DAY-59

------

MULTI-THREADING-2

-------------------------------------

REFER fig:1

There are 2 levels of multi-tasking:

-------------------------------------

1. Multi-processing/ process based multi-tasking

--> It is a process of executing several task simultaneoulsy where each task is a separate independent process.

2. Multi-threading / thread based multi-tasking.

--> It is a process of executing several threads simultaneoulsy where each thread is a separate independent, but all the threads are part of a task.

OVER LOADING OF RUN METHOD:

----------------------------

// example-1

// overloading of run method

class MyThread extends Thread

{

public void run()

{

System.out.println("no-parameter");

}

public void run( int i )

{

System.out.println("one-parameter");

}

}

class Demo

{

public static void main(String[] args)

{

MyThread t1 = new MyThread();

t1.start();

t1.run(5);

System.out.println("main method");

}

}

OUTPUT:

--------

one-parameter

no-parameter

main method

/\* NOTE : Method overloading is applicable on the run method which is present in the thread class.

but, whenever start method is invoked with the help of thread ref variable [t.start();] then by default zero-parameterized

run method will be called.

if parameterized run method is present then we must call it explicitly.

\*/

IMPORTANCE OF RUN METHOD:

--------------------------

// example-2

class MyThread extends Thread

{

}

class Demo1

{

public static void main(String[] args)

{

MyThread t1 = new MyThread();

t1.start();

System.out.println("main method");

}

}

NOTE : If we want to make use of concept of multi-threading then we must override run method.

In the above example the new thread is not activated.

OVERRIDING START METHOD:

------------------------

// example-3

// override of start method.

class MyThread extends Thread

{

public void start()

{

System.out.println("start method");

}

public void run()

{

System.out.println("run method");

}

}

class Demo2

{

public static void main(String[] args)

{

MyThread t1 = new MyThread();

t1.start();

System.out.println("main method");

}

}

/\* OUTPUT

start method

main method

If we over ride start method then overrided start method will execute instead of default start method.

In the above example new thread is not created.

\*/

// example-3.1

// override of start method.

class MyThread extends Thread

{

public void start()

{

super.start();

System.out.println("start method");

}

public void run()

{

System.out.println("run method");

}

}

class Demo2

{

public static void main(String[] args)

{

MyThread t1 = new MyThread();

t1.start();

System.out.println("main method");

}

}

/\* OUTPUT

start method

run method

main method

If we want to call default start method inspite of having overriden start method then we must make use of

super method[calling parent constructor]

\*/

RE-STARTING THE THREAD:

--------------------------

// example-4

class MyThread extends Thread

{

public void run()

{

System.out.println("run method");

}

}

class Demo3

{

public static void main(String[] args)

{

MyThread t1 = new MyThread();

t1.start();

System.out.println("main method");

t1.start();

}

}

/\*

OUTPUT:

---------

main method

run method

Exception in thread "main" java.lang.IllegalThreadStateException

at java.lang.Thread.start(Thread.java:708)

at Demo3.main(Demo3.java:16)

NOTE: Once the thread is started we can not re-start the thread.

GETTING THE THREAD NAMES:

SETTING AND GETTING THE THREAD NAMES:

--------------------------------------

// example-5

class MyThread extends Thread

{

public void run()

{

System.out.println("child thread");

}

}

class Demo4

{

public static void main(String[] args)

{

System.out.println("main method starts..!");

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

MyThread t1 = new MyThread();

t1.start();

System.out.println(t1.getName());

Thread.currentThread().setName("sagar");

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

t1.setName("sandesh");

System.out.println(t1.getName());

System.out.println(10/0);

}

}

/\*

OUTPUT:

----------

main method starts..!

main

Thread-0

sagar

child thread

sandesh

Exception in thread "sagar" java.lang.ArithmeticException: / by zero

at Demo4.main(Demo4.java:27)

// example-6

-------------

class MyThread extends Thread

{

public void run()

{

System.out.println("run method is executed by : "+Thread.currentThread().getName());

}

}

class Demo5

{

public static void main(String[] args)

{

MyThread t1 = new MyThread();

t1.start();

System.out.println("main method is executed by: "+Thread.currentThread().getName());

}

}

/\*

OUTPUT:

----------

main method is executed by: main

run method is executed by : Thread-0

CREATION OF THREADS BY IMPLEMENTING RUNNABLE INTERFACE:

----------------------------------------------------------

// example-7

-------------

class MyThread1 implements Runnable

{

public void run()

{

try

{

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

{

Thread.sleep(2000);

System.out.println("sq\_num: "+(i\*i));

}

}

catch (InterruptedException e)

{

}

}

}

class MyThread2 implements Runnable

{

public void run()

{

try

{

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

{

Thread.sleep(2000);

System.out.println("db\_num: "+(i\*2));

}

}

catch (InterruptedException e)

{

}

}

}

class Demo6

{

public static void main(String[] args)

{

System.out.println("main method");

MyThread1 r1 = new MyThread1();

MyThread2 r2 = new MyThread2();

Thread t1 = new Thread(r1);

Thread t2 = new Thread(r2);

t1.start();

t2.start();

}

}

/\*

OUTPUT:

------------

main method

sq\_num: 1

db\_num: 2

db\_num: 4

sq\_num: 4

db\_num: 6

sq\_num: 9

db\_num: 8

sq\_num: 16

db\_num: 10

sq\_num: 25

NOTE: Runnable interface dont have the start method to start the thread hence we need the help of thread class.

For thread class constructor pass the runnable object ref as argument.

example-8

-----------

class MyThread extends Thread

{

public void run()

{

System.out.println("child thread");

}

}

class Demo7

{

public static void main(String[] args)

{

System.out.println("main method");

MyThread mt = new MyThread();

Thread t1 = new Thread(mt);

mt.start();

}

}

/\*

OUTPUT:

--------

main method

child thread

example-9

-----------

class MyThread extends Thread

{

public void run()

{

System.out.println("child thread");

}

}

class Demo7

{

public static void main(String[] args)

{

System.out.println("main method");

MyThread mt = new MyThread();

Thread t1 = new Thread();

Thread t2 = new Thread(mt);

t1.start();// child thread not created

t2.start();// child thread is created.

t1.run();// run method as not implementation

t2.run();// run method is executed

mt.run();// child thread is not created ... run method is executing as normal method

In this case thread is not created...!

}

}