**THREAD :**

package day5;

import java.util.\*;

public class task1 {

public static void main(String args[])

{

System.***out***.println(Thread.*currentThread*());

System.***out***.println(Thread.*currentThread*().getName());

}

}

o/p :

Thread[#1,main,5,main]

main

**properties :**

System.***out***.println(Thread.*currentThread*().getPriority());

5

Thread.*currentThread*().setName("hari");

System.***out***.println(Thread.*currentThread*().getName());

Hari

Thread.*currentThread*().setPriority(10);

System.***out***.println(Thread.*currentThread*().getPriority());

10

Thread.*currentThread*().setPriority(10);

System.***out***.println(Thread.*currentThread*().getPriority());

Thread.*currentThread*().setPriority(118);

Exception in thread "hari" java.lang.IllegalArgumentException

System.***out***.println(Thread.*currentThread*().~~getId~~());

Thread.*currentThread*().setId(12);

1. 1

2. The method setId(int) is undefined for the type Thread (error)

The main thread is responsible for running the entire code by default.

But the thread can be created by using

1. Extending the thread class
2. Implement the runnable interface

package day5;

class First extends Thread

{

public void run()

{

for(int i=0;i<5;i++)

{

System.***out***.println(Thread.*currentThread*().getName());

}

}

}

public class task3 {

public static void main(String args[])

{

System.***out***.println(Thread.*currentThread*());

First f = new First();

f.run();

Thread[#1,main,5,main]

main

main

main

main

main

when an obj the call method by using the method name run() then the run() method is treated as a normal method

if the run method has to be trigged has an overrider method of the tread class then start you have to use the start() method to call the void run

package day5;

class First extends Thread

{

public void run()

{

for(int i=0;i<5;i++)

{

System.***out***.println(Thread.*currentThread*().getName());

}

}

}

public class task3 {

public static void main(String args[])

{

System.***out***.println(Thread.*currentThread*());

First f = new First();

f.start(); // class thread is run

//f.run(); main thread is run because the run() is treated as normal method

Thread[#1,main,5,main]

Thread-0

Thread-0

Thread-0

Thread-0

Thread-0

The class first run() method since if a normal function call thus the output of current thread is still main .

Where as the running method is call JVM create the new thread and thus a different thread name printed .

f.start();

f.start();

java.lang.IllegalThreadStateException

extends thread

class First extends Thread

{

public void run()

{

for(int i=0;i<5;i++)

{

System.***out***.println(Thread.*currentThread*().getName());

}

}

}

public class task3 {

public static void main(String args[])

{

System.***out***.println(Thread.*currentThread*());

First f = new First();

f.start();

f.run();

Thread[#1,main,5,main]

main

main

main

main

main

Thread-0

Thread-0

Thread-0

Thread-0

Thread-0

IMPLEMENTING THE TREAD USING RUNNABLE

package day5;

class Second1 implements Runnable

{

public void run()

{

for(int i=0;i<5;i++)

{

System.***out***.println(Thread.*currentThread*().getName());

}

}

}

public class task4 {

public static void main(String args[])

{

Second1 s = new Second1();

Thread th = new Thread(s);

th.start();

Second1 s1 = new Second1();

Thread th1 = new Thread(s1,"hari");

th1.start();

}

}

Thread-0

Thread-0

Thread-0

Thread-0

Thread-0

hari

hari

hari

hari

hari

package day5;

class Second2 implements Runnable{

public void run()

{

for(int i=0;i<15;i++)

{

System.***out***.println(Thread.*currentThread*().getName()+" "+i);

try {

Thread.*sleep*(1000);

}

catch(InterruptedException e) {

System.***out***.println(e);

}

}

}

}

public class task6 {

public static void main(String args[])

{

Second2 s = new Second2();

Thread th = new Thread(s,"hari");

th.start();

Second2 s1 = new Second2();

Thread th1 = new Thread(s1);

th1.start();

Second2 s2 = new Second2();

Thread th2 = new Thread(s2,"haran");

th2.start();

}

}

When a normal state of thread is interrupted by any means ex: sleep above throws InterruptedException .

