3. Exception Handling and multithreading.

The errors are coroing or mistaks that can make a program go coroing Errors may be logical or may be typing mistake

An error may produce incrocect output or may terminate the execution of the program.

So it is important to detect and manage errors

Types of Error

1] Compile time
2] Runtime

1] Compile time
When we Compile Java program the compiler will detect such errors and it will not create dot class file

Ex:
```
Class error1
 {
public static void main (String args[])
{
System.out.println ("Java"); missing
 }
 }
```

Above Program will show Compile time error as ; expected

System.out.println("Java");
                    ^

1 error

The common Compile time errors
1st missing Semicolony
2nd missing of Brackets
3rd misspeling of identifier & keyword
4th missing double code
5th use of undeclared variable

2] Run time Error
After Successfully compiling a program .class file is created.
But may not run properly due to wrong logic or Due to Run time error.

The commely occured Run time errors

1] main() method is not define then no such method error occured

2] When specific class or interface is not present in package

* Exception

An exception is an unusal situation that occurs Runtime. In other words exceptions are Runtime errors.

Ex:

```
Class error2
{
public static void main (String args[])
 {
int a = 10;
 int b = 5;
 int c = 5;
 int x = a / (b-c);
                    // Division by zero Error
 System.out.println("x" +x);
}
```

Java.lang.ArithmeticException : /0

- Types of Exception

1) check Exception
2) uncheck Exception

1] Check Exception
All those exception which Require Being caught and Handle during compile time are called as check Exception

Ex:
ClassNotFoundException

2] Uncheck Exception
All those exception which do not required Been Caught & handle during compile time are uncheck Exception

Ex:
ArrayIndexOutOfBound.

Exception Handling

Exception Handling mechanism actually provide mechanism to defect and Report and Exceptional circumtances so that corrective action can be taken

Exception Handling mechanism perform following task

1] Find the problem
hit the exception

2] inform that error has occured.
Throw the acception

3) Receive error information, Catch the exception

4) Take corrective action Handle the exception.

