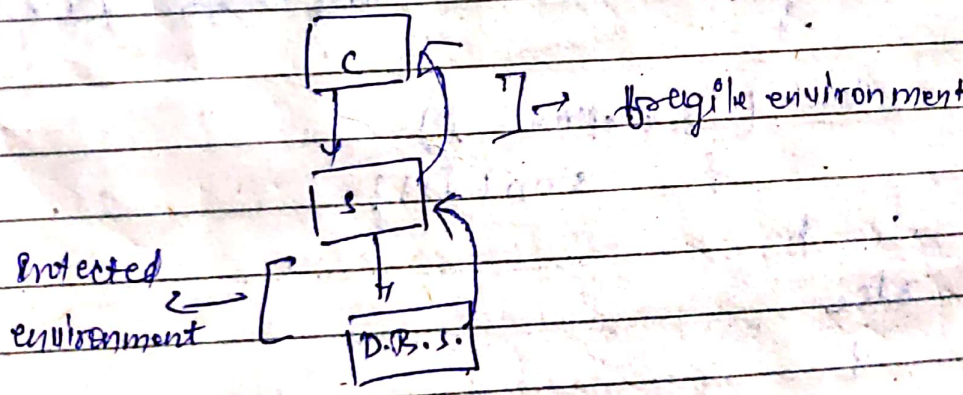
(A) Provide → (1) Routine Management
(2) Schema Definition
(3) Authorization.

[Q what is server?]

2 tier.

⇒ only client & server involved.
⇒ D.B. is associated with server.
(it is a drawback)

3 tier.



JAVA.

throw Keyword is used to throw the exception explicitly by the programmer, i.e. we want to throw the Exception according to our wish.

```
class Temp
{
    int age;
    void get (int age)
    {
        if (age < 10)
            {
        try
        {
            throw new ArithmeticException ("...
        }
        catch ( Exception i)
            { sop(i); }
    }
    else
    {
        this.age = age;
    }
}
psvm ( String s[])
    {
        new Temp() . get (10);
    }
```

⇒ All exception classes in Java, must have the constructor which takes the string arguments.

```
catch ( Exception e)
{ printStackTrace()
  s.o.p ( e.getMessage() );
}
```

Throws :

⇒ If a Method is capable of causing an exception that it does not handle, it must specify this behaviour so that the Caller of the method can guard themselves against that exception.

We achieve this, by adding a **throws** clause in method declaration. A **throws** clause list a type of exception that a method might throws

```
class Temp
{
  void get() throws AE, SQLE
  { int x = 10/0; }
  psvm ( String s[])
  {
    try {
      newTemp().get();
    }
    catch ( AE e)
    {
      s op(e);
    }
  }
```

Custom Exception,

```
class MyException extends Exception
  {
      MyException (String s)
      {
          super(s);
      }
  }
Class Temp
{
  int age;
  void get( int age ) throws MyException
   {
     If (age<18)
     throw new MyException("Invalid Age");
     else
      this.age =age;
   }
  public()
  {
    try
    {
      new Temp().get(10);
    }
  Catch ( MyException e)
    {
        Sop (i);
    }
```

Linux LABS.

ls: List directory contents.

i) -a, --all : do not hide entries starting with .

ii) -A, --almost-all : do not list implied . and ..

iii) --author : print the author of each file.

iv) -b, --escape : print octal escapes for non graphic ch.

v) --block-size=SIZE : use SIZE-byte blocks.

vi) -B, --ignore-backups : do not list implied entries ending with ~

vii) -c with -lt : sort by, and show, ctime (time of last modification of file status information) with -l: show ctime and sort by name otherwise: sort by ctim

viii) -C list entries by columns

ix) --color [=WHEN]
control whether color is used to distinguish file types. WHEN may be 'never', 'always', or 'auto'

x) -d, --directory:
List directory entity instead of contents, and do not dereference symbolic links.

xi) -D, --dired
generate o/p designed for Emacs' dired mode.

xii) -f do not sort, enable -aU, disable -lst.

stty : change and print terminal line setting

(1) -a, --all : print all current settings in human readable form.

(2) -g, --save : print all current settings in stty readable form.

(3) -F, --file = DEVICE
open and use the specified DEVICE instead of stdin.

(4) --help : display this help and exit.

(5) --version:
output version information and exit.

uname
print system information

-a, --all : print all info., in the following order

-s, --kernel-name : print the kernel name.

-n, --nodename : print the network node hostname

-r, --kernel-release : print the kernel release

-v, --kernel-version : print the kernel version

-m, --machine : print the machine h/w name.

