UNIVERSITÀ DI PISA

Laboratorio di reti

Java Thread: Runnable

Prof. Laura Ricci (ricci@unipi.it)

Assistant: Andrea De Salve (desalve@unipi.it)

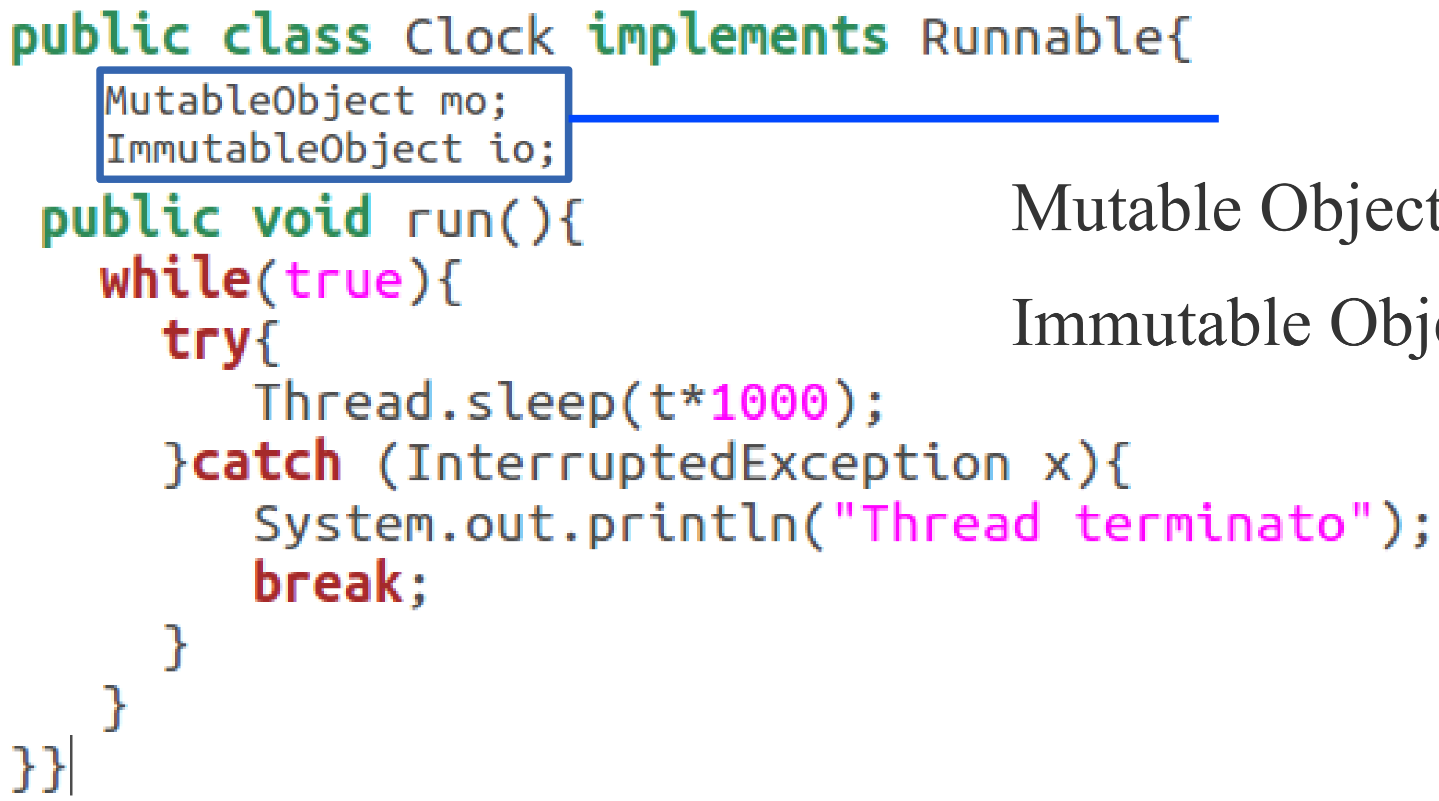
01-03-2016

2015/2016

Creating a Runnable object

- A **Runnable** object for Thread execution:

  Mutable and immutable object



```
public class Clock implements Runnable{
   MutableObject mo;
   ImmutableObject io;
 public void run(){
   while(true){
     try{
        Thread.sleep(t*1000);
     }catch (InterruptedException x){
        System.out.println("Thread terminato");
        break;
     }
   }
}}
```