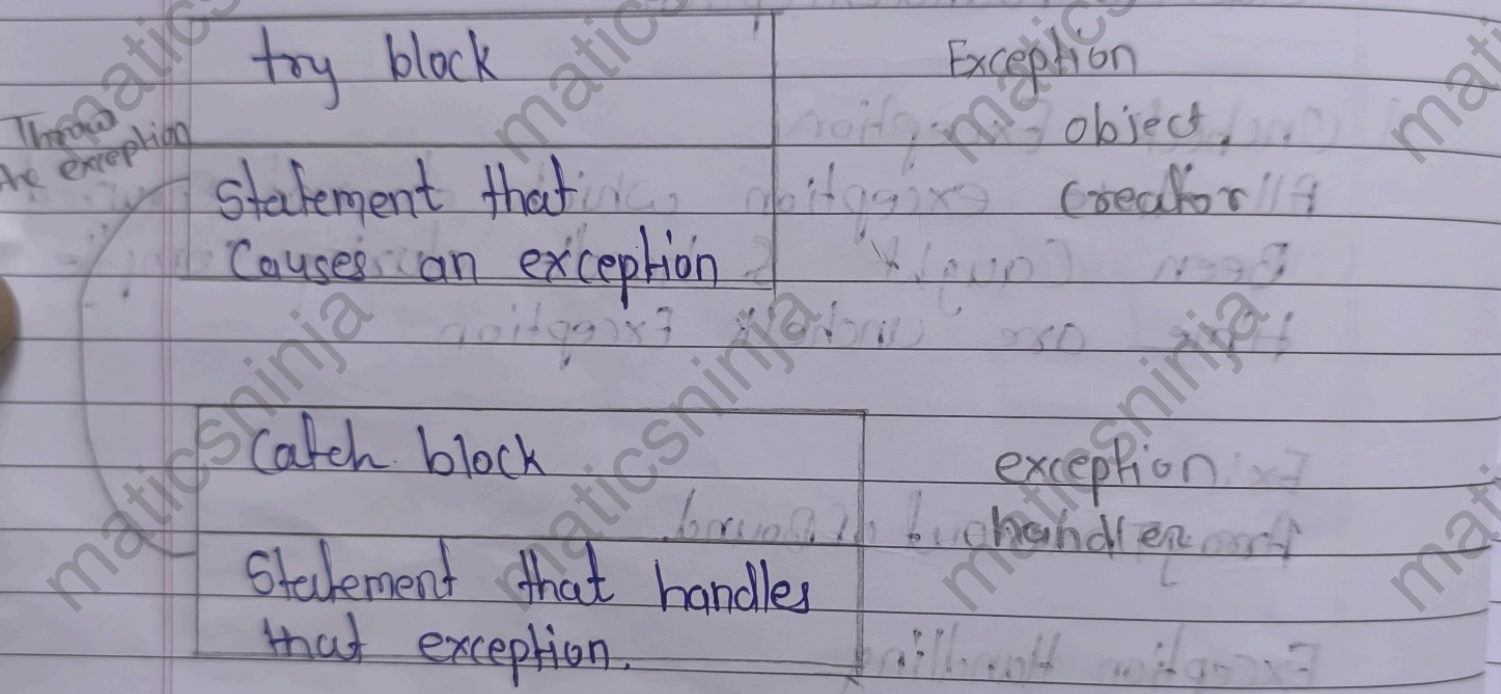


fig: Exception handling

General format of try

```
try
{
  Statement ;
  --11--
}
catch (exception type e)
{
  --11--
}
```

A Single try block may have multipk catch blocks Java try-block is used to enclosed the code that might through an exception.

if an Exception occures at particular Statement in try blocks Rest of code will not execute.

Program without Exception Handling

①
```
Public class trycatchexample
{
Public Static void main (String args[])
{
int data = 20/0;
System.out.println("Rest of code");
}
}
```

②
```
public class trycatchexample
{
Public Static void main (string args[])
{
try
{
int data = 20/0;
}
Catch (ArithmaticException e)
{
```

```
System.out.println(e);
}
System.out.println("rest of code");
}
}
```

o/p:
java.lang.ArithmaticException

When there is a single try block may have multiple catch block.

when there is a exception in try block when corresponding catch block handle the exception

Nested try

A try block within another try block is called Nested try

Syntax:

```
try
{
try
{
}
catch
{
}
}
```

- Name of Exception (Built in Exception)

1] Arithmatic exception
2] Filenotfound exception
3] IO Exception
4] ArrayIndexoutofBoundException

3] finally
The code coritten in finally block get executed even error occur which is not handle in the program

finally block executed in all the cases

4] Throw
The throw keyword in java is used to explicitly throw an exception from method or Any block of code used in method body.

5] Throws
Java throws keyword is used to declare an exception it is used with the method signature

Syntax:
Type methodname (parameter list) throws Exception list;

(iii)

```
import java.io.*;
Class m
{
Void method() throws IoException
{
throw new IoException ("Device error");
}
}

class testthrows example
{
public static void main (String args[]){
throws IoException
 m m = new m();
 m.method();
System.out.println("normal Flow of
                    exception");
 }
}
```

O/p

```
Exception in thread
 "main" java.io.IoException
    devic error
```

Difference between throw & throws

Throw	Throws
(I) It is used to throw an exception explicitly in the code inside a function	(I) It is used in method signature to declare an method exception
(II) This is followed by instance of Exception to be thrown	(II) The throw keyword followed by name of Exception
(III) Throw is used within the method	(III) It is used with method signature
(IV) We are allow to throw only one exception at a time	(IV) We can declare multiple exceptions using throws keyword.

Q4. Write a program to input name & age of person & throw and user defined Exception if the enter age is negative

→

```
import java.io.*;
class myException extends Exception
{
myException (string message)
{
super (message);
}
}

Class UI
{
String name;
int age;
~~public static void~~
Public Void input()
{
~~try~~
{
BufferedReader br = new BufferedReader
(New InputstreamReader (System.in));
System.out.println ("Enter your name");
name = br.readline();
System.out.println ("Enter age");
age = Integer.parseInt
(br.readline());
if( age < 0)
{
```

```
throw new myException ("Invalid age");
}
}
catch (Exception e)
{
System.out.println ("caught exception");
system.out.println (e.getMessage());
}
}

Public Static Void main (String args[])
{
UI u = new UI();
u.input();
}
}
```

- Combination of try-catch-finally

1) try-catch-finally → Mandatory block
2) try - finally
3) try - multiple catch finally

Syntax:

```
try
{
Statements = due to which exception occurs
}
catch (exception type -1)
```

{
≡
}
finally
{
Mandatory Statements
}

- Multithreading :

Process:
Process is a program in execution

Thread:
Thread is a light weight process each process has it's own PCB (Process Control Block) Context Switching, it Consist of two operation
① Context Save
② Context resume.