Hence when we have to pass the running state thread it is recommended to handle InterrupetedException

haran 0

hari 0

Thread-0 0

hari 1

Thread-0 1

haran 1

Thread-0 2

haran 2

hari 2

hari 3

haran 3

Thread-0 3

YIELD METHOD

package day5;

class ThreadClass implements Runnable

{

public void run()

{

for(int i=1;i<5;i++)

{

System.***out***.println(Thread.*currentThread*().getName());

//System.out.println(Thread.currentThread().getPriority());

Thread.*yield*();

}

}

}

public class task7 {

public static void main(String args[])

{

ThreadClass th = new ThreadClass ();

Thread t1= new Thread(th,"T1");

t1.start();

System.***out***.println(t1.getPriority());

ThreadClass th1 = new ThreadClass ();

Thread t2= new Thread(th1,"T2");

t2.start();

System.***out***.println(t2.getPriority());

t2.setPriority(10);

}

}

5

T1

5

T2

T2

T2

T2

T1

T1

T1

THREAD SYNCHORONIZATION :

class Trainner{

void printTable(int n) {

for(int i=1;i<=10;i++)

{

System.***out***.println(i + "\*" +n + "=" +(i\*n));

}

}

}

class Ece extends Thread{

Trainner t ;

public Ece(Trainner t)

{

this.t=t;

}

public void run()

{

t.printTable(6);

}

}

class Cse extends Thread{

Trainner t ;

public Cse(Trainner t)

{

this.t=t;

}

public void run()

{

t.printTable(5);

}

}

public class task8 {

public static void main(String args[])

{

Trainner t = new Trainner();

Cse c = new Cse(t);

Ece e = new Ece(t);

c.start();

e.start();

}

}

1\*6=6

2\*6=12

3\*6=18

1\*5=5

4\*6=24

2\*5=10

5\*6=30

3\*5=15

6\*6=36 the 5th and 6th table is mixed

4\*5=20 because the cause of synchronization

7\*6=42

5\*5=25

8\*6=48

6\*5=30

9\*6=54

7\*5=35

10\*6=60

8\*5=40

9\*5=45

10\*5=50

class Trainner{

🡪 synchronized void printTable(int n) {

for(int i=1;i<=10;i++)

{

System.***out***.println(i + "\*" +n + "=" +(i\*n));

}

}

}

class Ece extends Thread{

Trainner t ;

public Ece(Trainner t)

{

this.t=t;

}

public void run()

{

t.printTable(6);

}

}

class Cse extends Thread{

Trainner t ;

public Cse(Trainner t)

{

this.t=t;

}

public void run()

{

t.printTable(5);

}

}

public class task8 {

public static void main(String args[])

{

Trainner t = new Trainner();

Cse c = new Cse(t);

Ece e = new Ece(t);

c.start();

e.start();

}

1\*5=5

2\*5=10

3\*5=15

4\*5=20 this is called **THREAD SAFETY**

5\*5=25

6\*5=30

7\*5=35 the 5th and 6th table is printed based on

8\*5=40 priority

9\*5=45

10\*5=50

1\*6=6

2\*6=12

3\*6=18

4\*6=24

5\*6=30

6\*6=36

7\*6=42

8\*6=48

9\*6=54

10\*6=60

Sychro method at thread safe method that are used to manage the access of a shared resource without thread switching the above code the method printable is a common method bot the thread of CSE and ECE hence the printable method should be synchro to avoid thread swapping .