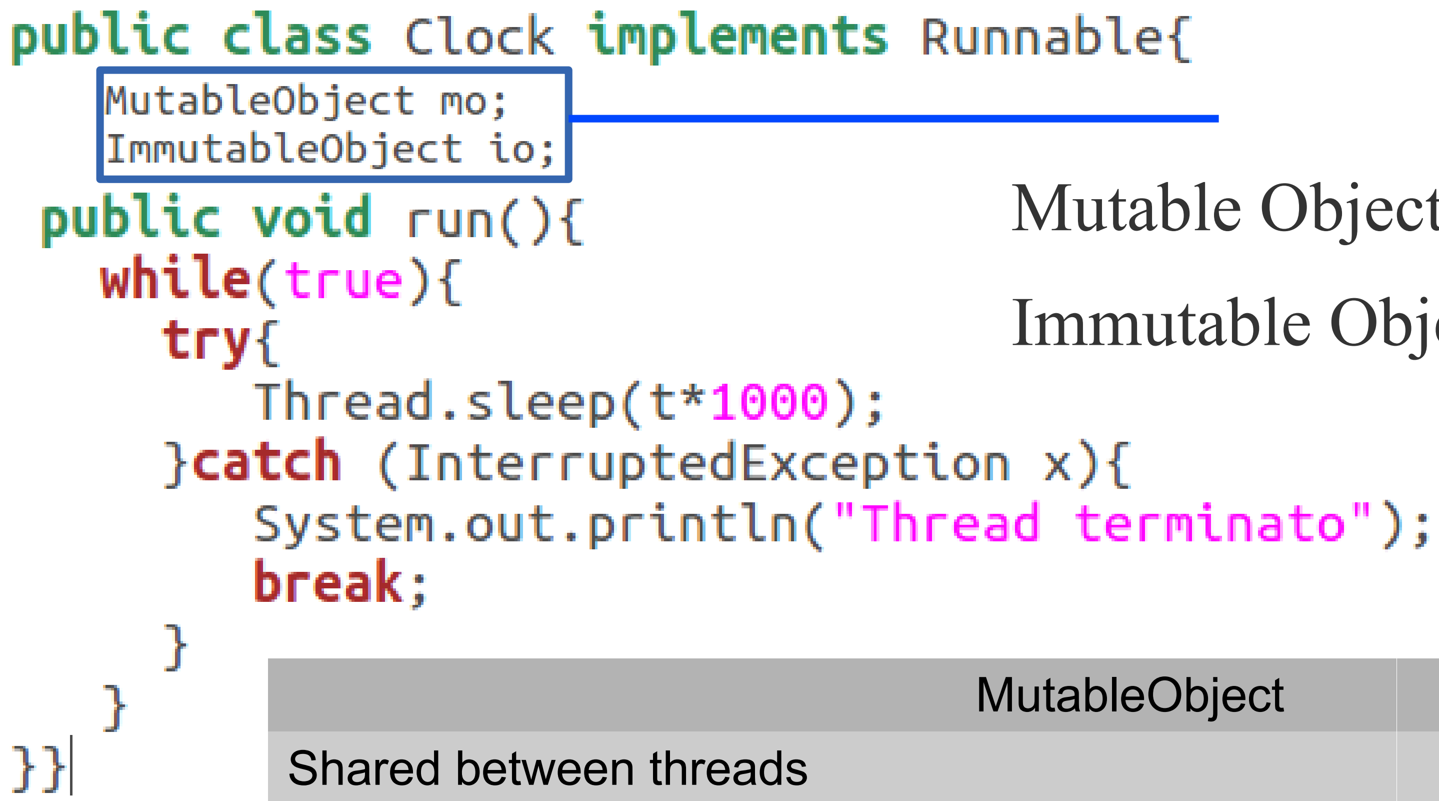
Mutable Object: getter and setter

Immutable Object: getter

Creating a Runnable object

- A **Runnable** object for Thread execution:

  Mutable and immutable object



```
public class Clock implements Runnable{
    MutableObject mo;
    ImmutableObject io;
 public void run(){
   while(true){
     try{
        Thread.sleep(t*1000);
     }catch (InterruptedException x){
        System.out.println("Thread terminato");
        break;
     }
   }
}}
```