There are two types of multi-tasking

Process based:
A process is a program that is extend it allows to run two or more program Concurrently.

Thread - based:
The threads are light weight process they share same

Multithreading refers to an application with multiple threads running with in a process

Java interpreter handles the switching of control between the threads.

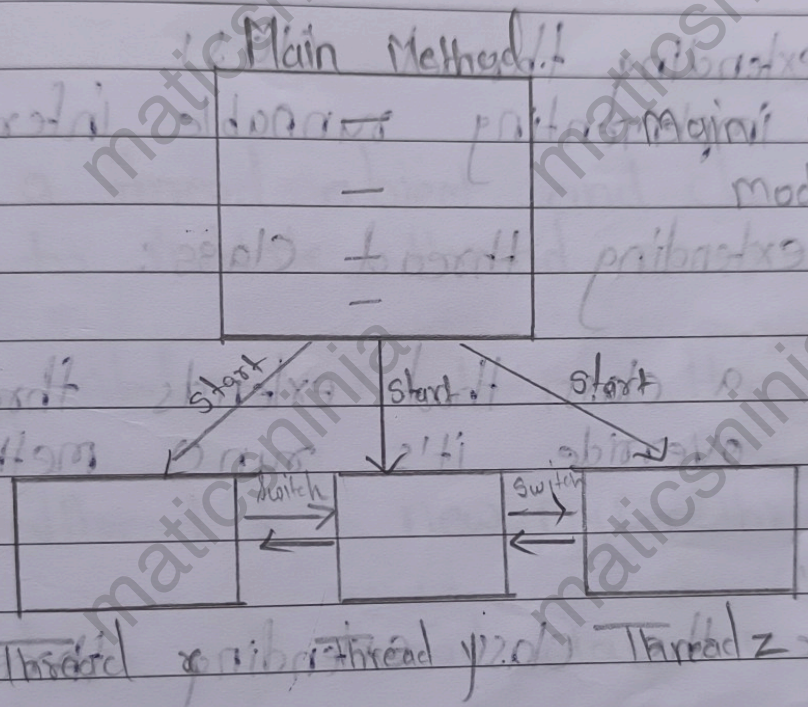


Fig: Java's Multithreaded program.

- Java program is shown with four threads
- one main and other three
- The main thread is actually main method module which is design to create and start other three threads which is x, y & z.

- once initiated by main thread the threads x, y, z run concurrently & share resources

Java Interpreter handles the switching of control between the threads.

How to create thread :
There are two ways of creating threads.

1] By extending thread class
2] By implementing runnable interface

① By extending thread class :

Define a class that extends thread class and override it's run() method

Syntax:

① Declare the class extending thread class.

```
class myfirstthread extends thread
{
    = statements;
}
```

Here my firstthread is created.

2] Implement run method.

The extending class must override run() method.

Syntax:

```
public void run()
{
  ==

}
```

3) Create a thread object and class start() method to initiate thread execution.

following statement creats new thread

① myFirstThread t = new myFirstThread();

Run instance of thread class or invoke thread method.

② t.start();

In first statement thread object is just created but not yet running

The second statement calls the start method to execute the thread.

Run Method

The run method is like a main method of thread life cycle its execution begin from run method.

```
class A extends thread
{
public void run()
 {
 int i;
 for(int i=1 ; i<=10 ; i++)
 {
System.out.println("Thread A:" +i);
 }
}

class B extends thread
{
public void run()
{
int i;
for(i=1 ; i<=10 ; i++)
{
System.out.println("Thread B:" +i);
 }
}

Class C extends thread
{
public void run()
{
```

```
int i;
for(int i=1; i<=10; i++)
{
System.out.println("Thread C :" + i);
}
}
}

class Threadexecubindemo
{
Public Static Void main (String args[])
{
System.out.println("main Starts");
A a1 = new A();
B b1 = new B();
C c1 = new C();
   a1.start();
   b1.start();
   c1.start();
System.out.println("main ends");
}
}
```

- By implementing Runable interface

1] Define a class that Implement Runable interface
Runnable Interface contains only one Method that is run method.

