Scala Thread Methods

Thread class provides various methods to deals with thread's states. You can use these methods to control the flow of thread.

The following table contains commonly used methods of Thread class.

Method	Description
public final String getName()	It returns thread's name.
public final int getPriority()	**It returns thread's priority.**
public Thread.State getState()	It returns the state of this thread. This method is designed for use in monitoring of the system state, not for synchronization control.
public final boolean isAlive()	It tests if this thread is alive. A thread is alive if it has been started and has not yet died.
public final void join() throws InterruptedException	It Waits for thread to die.
public void run()	If this thread was constructed using a separate Runnable run object, then that Runnable object's run method is called; otherwise, this method does nothing and returns.
public final void setName(String name)	It is used to set thread name.
public final void setPriority(int newPriority)	It is used to set priority of a thread.
public static void sleep(long millis) throws InterruptedException	It is used to sleep executing thread for the specified number of milliseconds.
public static void yield()	It causes the currently executing thread object to temporarily pause and allow other threads to execute.

Scala Thread sleep() Method

The sleep() method is used to sleep thread for the specified time. It takes time in milliseconds as an argument.

```
class ThreadExample extends Thread{

override def run(){

for(i<- 0 to 5){
```

```
println(i)
Thread.sleep(500)
}
}
}
object MainObject{
def main(args:Array[String]){
var t1 = new ThreadExample()
var t2 = new ThreadExample()
t1.start()
t2.start()
}
}
```

Output:

```
0
0
1
1
2
2
3
3
4
4
5
5
```

Scala Thread join() Method Example

The join() method waits for a thread to die. In other words, The join() method is used to hold the execution of currently running thread until the specified thread finished it's execution.

```
class ThreadExample extends Thread{

override def run(){

for(i<- 0 to 5){

println(i)

Thread.sleep(500)

}

}

}

object MainObject{

def main(args:Array[String]){

var t1 = new ThreadExample()

var t2 = new ThreadExample()

var t3 = new ThreadExample()

t1.start()

t1.join()

t2.start()

t3.start()

}

}
```

Output:

```
0

1

2
```

3

4

5

0

0

1

1

2

2

3

3

4

4

5

5

Scala setName() Method Example

In the following example, we are setting and getting names of threads.

```
class ThreadExample() extends Thread{

override def run(){

for(i<- 0 to 5){

println(this.getName()+" - "+i)

Thread.sleep(500)

}

}

}
```

```
object MainObject{

def main(args:Array[String]){

var t1 = new ThreadExample()

var t2 = new ThreadExample()

var t3 = new ThreadExample()

t1.setName("First Thread")

t2.setName("Second Thread")

t1.start()

t2.start()

}

}
```

Output:

```
First Thread - 0

Second Thread - 0

Second Thread - 1

First Thread - 1

Second Thread - 2

First Thread - 2

Second Thread - 3

First Thread - 3

Second Thread - 4

First Thread - 4

Second Thread - 5

First Thread - 5
```

Scala Thread Priority Example

You can set thread priority by using it's predefined method. The following example sets priority for the thread.

```
class ThreadExample() extends Thread{

override def run(){

for(i<- 0 to 5){

println(this.getName())

println(this.getPriority())

Thread.sleep(500)

}

}

}

object MainObject{

def main(args:Array[String]){

var t1 = new ThreadExample()

var t2 = new ThreadExample()

t1.setName("First Thread")

t2.setName("Second Thread")

t1.setPriority(Thread.MIN_PRIORITY)

t2.setPriority(Thread.MAX_PRIORITY)

t1.start()

t2.start()

}

}
```

Output:

First Thread

Second Thread

10

1

Second Thread

10

First Thread

1

Second Thread

10

First Thread

1

Second Thread

10

First Thread

1

Second Thread

10

First Thread

1

Second Thread

10

First Thread

1

Scala Thread Multitasking Example

The following example is running multiple tasks by using multiple threads. This example

explains that how can we implement multitasking in Scala.

```
class ThreadExample() extends Thread{
override def run(){
for(i<- 0 to 5){
println(i)
Thread.sleep(500)
}
}
def task(){
for(i<- 0 to 5){
println(i)
Thread.sleep(200)
}
}
}

object MainObject{
def main(args:Array[String]){
var t1 = new ThreadExample()
t1.start()
t1.task()
}
}
```

Output:

```
0
0
```

1

2

1

3

4

2

5

3

4

5