-p, --processor : print the processor type.

-i, --hardware-platform : print the hardware platform

-o, --operating-system :
print the operating system.

--help : display this help and exit.

Multitasking

Process Based — Thread Based

- Two type of threads in JAVA
  - ⇒ Daemon Thread (-that thread that run to support user thread)
  - ⇒ User Thread - Main thread

SetDaemon (boolean) // set a user thread to Daemon thread

- Thread can be made by two methods:
  1) Extending Thread class
  2) ~~Running~~ interface Runnable

Thread is an independent seq. path of execution with in a program. Many thread can run concurrently (eg in MS word)

- At run time thread in a program exist in a common memory and can therefore share both data and code i.e. they are light weight as compare to process.

In JAVA, thread can be created using unique object of "java.lang.Thread" class.

- Two type of thread — User, Daemon

as long as User is alive JVM does not terminate.

Daemon is at the mercy of RunTime system. It is stopped if there are no more user thread is running.

When a standalone app. is run, a user thread is automatically created to execute the main method. This thread is called the **main Thread**.

Constructor of thread class.

1) Thread
2) Thread (Runnable r)
3) Thread (Runnable r, String name)
4) Thread (String &"name");
5) Thread (ThreadGroup tg, Runnable r);
6) " (" , " , String name);

A thread which is created based on an object, that implements the runnable interface, will execute the code defined in public method "Run". In otherwords the code in "run" method defines an independent path of execution and thereby the entry and the exit for the thread. A Thread ends when "run" method ends either by normal completion or by throwing an uncaught exception.

⇒ Procedure for creating Threads based on runnable Interface

(i) A class implement the runnable interface providing the "run" method. (An object of this class is Runnable object)

(ii) An object of **Thread** class is created by passing a runnable object as argument to the **thread** constructor.

The Thread object is new runnable object that implements Run method
The start method is invoked on the thread object created in previous steps.

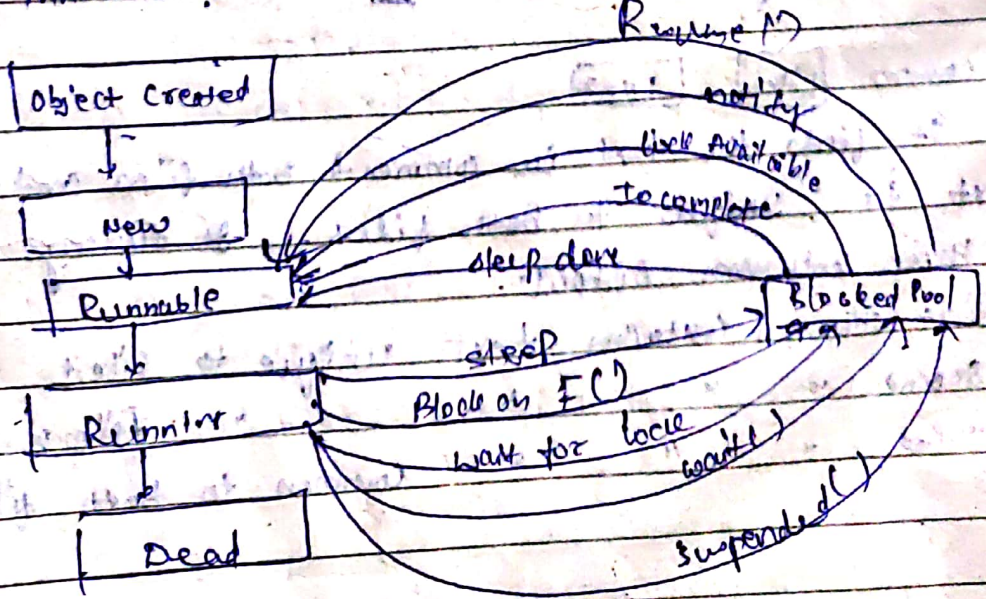
class Thread2 ; Implement Runnable
{
Thread t;
Thread 1 (string s)
{
t = new Thread (this, s);
t. start( );
}
public void run( )
{
}

II) Extending the thread class :
i) A class " " " " overwrites the run method from the thread class to define the part executed by the thread.
ii) This subclass may call a thread constructor in its const. or initialize the thread object using super call. The start method inherited from the thread class is invoked on the object of the class to make the thread eligible for running.

```
class Thread1 extends Thread
{
    Thread1(String s)
    {
        super(s);
        start();
    }
    public void run()
    {
    }
```

[Why implementing the Runnable interface may be preferable to extending the thread class?]

