SNS COLLEGE OF ENGINEERING

Kurumbapalayam(Po), Coimbatore – 641 107
Accredited by NAAC-UGC with 'A' Grade
Approved by AICTE, Recognized by UGC & Affiliated to Anna University, Chennai

Department of Information Technology

19CS204 OBJECT ORIENTED PROGRAMMING

I YEAR /II SEMESTER

Topic – Thread Methods

Thread methods

- ✓ In Java, threads are a way to achieve **concurrent execution of code**.
- ✓ Each thread **represents an independent flow of control within a program**.
- ✓ Java provides several methods and functionalities to work with threads through the Thread class and other related classes in the java.lang package.
- ✓ some commonly used methods of the Thread class in Java:
- ✓ **start()**: This method is used to **start the execution of a thread**. When called, it invokes the **thread's run() method**.
- ✓ **run()**: This method **contains the code that is executed by the thread**. It is the entry point of the thread's execution logic. You need to override this method in a custom class that extends Thread.

Thread methods

sleep(long millis): This method **pauses the execution of the current thread for the specified number of milliseconds.** It is commonly used to introduce delays or control timing in multithreaded programs.

join(): This method allows **one thread to wait for the completion of another thread.** When a thread calls join() on another thread, it will block until that thread finishes its execution.

isAlive(): This method checks **whether a thread is still active or alive**. It returns **true if the thread is alive, and false otherwise**.

Thread methods

interrupt(): This method **interrupts the execution of a thread**. It sets the interrupt status of the thread, which can be checked using the isInterrupted() method.

getName() and **setName(String name)**: These methods allow **getting and setting the name of a thread, respectively**. The thread's name can be helpful for **identification and debugging purposes.**

Thread methods

Example – isAlive()

```
public class JavaIsAliveExp extends Thread
{
    public void run()
    {
        try
        {
            Thread.sleep(300);
            System.out.println("is run() method isAlive "+Thread.currentThread().isAlive());
        }
        catch (InterruptedException ie) {
        }
    }
    public static void main(String[] args)
    {
        JavaIsAliveExp t1 = new JavaIsAliveExp();
        System.out.println("before starting thread isAlive: "+t1.isAlive());
        t1.start();
        System.out.println("after starting thread isAlive: "+t1.isAlive());
    }
}
```

Thread methods

Example – join()

```
public class TestJoinMethod1 extends Thread{
 public void run(){
  for(int i=1;i<=5;i++){
   try{
    Thread.sleep(500);
   }catch(Exception e){System.out.println(e);}
  System.out.println(i);
  }   }
public static void main(String args[]){
 TestJoinMethod1 t1=new TestJoinMethod1();
 TestJoinMethod1 t2=new TestJoinMethod1();
 TestJoinMethod1 t3=new TestJoinMethod1();
 t1.start();
 try{
  t1.join();
 }catch(Exception e){System.out.println(e);}
  t2.start();
  /*try{
  t2.join();
 }catch(Exception e){System.out.println(e);} */
 t3.start();
 }  }
```

Thread methods

Example : isAlive() and join()

```
class NewThread implements Runnable {
String name; // name of thread
Thread t;
NewThread(String threadname) {
name = threadname;
t = new Thread(this, name);
System.out.println("New thread: " + t);
t.start(); // Start the thread
}
// This is the entry point for thread.
public void run() {
try {
for(int i = 5; i > 0; i--) {
System.out.println(name + ": " + i);
Thread.sleep(1000);
}
} catch (InterruptedException e) {
System.out.println(name + " interrupted.");
}
System.out.println(name + " exiting.");
}
}
```

```
class DemoJoin {
public static void main(String args[]) {
NewThread ob1 = new NewThread("One");
NewThread ob2 = new NewThread("Two");
NewThread ob3 = new NewThread("Three");
System.out.println("Thread One is alive: " + ob1.t.isAlive());
System.out.println("Thread Two is alive: " + ob2.t.isAlive());
System.out.println("Thread Three is alive: " + ob3.t.isAlive());
// wait for threads to finish
try {
System.out.println("Waiting for threads to finish.");
ob1.t.join();
ob2.t.join();
ob3.t.join();
} catch (InterruptedException e) {
System.out.println("Main thread Interrupted");
}
System.out.println("Thread One is alive: " + ob1.t.isAlive());
System.out.println("Thread Two is alive: " + ob2.t.isAlive());
System.out.println("Thread Three is alive: " + ob3.t.isAlive());
System.out.println("Main thread exiting.");
} }
```

Thread methods

Thread Priorities

- Thread priorities are used by the thread scheduler to decide when each thread should be allowed to run.

- In theory, over a given period of time, higher-priority threads get more CPU time than lower-priority threads.

- **Thread priority decides when to switch from one running thread to another, process is called context switching**

- A higher-priority thread can also preempt a lower-priority one.

- To set a thread's priority, use the setPriority( ) method, which is a member of Thread.
  This is its general form:
  final void setPriority(int level)

- Here, level specifies the new priority setting for the calling thread.

Thread methods

Thread Priorities

- In place of defining the priority in integers, we can use MIN_PRIORITY, NORM_PRIORITY or MAX_PRIORITY.
- The value of level must be within the range MIN_PRIORITY and MAX_PRIORITY.
- Currently, these values are 1 and 10, respectively.
- To return a thread to default priority, specify NORM_PRIORITY, which is currently 5.
- These priorities are defined as static final variables within Thread. For example
  - public static int MIN_PRIORITY
- You can obtain the current priority setting by calling the getPriority( ) method of Thread, shown here:

  final int getPriority( )

Thread methods

Thread Priorities

```
public class TestMultiPriority1 extends Thread{
 public void run(){
   System.out.println("running thread name is:"+Thread.currentThread().getName());
   System.out.println("running thread priority is:"+Thread.currentThread().getPriority());

  }
 public static void main(String args[]){
  TestMultiPriority1 m1=new TestMultiPriority1();
  TestMultiPriority1 m2=new TestMultiPriority1();
  TestMultiPriority1 m3=new TestMultiPriority1();
  System.out.println("Default Priority: "+Thread.currentThread().getPriority());
  m1.setPriority(Thread.MIN_PRIORITY);
  m2.setPriority(Thread.MAX_PRIORITY);
  m3.setPriority(Thread.NORM_PRIORITY);
  m1.start();
  m2.start();
  m3.start();
 }
}
```

THANK YOU