Mutable Object: getter and setter

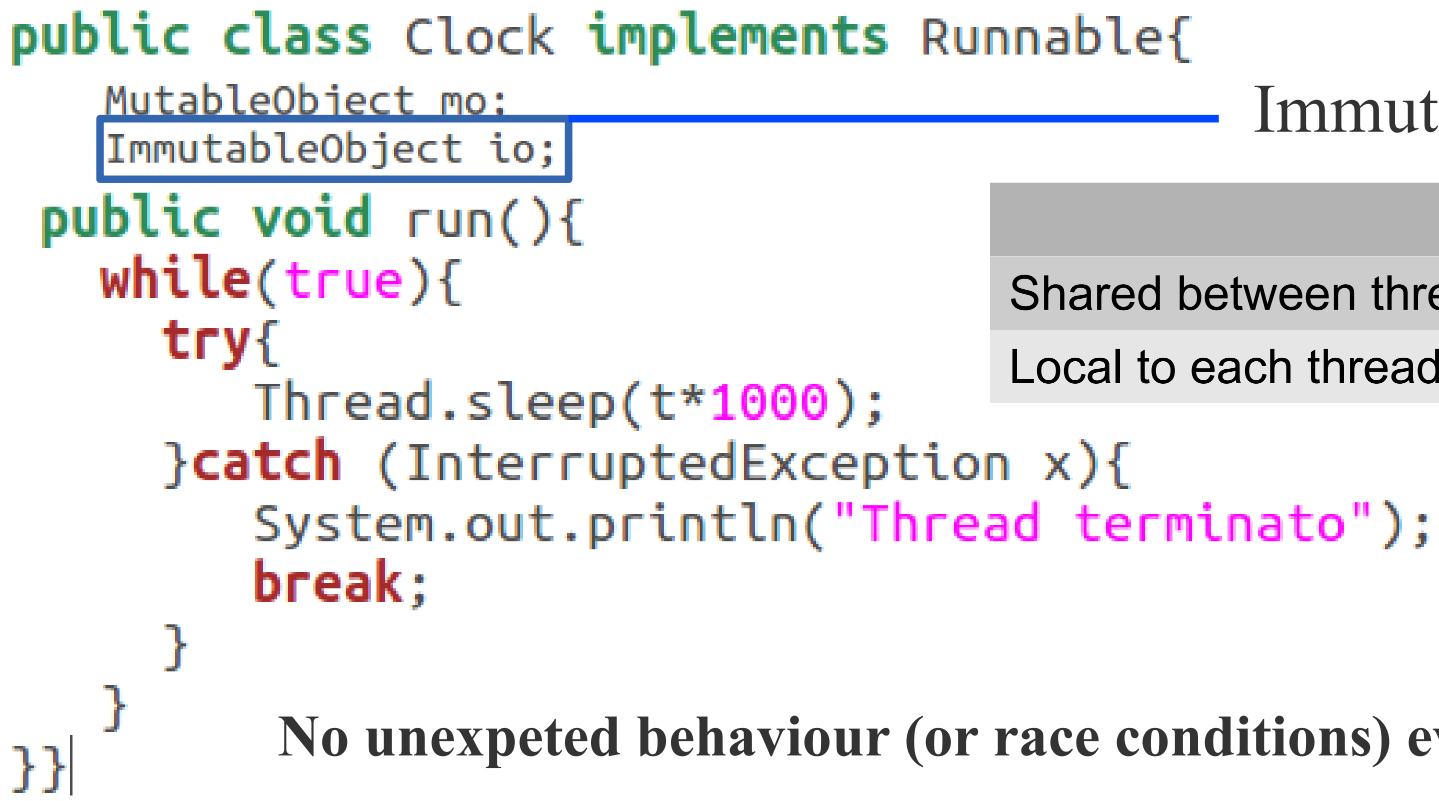
Immutable Object: getter

	MutableObject	ImmutableObject
Shared between threads		
Local to each thread		

Creating a Runnable object

- A **Runnable** object for Thread execution:

  Mutable and immutable object



```
public class Clock implements Runnable{
    MutableObject mo;
    ImmutableObject io;
 public void run(){
   while(true){
     try{
        Thread.sleep(t*1000);
     }catch (InterruptedException x){
        System.out.println("Thread terminato");
        break;
     }
   }
}}
```

