1.       Write  
a java program that display thread object methods.  
**package** multithreading.assignments;

SANJANA KUMBHAR

**class** MyThread **extends** Thread {

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

**try** {

Thread.*sleep*(2000); // Simulate some work

} **catch** (InterruptedException e) {

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

}

}

}

**public** **class** ThreadMethodsDemo {

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

// Create and start a thread

MyThread thread = **new** MyThread();

thread.start();

// Show thread methods

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

System.***out***.println("Thread ID: " + thread.~~getId~~());

System.***out***.println("Thread is Alive: " + thread.isAlive());

System.***out***.println("Thread State: " + thread.getState());

// Wait for the thread to finish

thread.join();

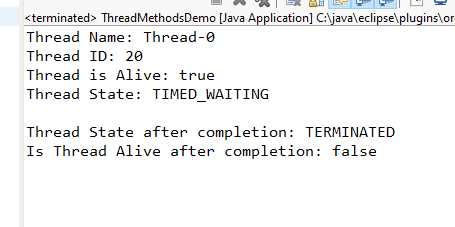
// After thread finishes

System.***out***.println("\nThread State after completion: " + thread.getState());

System.***out***.println("Is Thread Alive after completion: " + thread.isAlive());

}

}

   
  
  
2.       Write  
a multi- threaded program that prints even and odd number one after another  
.create two separate classes to print even and odd number and call there  
objects in main.

**package** multithreading.assignments.que2;

**class** EvenNumberThread **extends** Thread {

**private** **final** Object lock;

// Constructor to pass the lock object

**public** EvenNumberThread(Object lock) {

**this**.lock = lock;

}

@Override

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

**for** (**int** i = 0; i <= 10; i += 2) {

**synchronized** (lock) {

System.***out***.println("Even: " + i);

lock.notify();

**try** {

lock.wait();

} **catch** (InterruptedException e) {

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

}

}

}

}

}

**package** multithreading.assignments.que2;

**class** OddNumberThread **extends** Thread {

**private** **final** Object lock;

**public** OddNumberThread(Object lock) {

**this**.lock = lock;

}

@Override

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

**for** (**int** i = 1; i <= 10; i += 2) {

**synchronized** (lock) {

System.***out***.println("Odd: " + i);

lock.notify();

**try** {

lock.wait();

} **catch** (InterruptedException e) {

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

}

}

}

}

}

**package** multithreading.assignments.que2;

**public** **class** MainThread {

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

Object lock = **new** Object();

EvenNumberThread evenThread = **new** EvenNumberThread(lock);

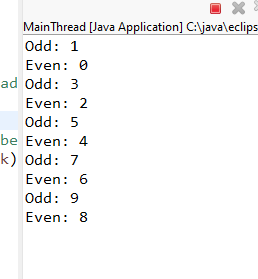
OddNumberThread oddThread = **new** OddNumberThread(lock);

evenThread.start();

oddThread.start();

}

}

   
  
  
   
  
  
3.       Create  
three classes message1 ,message2 and message3  and call   
their methods one after another and display message .  
  
In above case set priority level of all  
object and run them according to priority.  
  
**package** multithreading.assignments.que3;

**public** **class** Message1 **extends** Thread{

**public** **void** show() {

System.***out***.println("Message 1");

}

}

 **package** multithreading.assignments.que3;

**public** **class** Message2 **extends** Thread{

**public** **void** show() {

System.***out***.println("Message 2");

}

}

**package** multithreading.assignments.que3;

**public** **class** Message3 **extends** Thread{

**public** **void** show() {

System.***out***.println("Message 3");

}

}

**package** multithreading.assignments.que3;

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

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

// Create objects of the message classes

Message1 m1 = **new** Message1();

Message2 m2 = **new** Message2();

Message3 m3 = **new** Message3();

// Set thread priorities

m1.setPriority(Thread.***MAX\_PRIORITY***); // Highest priority

m2.setPriority(Thread.***NORM\_PRIORITY***); // Default priority

m3.setPriority(Thread.***MIN\_PRIORITY***); // Lowest priority

// Start the threads

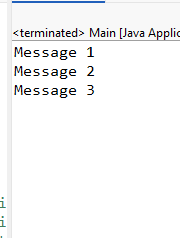
m1.show();

m2.show();

m3.show();

}

}

  
  
Create another class message4 and make sure  
it should not be called before previously define class. Hint use join  
  
**package** multithreading.assignments.que3;

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

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

// Create objects of the message classes

Message1 m1 = **new** Message1();

Message2 m2 = **new** Message2();

Message3 m3 = **new** Message3();

Message4 m4 = **new** Message4();

// Set thread priorities

m1.setPriority(Thread.***MAX\_PRIORITY***); // Highest priority

m2.setPriority(Thread.***NORM\_PRIORITY***); // Default priority

m3.setPriority(Thread.***MIN\_PRIORITY***); // Lowest priority

m4.setPriority(Thread.***NORM\_PRIORITY***); // Normal priority for Message4

// Start the threads

m1.show();

m2.show();

m3.show();

**try** {

// Ensure the main thread waits for m1, m2, m3 to finish

m1.join();

m2.join();

m3.join();

// Start m4 only after m1, m2, and m3 are finished

m4.show();

m4.join();

// Wait for m4 to finish before exiting the main thread

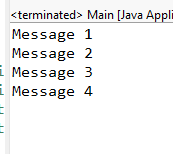
} **catch** (InterruptedException e) {

System.***out***.println("Main thread interrupted");

}

}

}

   
  
  
4.       Write  
a program that contains method even and another method odd , call both of the  
methods in main and print their values after 2000 millisecond and make sure  
that another method should not be called before the task of one is complete .

**package** multithreading.assignments.que4;

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

**public** **synchronized** **void** even() {

**for** (**int** i = 0; i <= 10; i += 2) {

System.***out***.println("Even: " + i);

}

}

**public** **synchronized** **void** odd() {

**for** (**int** i = 1; i <= 10; i += 2) {

System.***out***.println("Odd: " + i);

}

}

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

Main obj = **new** Main();

obj.even();

Thread.*sleep*(2000);

obj.odd();

}

}