1. Light weight
2. Inheriting the thread class makes the sub class so heavy.
3. Java doesn't support multiple inheritance, so once you inherit a class it won't be further inherited.



JAVA

Thread t = Thread.currentThread();

t.getName();
t.setName();

Playing with threads:

```
class Threads
{   public static void main(String a[])
    {   Thread t = Thread.currentThread();
        sop("Current thread:" + t.getName());
        t.setName("mainchanged");
        sop(t.getName);
    }
}

class Threads2 extends Thread
{   Threads2(String s)
    {   super(s);
        start();
    }
public void run()
{
for(int i=0; i<5; i++)
    {   sop(Thread.currentThread().getName());

try
{   Thread.sleep(1000);
}
catch(Exception e)
    {   sop(e);}
}
}
}
```

```
class Thread3 extend Thread
    { = same as Thread2
    }
class RunThread
  { p s v m ( String s[])
      { Thread2 t2 = new Thread2("Thread1");
        Thread3 t3 = new Thread3("Thread2");
    for ( i=0; i<=5; i++)
        {
          try
          { Thread.sleep(1000);
          }
          catch ( Exception e)
          { sop(e);
          }
        sop("main thread");
  }
```

Synchronization in Thread:

lock

it can be associated with a shared resources, thread gain access to the shared resources by

i) first acquiring the "lock" associated with a resource

at any given time atmost one thread can hold the lock

Object lock Mechanism

A thread must acquire the object lock, associated with a shared resource. by it can enter into the shared resource.

The run time sys. ensures that no other thread can enter in the shared resource if another thread already hold the object lock associated with the shared resource.

There are two ways in which execution of code can be synchronized.

i) Synchronize method, ii) synchronize Block

A thread release the lock simply by returning from the synchronized method

for method

```
synchronized void disp()
{
}
```

for Block

```
synchronized
{
}
```

```
class Shared
{
    synchronized public void show(String s)
    {
        sop(" starting " + s);
        try { sleep(4000); }
        catch (Exception e) { }
    }
    sop("END of " + s);
}
```

```
class CustomThread extends Thread
{
    Shared s;
    public CustomThread(Shared s, String str)
    {
        super(str);
        this.s = s;
        start();
    }
    public void run()
    {
        synchronized(s)
        {
            s.show(Thread.currentThread().getName());
        }
    }
}

class RunSync
{
    public static void main(String[] args)
    {
        Shared sd = new Shared();
        CustomThread t1 = new CustomThread(sd, "one");
        CustomThread t2 = new CustomThread(sd, "two");
        CustomThread t3 = new CustomThread(sd, "three");
    }
}
```

We create a thread group, so that we can simultaneously work with a group of thread.

NIC
Network Installation Card

NIS
Network Information service.

Linux.

(1) ps : process statement (2) nice → reduce priority of process
(3) & → to run a process in background
(4) kill → $ kill or Terminate a process.
(5) bg → push the foreground job to background.

Network Installation:

first step is setting a networking on Linux is to install a NIC. several brands of NIC are currently available based on diff. types of chipsets. There are different types of installation such as FTP during or ntwork install or server or workstation config.

1) NIC : This is usually ethernet path and has name TTH0 & TTH1.

2) Host Name: The computers will be identified by its name on the internet. Don't use localhost name of yahoo.com or gmail.com

argument. The name of the file system to be mounted and second is the name of directory where it is to be mounted.

mount → /dev/NAV1 /user1

- /dev/ → device
- NAV1 → Name of file system
- /user1 → empty directory where mounting done

(3) unmounting file sys,

It is achieved by "umount" comm which req either the file system or the mount point as argument.

Syntax:

umount → user1

umount → /dev/NAV1.

unmounting the file system is not possible if some file is opened in it.

Cut-throat competition

Manager

Role

1) Interpersonal
   - Figurehead
   - Leader
   - Liaison
2) Informational
   - Monitor
   - Disseminator
   - Spokesperson
3) Decisional
   - Enterpreneur
   - Dist handler
   - Negotiator
   (अर्थात् और की मनाना)

Description
mgt function

① planning
clear target, information, collection, course of action, establish of policies

② Organizing (putting the thing in place)
getting into action, resource distribution, Delegation of authority, structure, rel^n for smooth coord^n.

③ staffing
No of person req., promotion.

④ Leading / Directing
Resolving conflict, Communication

⑤ Controlling
Comparing actual performance.

Mgt System: Framework for maintaining, managing, improving org. policies.