Immutable Object: getter

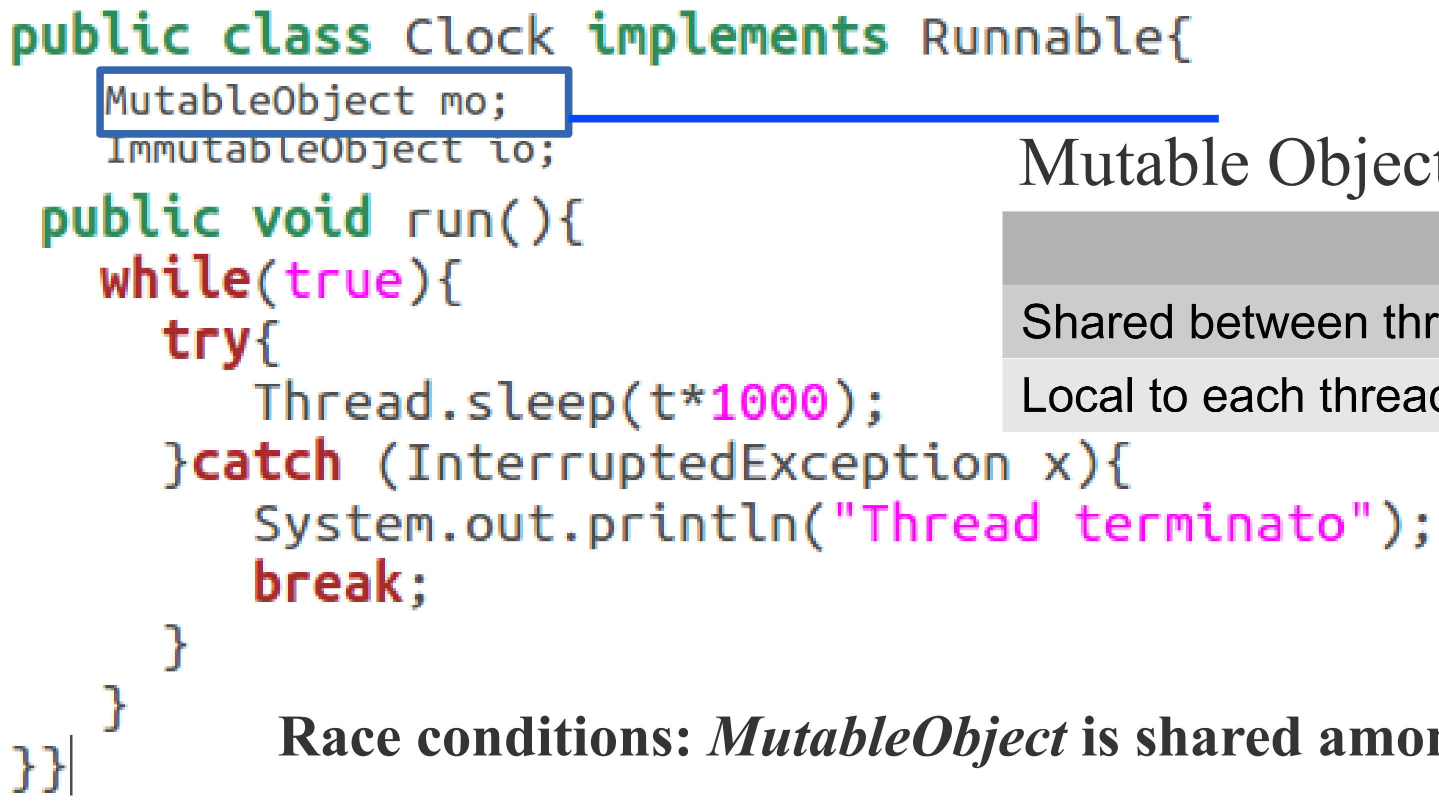
	ImmutableObject
Shared between threads	safe
Local to each thread	safe

No unexpeted behaviour (or race conditions) even if *ImmutableObject* is shared among different threads.

Creating a Runnable object

- A **Runnable** object for Thread execution:

  Mutable and immutable object



```
public class Clock implements Runnable{
    MutableObject mo;
    ImmutableObject io;
 public void run(){
   while(true){
     try{
        Thread.sleep(t*1000);
     }catch (InterruptedException x){
        System.out.println("Thread terminato");
        break;
     }
   }
}}
```

Mutable Object: getter and setter

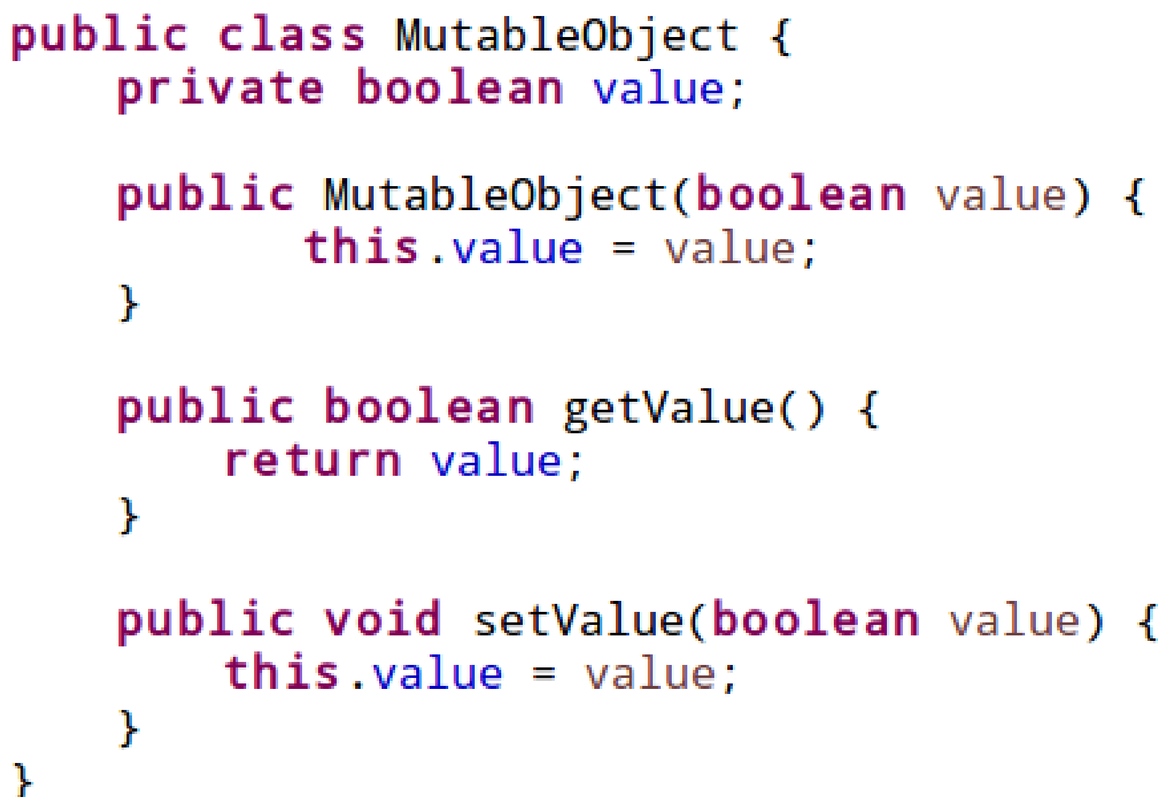
	MutableObject
Shared between threads	race conditions
Local to each thread	safe