Ex:

```
class multi implements 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();
}
}
```

Thread Life cycle

Thread can enter into many stage / steps during its life time
The five types of steps are

1. New born step
2. Runnable
3. Running
4. block
5. dead

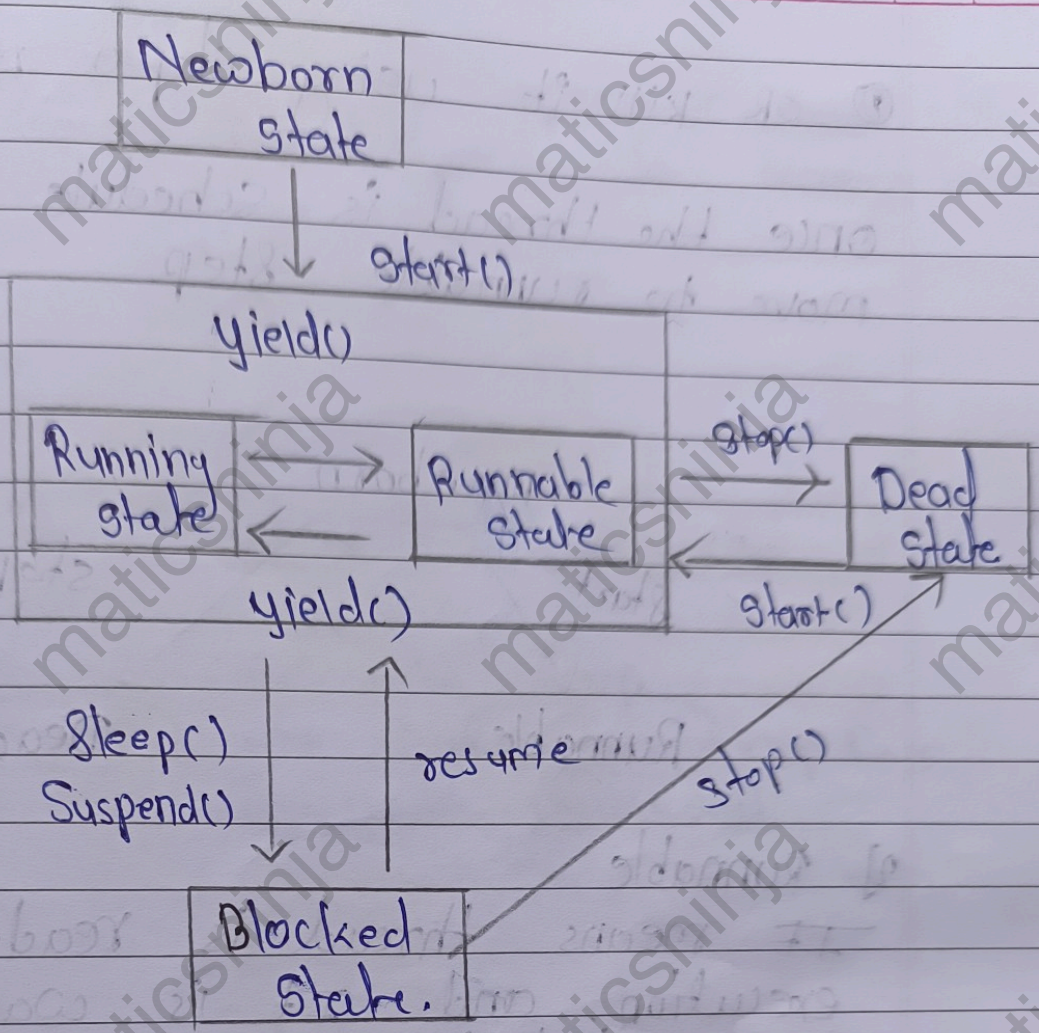


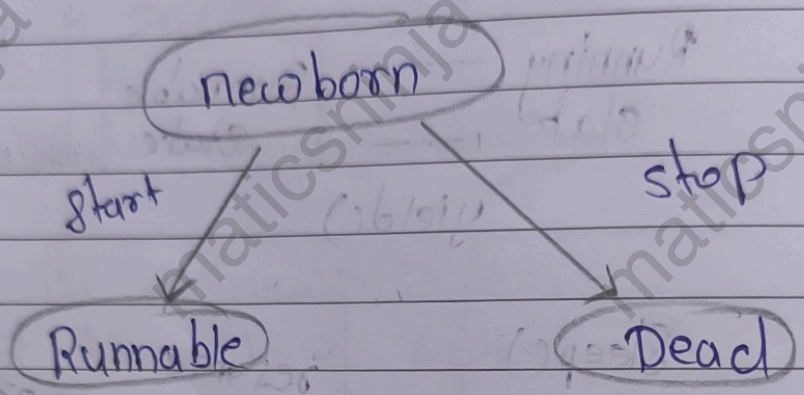
Fig:- Life cycle of thread

- The thread can be in 1 state from these five states
- It can move from one state to another state