**Producer Consumer problem :**package day5;

class Stock{

int n;

public void set(int n) {

System.***out***.println("set :"+n);

this.n=n;

}

public void get() {

System.***out***.println("get :"+n);

}

}

class Producer implements Runnable {

Stock s;

int i=0;

public Producer(Stock s) {

this.s=s;

Thread t1=new Thread(this,"producer"); // convert the producer obj into thread-"producer"

t1.start();

}

public void run() {

int i=0;

while(true)

{

s.set(i++);

try {

Thread.*sleep*(1000);

}

catch(InterruptedException e) {

System.***out***.println(e);

}

}

}

}

class Consumer implements Runnable{

Stock s;

public Consumer(Stock s){

this.s=s;

Thread t2 = new Thread(this,"consumer");

t2.start();

}

public void run() {

while(true)

{

s.get();

try {

Thread.*sleep*(1000);

}

catch(InterruptedException ee) {

System.***out***.println(ee);

}

}

}

}

public class task9 {

public static void main(String args[]) {

Stock s = new Stock();

Producer p = new Producer(s);

Consumer c = new Consumer(s);

}

}

The above code the producer and consumer are not in sinks does either the producer keeps on producing the and puts the consumer in the starvation.

package day5;

class Stock{

int n;

boolean status = false;

synchronized public void set(int n) {

System.***out***.println("set :"+n);

this.n=n;

while(status)

{

try {

wait();

}

catch(Exception e)

{

System.***out***.println(e);

}

status=true;

notify();

}

}

synchronized public void get() {

while(!status)

{

try {

wait();

}

catch(Exception e)

{

System.***out***.println(e);

}

status=false;

notify();

}

System.***out***.println("get :"+n);

}

}

class Producer implements Runnable {

Stock s;

int i=0;

public Producer(Stock s) {

this.s=s;

Thread t1=new Thread(this,"producer"); // convert the producer obj into thread-"producer"

t1.start();

}

public void run() {

int i=0;

while(true)

{

s.set(i++);

try {

Thread.*sleep*(1000);

}

catch(InterruptedException e) {

System.***out***.println(e);

}

}

}

}

class Consumer implements Runnable{

Stock s;

public Consumer(Stock s){

this.s=s;

Thread t2 = new Thread(this,"consumer");

t2.start();

}

public void run() {

while(true)

{

s.get();

try {

Thread.*sleep*(1000);

}

catch(InterruptedException ee) {

System.***out***.println(ee);

}

}

}

}

public class task9 {

public static void main(String args[]) {

Stock s = new Stock();

Producer p = new Producer(s);

Consumer c = new Consumer(s);

}

}

set :0

set :1

set :2

set :3

set :4

set :5

set :6

set :7

set :8

set :9

set :10

set :11

set :12

set :13

set :14

set :15

set :16

set :17

set :18

set :19

set :20

set :21

set :22

Join :

Thread :  
Join

package day6;

class TaskThread extends Thread{

private String ThreadName;

public TaskThread(String ThreadName) {

this.ThreadName=ThreadName;

}

public void run() {

for(int i=0;i<3;i++)

{

System.***out***.println(this.ThreadName);

try {

Thread.*sleep*(1000);

}

catch(InterruptedException e)

{

System.***out***.println(e);

}

System.***out***.println("all finished");

}

}

}

public class task1 {

public static void main(String args[])

{

TaskThread t1 = new TaskThread("Thread1");

TaskThread t2 = new TaskThread("Thread2");

TaskThread t3 = new TaskThread("Thread3");

t1.start();

try {

t1.join(); // the main block is wait until the t1 all execution are finished

}

catch(InterruptedException e){

System.***out***.println(e);

}

t2.start();

try {

t2.join(); // the main block is wait until the t2 all execution are finished

}

catch(InterruptedException e){

System.***out***.println(e);

}

t3.start();

try {

t3.join(); // the main block is wait until the t3 all execution are finished

}

catch(InterruptedException e){

System.***out***.println(e);

}

System.***out***.println("In main block :"+Thread.*currentThread*().getName());

}

}

Thread1

all finished

Thread1

all finished

Thread1

all finished

Thread2

all finished

Thread2

all finished

Thread2

all finished

Thread3

all finished

Thread3

all finished

Thread3

all finished

In main block :main