Why: profit, Competition, Globalization, Growth, Adaptability

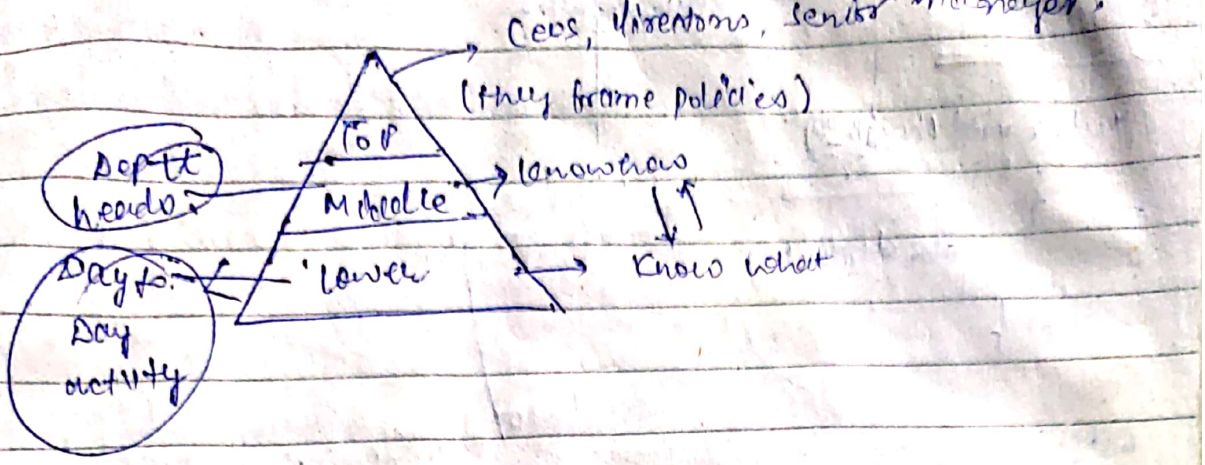
Mgt system, includes continously reviewing the policies, reshaping etc.

Effective M.S.

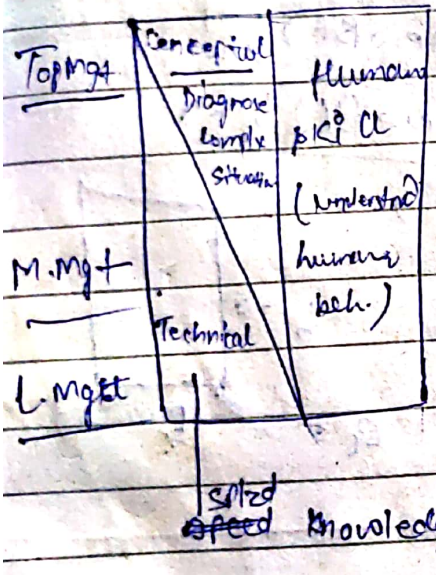
- manage social and financial risk.
- operational effectiveness
- Promote innovation
- Incr stakeholder satisfaction

6 Achieve control

Level of management



Managerial skill



Commonly used Byte oriented stream class are

(1) Input stream: is an abstract class that is extended by all I/P stream.

(2) Output stream: is an abstract class that is extended by all O/P stream.

~~Byte Array~~:

(3) ByteArray Input stream: this class is used to read bytes from a byte array.

(4) File Input Stream: - this class is used to read byte from a file.

(5) Buffer Input stream: This class is used to read bytes from Buffer.

(6) ByteArray Output Stream: This class is used to write bytes into a byte array.

(7) File Output Stream: This class is used to write bytes into a file.

(8) Buffered Output Stream: This class is used to write Bytes into a Buffer.

(9) Print Stream: This class provides methods that are used to write primitive types, String and object.

(a) InputStream class provide an abstract method read
i.e. is used to read byte from any I/p stream.

public int read() throws IOException;

-1 represents end of file.

(b) O/P stream class provide an abstract method write(), is used to write bytes on the output stream.

public void write(byte b) throws IO Exception

⇒ General signature of constructor.

[public TypeInputStream (Object source);
public TypeOutputStream (Object sink);]

Problems:

① To Read Byte from a ByteArray. bo = {'1','2','3',...}

ByteArrayInputStream b = new (bo);

b.read();

② To Read Bytes from a File (A.txt) → source.

FileInputStream.

ByteArrayInputStream bin = new ("A.txt");

bin.read();

③ To Read the data byte by byte from the Keyboard

BufferedInputStream bin = new (Keyboard);

in.read(); System.in