Task 6. Threading

* Creation of thread in Java applications
* Multithreading

| **Constructor/** | **Prototype** | **Description** |
| --- | --- | --- |
| **Thread () constructor** | Thread() | Default constructor to create a Thread object. |
|  | Thread(String name) | Constructor to create a Thread object with specified name. |
|  | Thread(Runnable r) | Create a Thread instance with specified Runnable interface object. |
|  | Thread(Runnable r,String name) | Create a Thread instance with specified Runnable interface object and given name |
| **run** | public void run() | Run method performs the action for a thread. Invokes the thread. |
| **start** | public void start() | Used to start the execution of the thread. Internally the JVM calls run () method on this thread. |
| **sleep** | public void sleep(long milliseconds) | The execution of the current thread is stopped for specified milliseconds. |
| **join** | public void join() | Wait for the thread to die |
| public void join(long milliseconds) | Wait for specified milliseconds for the thread to die. |
| **getPriority** | public int getPriority() | Return the thread priority |
| **setPriority** | public int setPriority(int priority) | Change the thread priority to specified priority |
| **getName** | public String getName() | return the name of the thread. |
| **setName** | public void setName(String name) | Set the name of the thread to specified string |
| **currentThread** | public Thread currentThread() | Returns the reference of the thread that is currently active |
| **getId** | public int getId() | Return thread Id |
| **getState()** | public Thread.State getState() | Returns current state of the thread |
| **isAlive** | public boolean isAlive() | Check if the thread is alive and return true if yes. |
| **yield** | public void yield() | Temporarily pauses the current thread and allows other threads to execute. |
| **isDaemon** | public boolean isDaemon() | Check if the thread is a daemon thread; return true if yes. |
| **setDaemon** | public void setDaemon(boolean b) | Set the thread as a daemon thread if b=true; else set as user thread. |
| **interrupt** | public void interrupt() | Interrupt the current thread. |
| **isInterrupted** | public boolean isInterrupted() | Check if thread is interrupted. |
| **interrupted** | public static boolean interrupted() | Check if current thread has been interrupted. |
| **dumpStack** | Static void dumpStack() | Prints a stack trace of the current thread to the standard error stream. |
| **suspend** | public void suspend() | Suspends all threads. (\*\* method is deprecated in latest Java versions) |
| **resume** | public void resume() | Resume suspended thread. (\*\* method is deprecated in latest Java versions) |
| **stop** | public void stop() | Stops the thread. (\*\* method is deprecated in latest Java versions) |

6.1 Create a new thread class that inherits the Thread class and print the thread name.

**public** **class** CreateThread **extends** Thread{

**public** **void** run()

{

System.***out***.println("Thread has been Started");

}

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

{

CreateThread ct=**new** CreateThread();

ct.start();

}

}

6.2 Create a new thread class that implements the Runnable class and print the thread name.

**public** **class** CreateThread1 **implements** Runnable {

**public** **void** run()

{

System.***out***.println("Thread Started");

}

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

{

Thread t1=**new** Thread(**new** CreateThread1());

t1.start();

System.***out***.println("Name of thread is :"+t1.getName());

}

6.3 Create a new multithreading class that inherits the Thread class

//class inherited from "Thread"

**class** MultiThreading **extends** Thread {

**private** **int** number;

//class constructor

**public** MultiThreading(**int** number) {

**this**.number = number;

}

//run method => execution code for thread

**public** **void** run() {

**int** counter = 0;

**int** numInt = 0;

//prints the number till specified number is reached, starting from 10

**do** {

numInt = (**int**) (counter + 10);

System.***out***.println(**this**.getName() + " prints " + numInt);

counter++;

} **while**(numInt != number);

System.***out***.println("\*\* Correct! " + **this**.getName() + "printed " + counter + " times.\*\*");

}

**public** **class** Main {

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

System.***out***.println("Starting thread\_1...");

//create a thread class instance

Thread thread\_1 = **new** MultiThreading(15);

//start the thread thread\_1

thread\_1.start();

**try** {

//wait for thread\_1 to die

thread\_1.join();

} **catch** (InterruptedException e) {

System.***out***.println("Thread interrupted.");

}

System.***out***.println("Starting thread\_2...");

Thread thread\_2 = **new** MultiThreading(20);

//start thread\_2

thread\_2.start();

System.***out***.println("main() is ending...");

}

}

}

6.4 Create a new multithreading class that implements the Runnable Class

**class** MutliThreading1 **implements** Runnable {

**private** String message;

**public** MutliThreading1(String message) {

**this**.message = message;

}

**public** **void** run() {

**while**(**true**) {

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

}

}

**public** **class** Main {

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

Runnable hello = **new** MutliThreading1("Hello, Greetings!!!");

Thread thread1 = **new** Thread(hello);

thread1.setDaemon(**true**); //set this thread as daemon

thread1.setName("hello");

System.***out***.println("Starting First thread...");

thread1.start();

Runnable bye = **new** MutliThreading1("Bye for now!!");

Thread thread2 = **new** Thread(bye); thread2.setPriority(Thread.***MIN\_PRIORITY***); //set priority to min

thread2.setDaemon(**true**); //set as daemon thread

System.***out***.println("Starting goodbye thread...");

thread2.start();

System.***out***.println("main() is ending...");

}

}

}