**Multithreading 🡪 part 1**

Executing multiple threads (sub process, small task) at single time

Music playing.

Updating time.

Animation is running

It is implemented in games

Updating score

Directions

Element movement

**Difference between process and thread**

1. Process is a program which is in executing state where thread is a subpart of the process
2. Process is a heavy weight where thread is of light weight.

**2 ways to create a thread**

* **By extending Thread class**
* **Implementing Runnable interface**

**Note:** thread class present in java.lang package

Main thread is the default thread created by jvm

1. **By extending Thread class**

**package thread;**

**public class Test extends Thread{**

**@Override**

**public void run() {**

**}**

**public static void main(String[] args) {**

**Test t =new Test();**

**t.start();**

**}**

**}**

**Thread life cycle**

**1. new state:**

**we are creating the object of test class**

**2. runnable state:**

**new thread reaches runnable state as we call start method**

**3. Running state:**

**when we say t.start(); it will start executing the run method**

**the implementation of start method is done in such a way that it will**

**start executing the thread**

**after runnable stage Jvm will allocate the processor for thread and**

**thread scheduler will decide which thread to be executed based on**

**the algorithm**

**now thread is start executed**

**4. dead state:**

**thread is complete its task**

**2.5 Non runnable state**

**between runnable and running state thread can be in non-runnable state.**

**waiting or sleeping state.**

**after sometime it may come to runnable state and then it move forward to**

**running state.**

**package java.lang;**

**class Thread implementing Runnable{**

**// constructors**

**// methods**

**start()**

**run()**

**sleep()**

**}**

**public interface Runnable{**

**public void run();**

**}**

**We cant invoke same thread multiple times**

**package thread;**

**public class Test extends Thread{**

**@Override**

**public void run() {**

**System.*out*.println("Thread is running");**

**}**

**public static void main(String[] args) {**

**Test t =new Test();**

**t.start();**

**// t.start();// we cant invoke thread multiple times**

**// we are getting java.lang.IllegalThreadStateException**

**}**

**}**

**Implementing runnable interface**

**package thread;**

**public class Test implements Runnable{**

**@Override**

**public void run() {**

**System.*out*.println("Thread is running");**

**}**

**public static void main(String[] args) {**

**Test t =new Test();**

**// start method is present in thread class so we need to**

**// create an object of thread class and pass test class reference into**

**// the thread class object**

**Thread th = new Thread(t);**

**th.start();**

**}**

**}**

Which is the efficient way of creating the thread class

By implementing the runnable interface

**Reason**:

In programming we are using inheritance concept. If in case if a class is already inheriting from a class when it tried to inherit thread class, there we come across the concept of **multiple inheritance**.

We know that one class can inherit a class and implement n number of interfaces. So, implementing interface is the best way of creating the thread.

**Performing single task from single thread**

**package thread;**

**public class Test extends Thread{**

**@Override**

**public void run() {**

**System.out.println(“thread is running”);**

**}**

**public static void main(String[] args) {**

**Test t =new Test();**

**t.start();**

**}**

**}**

**Performing single task from Multiple thread**

**package thread;**

**public class Test extends Thread{**

**@Override**

**public void run() {**

**System.*out*.println("thread is running");**

**}**

**public static void main(String[] args) {**

**Test t1 =new Test();**

**t1.start();**

**Test t2 = new Test();**

**t2.start();**

**}**

**}**

**Performing Multiple task from single thread**

**This is not possible**

**Performing Multiple task from multiple thread**

**package thread;**

**public class Test extends Thread{**

**@Override**

**public void run() {**

**System.*out*.println("Test thread is running");**

**}**

**public static void main(String[] args) {**

**Test t1 =new Test();**

**t1.start();**

**thread1 th1=new thread1();**

**th1.start();**

**thread2 th2= new thread2();**

**th2.start();**

**}**

**}**

**class thread1 extends Thread{**

**@Override**

**public void run() {**

**System.*out*.println("thread1 is running");**

**}**

**}**

**class thread2 extends Thread{**

**@Override**

**public void run() {**

**System.*out*.println("thread2 is running");**

**}**

**}**

**// execution depends on algorithm and decided by jvm ie thread scheduler**

**//every thread executes simultaneously**

**Constructors and methods of thread**

1. **Thread() 🡪 default constructor**
2. **Thread(Runnable target) 🡪 thread th=new thread(t);**
3. **When we extending thread**