1] Newborn
When thread object is created the thread is born and called as thread is in newborn step.
When the thread is in new born state It can be
① either shecudle for running using start() method

② OR kill it using stop() method

once the thread is schedule it can be move to runnable step.



2] Runnable

It means thread is ready for the execution and It is waiting for the availability of the processor.

3] Running

If Running state means that the processor has given its time to thread for its execution.

Running thread can release its control any one of the following situation

1] Situation

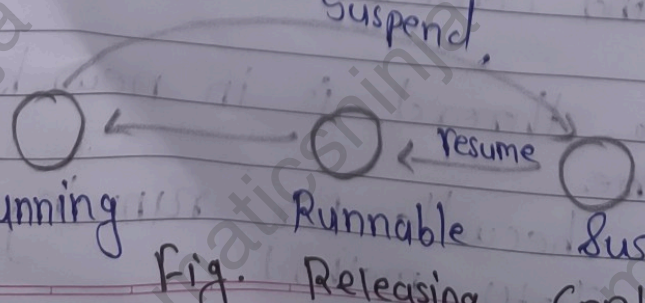
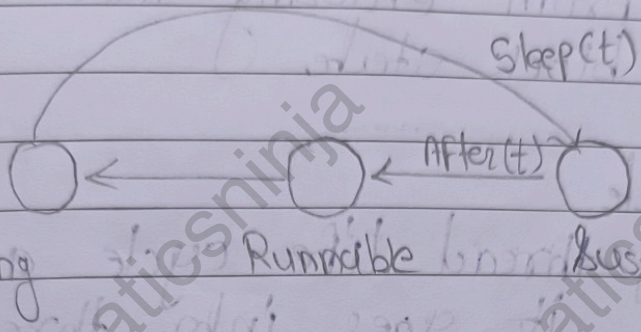


Fig. Releasing control by using

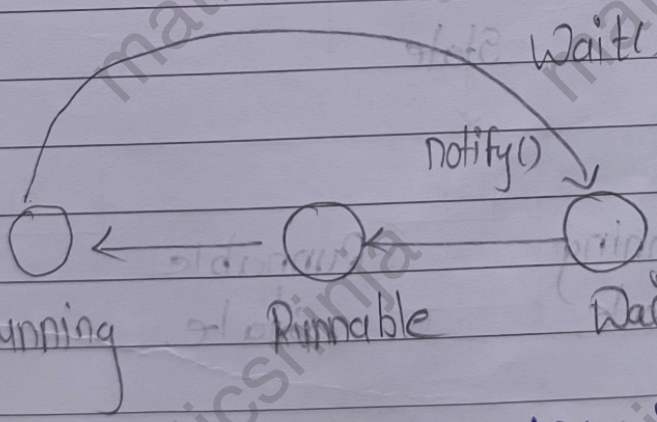
It is suspended using suspend method which can be used to suspend a thread form some time.

(ii)



- It can be made to sleep
- The thread can made to sleep for specified time given in milisecond.

(iii)



- Thread is waiting uti! some event occurs & wait(), notify() method used

Wait() method is used to wait for condition

notify() method is used to schedule thread to run again.

block State ()
- block means not runnable that is dead
- That is fully eligible to run again.
prevent from entering into runnable step
called block state ()

Dead State
As every thread life cycle after completing execution it goes into thread state it is natural dead of thread.