Race conditions: *MutableObject* is shared among different threads

Coordination among different threads is required to access *mo*

Example:

- A mutable object shared between two threads:



```java
public class MutableObject {
    private boolean value;

    public MutableObject(boolean value) {
            this.value = value;
    }

    public boolean getValue() {
        return value;
    }

    public void setValue(boolean value) {
        this.value = value;
    }
}
```

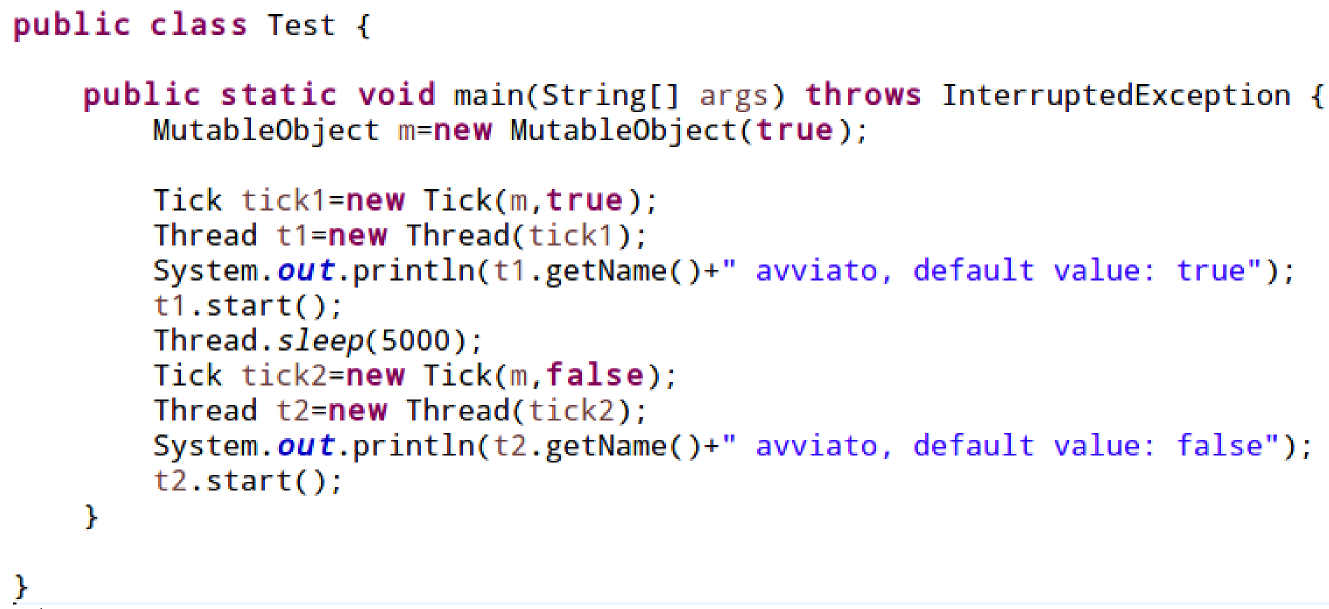
A task for the object

- A task which uses the mutable object:
  - Sets the MutableObject to the default value

```java
public class Tick implements Runnable{

    private MutableObject mo;
    private final boolean defaultValue;

    public Tick(MutableObject mo,boolean defaultValue){
        this.mo=mo;
        this.defaultValue=defaultValue;
    }

    @Override
    public void run() {
        while(true){
            mo.setValue(defaultValue);
        try {
            Thread.sleep((long) (Math.random()*10000));
        } catch (InterruptedException e) {
            return ;
        }
        System.out.printf("Thread %s value %b \n",Thread.currentThread().getName(),mo.getValue());
        }
    }
}
```