**Thread(String name) 🡪 when we create a thread by default provide name. if we want we can do it explicitly.**

1. **When we implementing runnable**

**Thread(Runnable target, String name);**

1. **Thread(Threadgroup tg , Runnable target)**
2. **Thread(Threadgroup tg , String name)**
3. **Thread(Threadgroup tg , Runnable target,String name)**
4. **Thread(Threadgroup tg , Runnable target,String name,long stacksize)**

**Methods**

1. **run()**
2. **start()**
3. **currentThread 🡪 static**
4. **isAlive() 🡪 thread is in execution state or not**
5. **getName(), setName(String name)**
6. **demon thread related**

**isDaemon() setDaemon(boolean b)**

**true – daemon thread created**

**false- not created**

1. **getPriority() setPriority(int priority)**
2. **preventing thread execution method**

**sleep() yield() join()**

**Thread interruption method**

**interrupt();**

**isInturrupted();**

**Inturrupted();**

**Inter thread communication**

**Wait() notify() notifyAll()**

**getName() and setName() method**

* **for main method**

**package thread;**

**public class Test{**

**public static void main(String[] args) {**

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

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

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

**// System.out.println(1/0);**

**//Exception in thread "tom"**

**}**

**}**

**package thread;**

**public class Test extends Thread{**

**public void run() {**

**// how to change threadname here**

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

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

**//Thread-0 default name assigned by jvm**

**System.*out*.println("running");**

**}**

**public static void main(String[] args) {**

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

**Test t= new Test();**

**t.start();**

**Test t2=new Test();**

**t2.start();**

**Test t3=new Test();**

**t3.setName("tom");**

**t3.start();**

**}**

**}**

**isAlive()package thread;**

**public class Test extends Thread{**

**public void run() {**

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

**}**

**/\***

**true**

**true**

**true**

**\*/**

**public static void main(String[] args) {**

**Test t= new Test();**

**t.start();**

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

**System.*out*.println(t.isAlive());**

**}**

**}**

**Daemon thread**

* it runs in the background of another thread
* it provide service to the thread

eg : garbage collector

it works in the background of the main thread

it resolve problems of main method like memory full it delete waste objects

ex: spelling checker in ms word

**Methods on demon thread**

**Public final void setDaemon(Boolean b){}**

**Public final Boolean isDaemon(){}**

**Note1: before starting the thread we need to create the daemon thread. Else exception illegal-thread-state**

**Note: we cant create main method as daemon thread because it is already started by JVM.**

**Daemon thread inherit properties from its parent**

**package thread;**

**public class Test extends Thread{**

**public void run() {**

**if(Thread.*currentThread*().isDaemon())**

**System.*out*.println("daemon thread");**

**else**

**System.*out*.println("childthread");**

**}**

**public static void main(String[] args) {**

**System.*out*.println("main thread");**

**Test t= new Test();**

**t.setDaemon(true);**

**t.start();**

**}**

**}**

**When main method is not doing any task it wont execute daemon thread. Because its purpose only to provide service to main thread**

**package thread;**

**public class Test extends Thread{**

**public void run() {**

**if(Thread.currentThread().isDaemon())**

**System.out.println("daemon thread");**

**else**

**System.out.println("childthread");**

**}**

**public static void main(String[] args) {**

**// System.out.println("main thread");**

**Test t= new Test();**

**t.setDaemon(true);**

**t.start();**

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

**}**

**}**

**We cant set main thread as daemon thread and we cant set thread as a daemon thread after its running**

**package thread;**

**public class Test extends Thread{**

**public void run() {**

**if(Thread.*currentThread*().isDaemon())**

**System.*out*.println("daemon thread");**

**else**

**System.*out*.println("childthread");**

**}**

**public static void main(String[] args) {**

**// Thread.currentThread().setDaemon(true);**

**Test t= new Test();**

**t.setDaemon(true);**

**t.start();**

**t.setDaemon(true);**

**}**

**}**