HW

1]

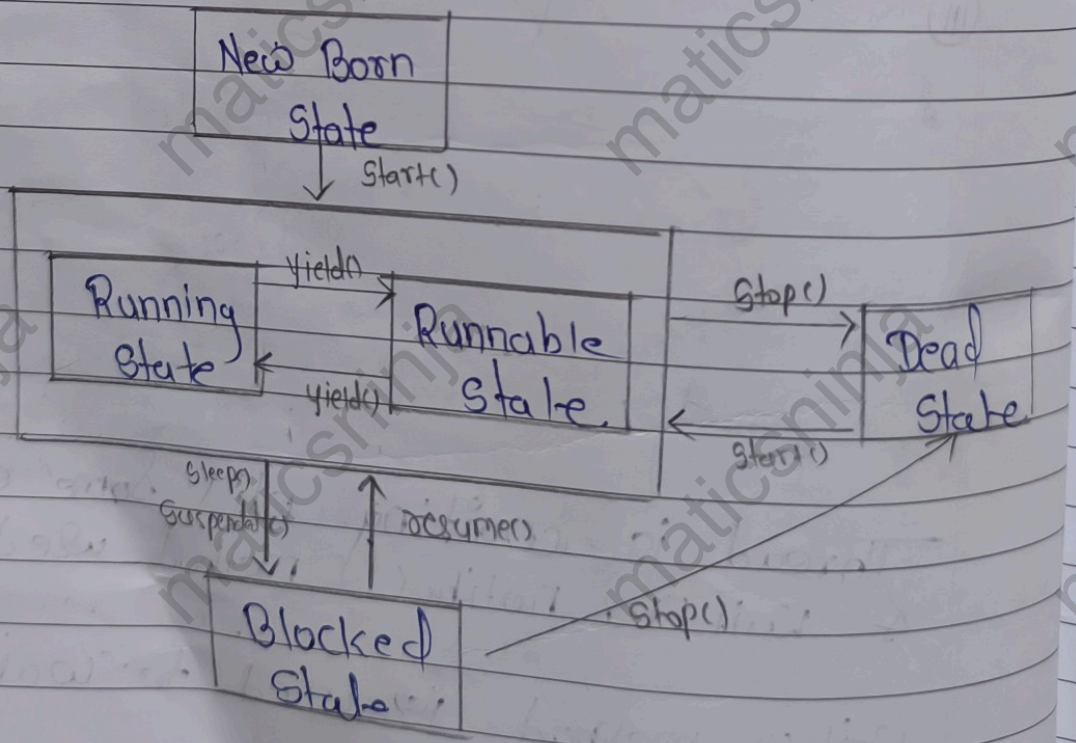


Fig. Life cycle of thread.

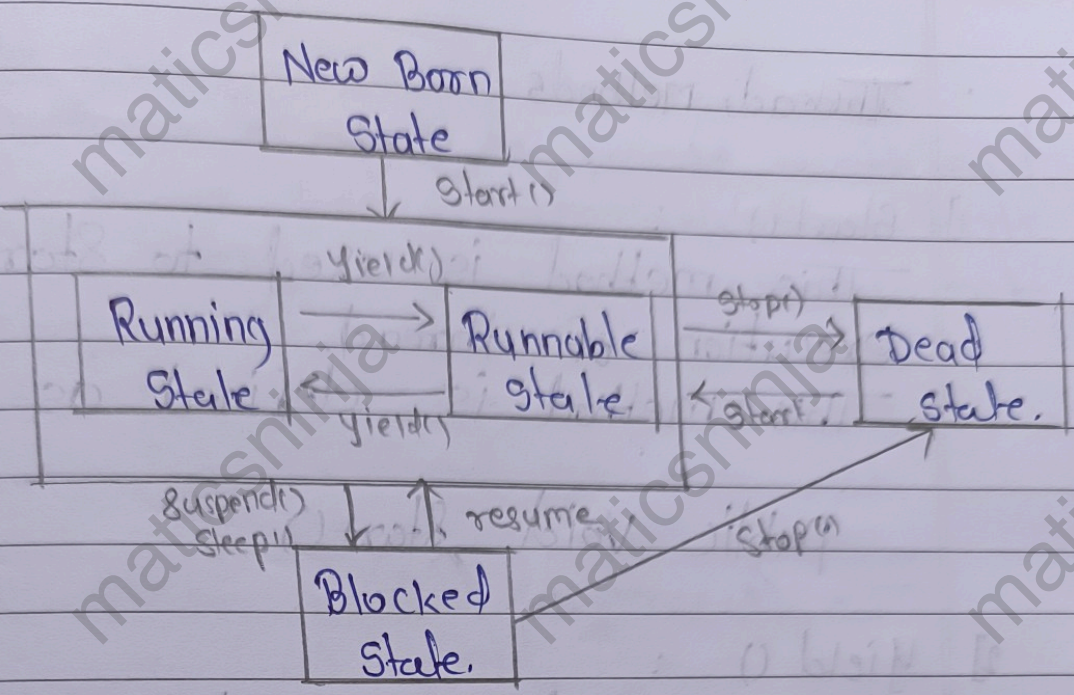


Fig. Life cycle of threads.