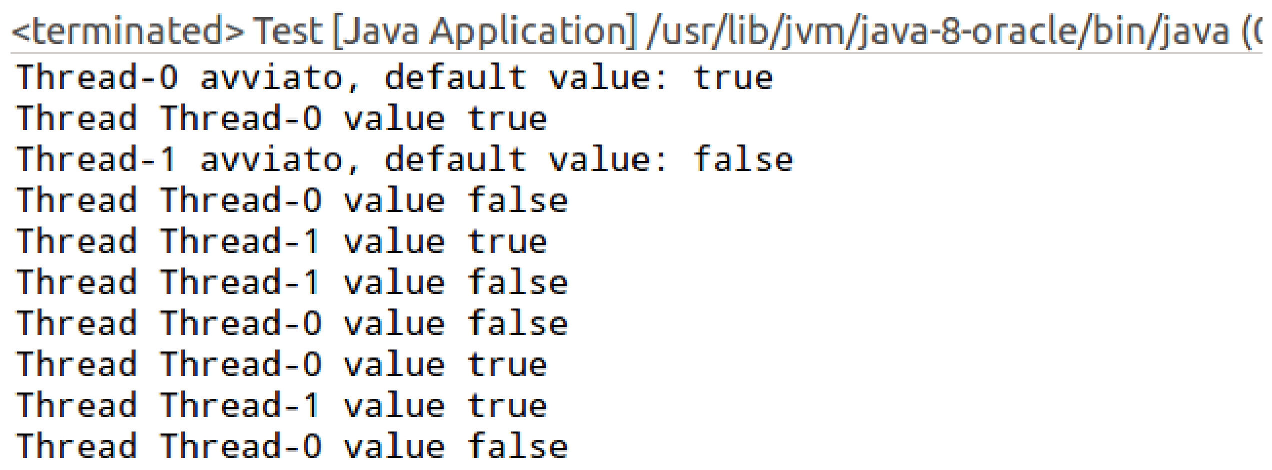
The applicatioin

- The main:



```
public class Test {

    public static void main(String[] args) throws InterruptedException {
        MutableObject m=new MutableObject(true);

        Tick tick1=new Tick(m,true);
        Thread t1=new Thread(tick1);
        System.out.println(t1.getName()+" avviato, default value: true");
        t1.start();
        Thread.sleep(5000);
        Tick tick2=new Tick(m,false);
        Thread t2=new Thread(tick2);
        System.out.println(t2.getName()+" avviato, default value: false");
        t2.start();
    }

}
```

- Output:



```
<terminated> Test [Java Application] /usr/lib/jvm/java-8-oracle/bin/java (
Thread-0 avviato, default value: true
Thread Thread-0 value true
Thread-1 avviato, default value: false
Thread Thread-0 value false
Thread Thread-1 value true
Thread Thread-1 value false
Thread Thread-0 value false
Thread Thread-0 value true
Thread Thread-1 value true
Thread Thread-0 value false
```

Creating a Runnable object

- How to have a **Runnable** object for Thread execution:

2. Construct a Thread passing an inner class that is a *Runnable*

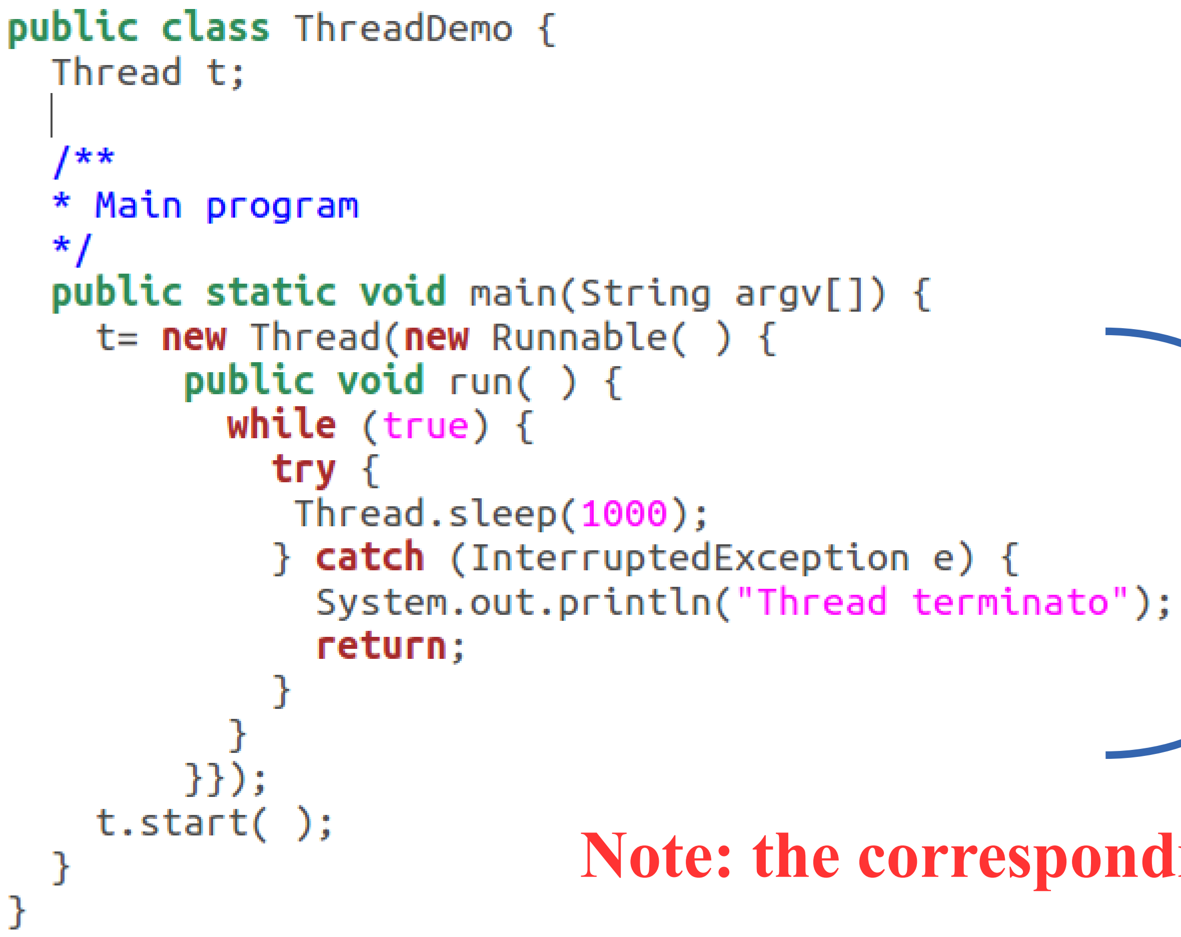
```
public class ThreadDemo {
  Thread t;
  |
  /**
  * Main program
  */
  public static void main(String argv[]) {
    t= new Thread(new Runnable( ) {
        public void run( ) {
          while (true) {
            try {
             Thread.sleep(1000);
            } catch (InterruptedException e) {
             System.out.println("Thread terminato");
             return;
            }
          }
        }});
    t.start( );
  }
}
```

```
public static void main(String argv[]) {
  t= new Thread(new Clock());
  t.start( );
}
```

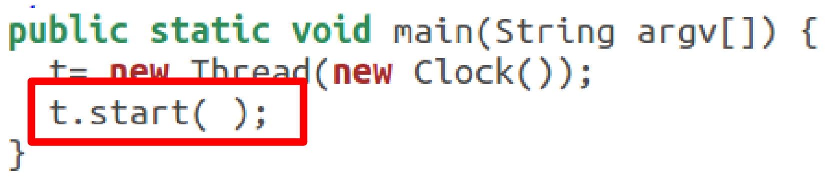
Creating a Runnable object 2

- How to have a **Runnable** object for Thread execution:

2. Construct a Thread passing an inner class that is a *Runnable*



```java
public class ThreadDemo {
  Thread t;
  |
  /**
  * Main program
  */
  public static void main(String argv[]) {
    t= new Thread(new Runnable( ) {
        public void run( ) {
          while (true) {
            try {
             Thread.sleep(1000);
            } catch (InterruptedException e) {
              System.out.println("Thread terminato");
              return;
            }
          }
        }});
    t.start( );
  }
}
```



```java
public static void main(String argv[]) {
  t= new Thread(new Clock());
  t.start( );
}
```

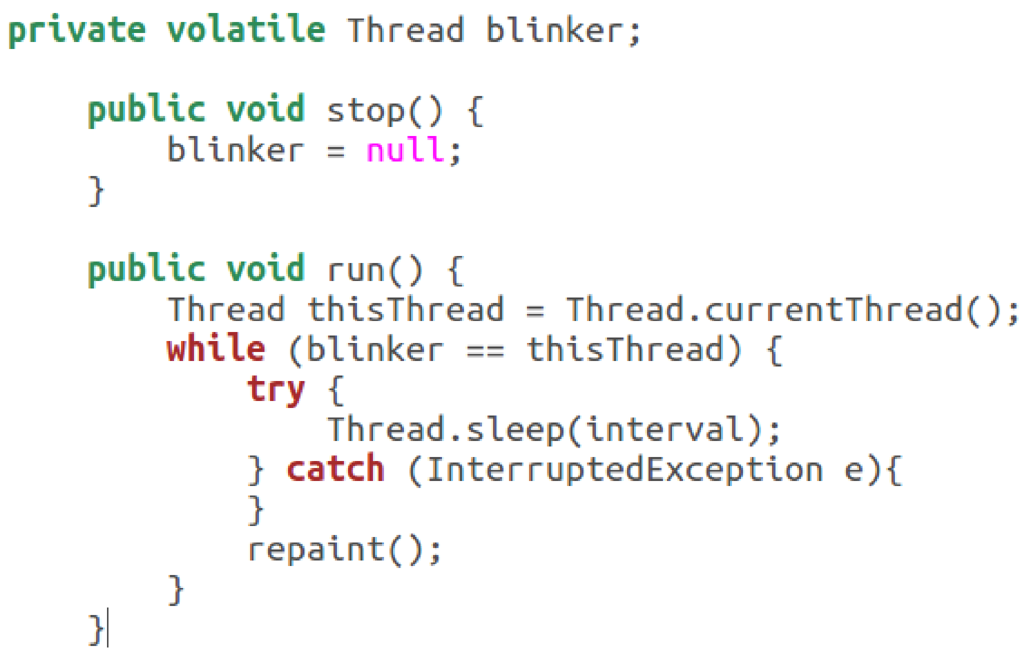
Resulting in *run()* execution

Note: the corresponding *stop()* is deprecated

Recommended Thread Stop: Java

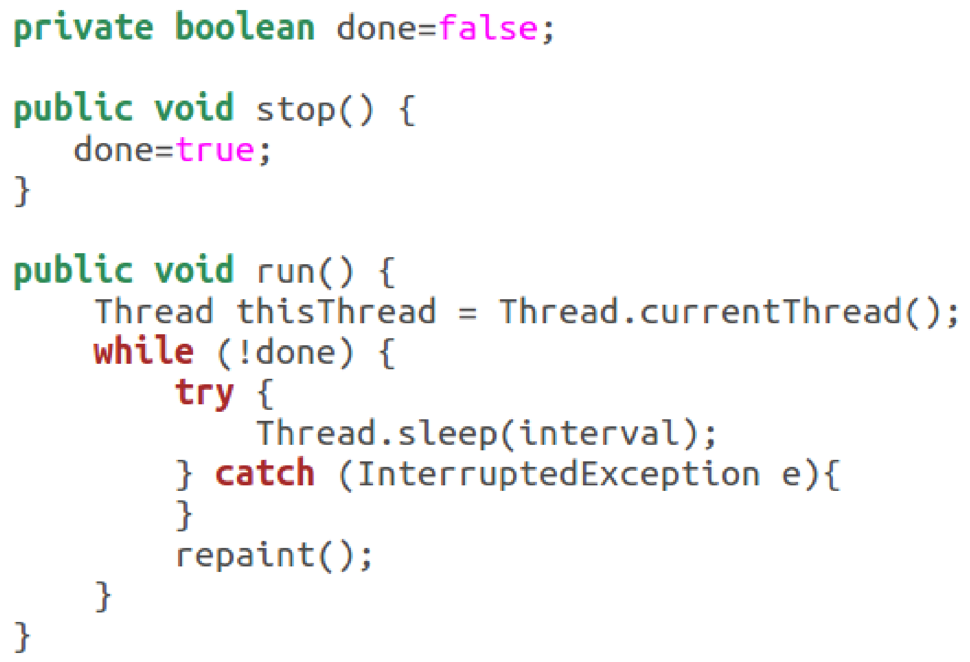
The recommended method: use a boolean variable in the main loop of the *run()* method.

Example 1



```
private volatile Thread blinker;

    public void stop() {
        blinker = null;
    }

    public void run() {
        Thread thisThread = Thread.currentThread();
        while (blinker == thisThread) {
            try {
                Thread.sleep(interval);
            } catch (InterruptedException e){
            }
            repaint();
        }
    }
```

Example 2



```
private boolean done=false;

public void stop() {
   done=true;
}

public void run() {
    Thread thisThread = Thread.currentThread();
    while (!done) {
        try {
            Thread.sleep(interval);
        } catch (InterruptedException e){
        }
        repaint();
    }
}
```

More Info

- Java Tutorial: https://docs.oracle.com/javase/tutorial/
- JavaDoc:

  https://docs.oracle.com