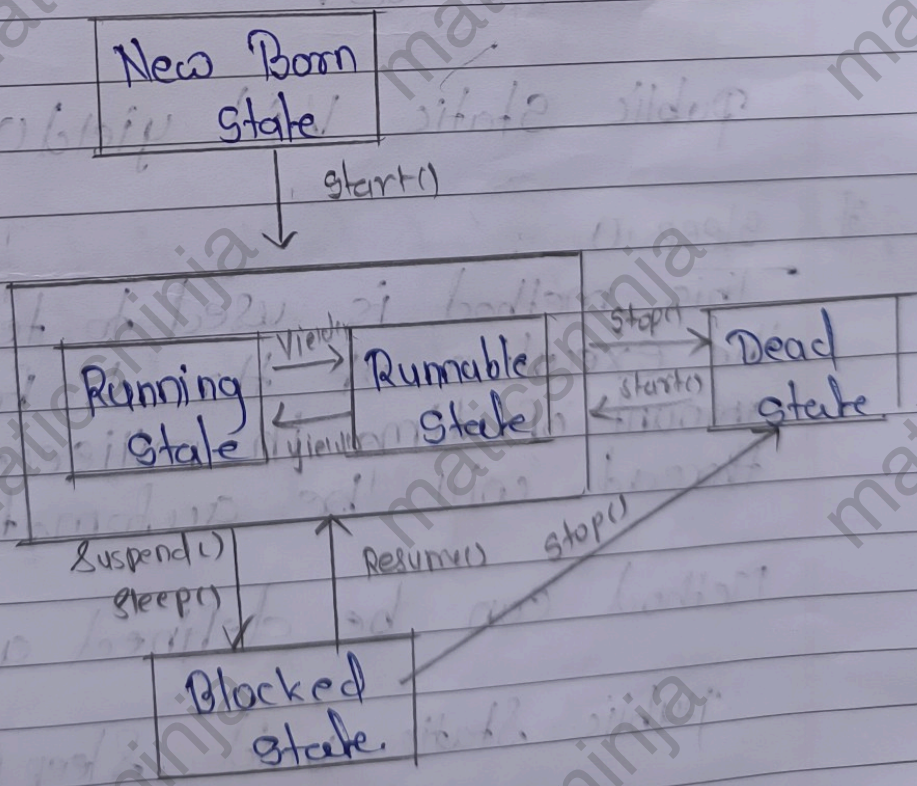


Fig. Life cycle of threads

Thread methods

1] Start() :
This method is used to start the execution of thread.
This method is defined as

```
public void start()
```

2] yield() :
This method is used to transfer the control from running to runable state. This method will() temproperly skip its current turn. This method is defined as:

```
Public Static Void yield()
```

3] Sleep() :
This method is used to temproperly pause the execution of thread for specified amount of time in milisecond. After time thread will be automatically Resume.
Method can be defined as:

```
Public Static void Sleep(t)
```

4] Suspend () :
This method is used to pause a thread but it will not automatically starts on By using resume method It will start.

It can be defined as:

public final void Suspend()

5] Resume () :
This method is used to resume a Suspended thread.
This method is defined as

public final void resume()

6] Stop () :
This method is used to manually stop the execution of thread
This method is defined as:

public final void stop()

Thread class also has some other method which help in managing Threads

1) getName()
It obtain the name of thread

2) SetName()
It Sets the name of thread

3) isAlive()
It returns whether thread is still running or not.

Thread priority

priority decides the execution of time of thread means at what time the thread is to be executed.

Thread is assign a priority which affect the order in which it is scheduled for execution.

Java assign priority to each thread to determine How thread should be treated. Higher priority thread gets more cpu time than low priority threads.

There are two methods

1] get priority()
2] Set priority()

1] It will return the priority
2] We can set the priority

minimum priority = 1
normal / medium priority = 6
maximum priority = 10.

- Thread Synchronization.

Synchronization is the process of assuring that the resource will be used by only one thread at a time

Java provides a keyword Synchronizaed.

- Thread exception

Mostly call to Sleep method is enclosed within try block and followed by catch block

If exeception thread is not Catch then Program will not compile and it will throw Illegal threadStateException