First Semester 2017/2018

Multi-threading Lecture Notes

Summarized by:

Miss Rola Al-Khalid **Mrs. Aseel Al-Anani**

Reference:

Java How to Program, Deitel and Deitel, Edition 9. Publisher: Prentice Hall, New Jersey 2012.

ال thread معناته اكتر من برنامج او اكتر من code بدهم يتنفذو بنفس الوقت ، اذا كان عنا اكتر من processor يعني بكون truely concurrent ، و اذا كان عنا processor واحد يعني في simulation بتم توزيع الوقت بطريقة معينة حسب نظام التشغيل المستخدم حتى يقدر يوازن بين الأشياء اللي لازم تتنفذ.

Multithreading

❖ Introduction

- Human body perform a great variety of operation in parallel → Concurrently
- Computer can perform operations concurrently (Compile a program, sent message, print file, receive email).
- Only computers that have multiple processors can truly execute operations concurrently
- Operating systems on single processor computers use various techniques to simulate concurrency.
- Most programming language do not enable programmers to specify concurrent activities. Rather, the languages generally provide control statements that only enable programmer to perform one action at a time.

- Java provides built-in multithreading
  - Multithreading improves the performance of some programs.
  - The programmer specifies that applications contain threads of execution, where each thread designates a portion of program that may execute concurrency with other threads.
  - Multithreading is not available in C and C++

- Writing multithreaded programs can be tricky.
- Although the human mind can perform functions concurrently, people find it difficult to jump between parallel trains of thought.
  - Ex. Reading three different books concurrently.

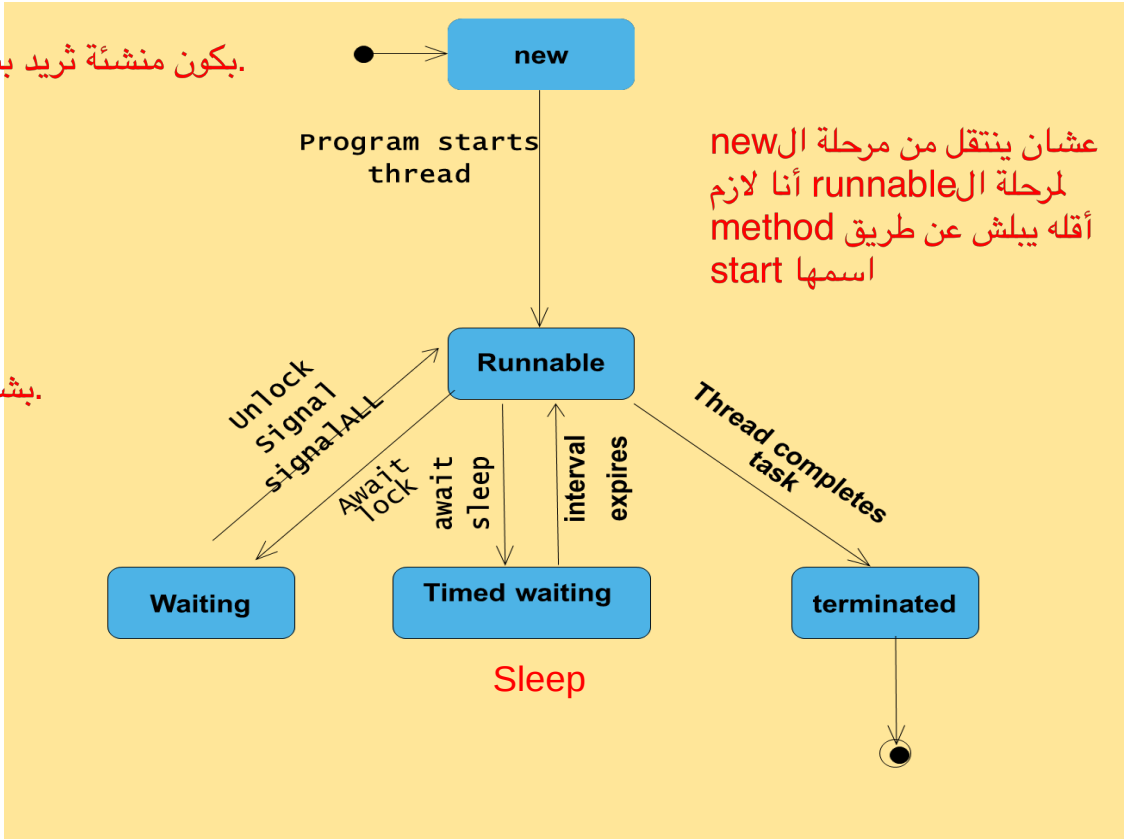
Reference:

Java How to Program, Deitel and Deitel, Edition 9. Publisher: Prentice Hall, New Jersey 2012.

- Thread States: Life Cycle of a Thread
  - At any time, a thread is said to be in one of several thread states:

مهمة بيجي عليها أسئلة

new: بكون منشئة ثريد بس لسا مش شغال.



عشان ينتقل من مرحلة الnew لمرحلة الrunnable أنا لازم أقله يبلش عن طريق method اسمها start

runnable: بشتغل ال ثريد.

  - **New state:**
    - A new thread begins its life cycle in the new **state**
    - Thread was just created.
    - Remains in this state until the program starts the thread.
  - **Runnable state:**
    - **Considered to be executing its task.**
    - At the operating-system level, Java's *runnable* state typically encompasses two separate states

عبارة عن مرحلتين
ready and running

    - Ready state
      - When a thread first transitions to the *runnable* state from the new state, it is in the *ready* state (Thread's start method invoked)

Reference:

Java How to Program, Deitel and Deitel, Edition 9. Publisher: Prentice Hall, New Jersey 2012.

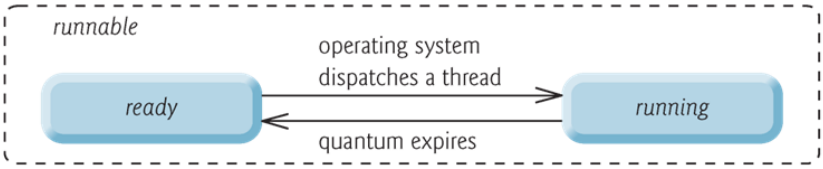
- A thread is waiting for the operating system to assign a processor.

- Running state

المصطلح المستخدم أني أنا بستنا نظام التشغيل بيعطيني ال processor ك thread واشتغل و أفوت على مرحلة ال running اسمه dispatch

dispatch: to assign the processor to the thread.

  - A *ready* thread enters the *running* state (i.e., begins executing) when the operating system assigns it to a processor—also known as dispatching the thread.
  - Typically, each thread is given a quantum or time slice in which to perform its task.



لما تخلص ال quantum برجع عال ready

- **Waiting state:**
  - Thread waits for another thread to perform a task.
  - A thread transitions back to the runnable state only when another thread signals the waiting thread to continue executing.

- **Timed waiting:** sleep
  - Enter the state for a specified interval of time.
  - Transitions back to the runnable state when the time interval expires.
  - Timed waiting threads cannot use a processor, even if one is available.
  - A sleeping thread remains in the *timed waiting* state for a designated period of time (called a sleep interval), after which it returns to the *runnable* state.

- **Terminates state:**
  - Thread enter this state when it complete its task.
  - Eventually disposed of by system.

Reference:

Java How to Program, Deitel and Deitel, Edition 9. Publisher: Prentice Hall, New Jersey 2012.

- **Thread Priorities and Thread Scheduling**
  - The process that an operating system uses to determine which thread to dispatch is called thread scheduling.
  - Java thread priority:
    - Every Java thread has a thread priority that helps determine the order in which threads are scheduled.
    - Threads have priority from 1 to 10.
    - The threads so far had same default priority (NORM_PRIORITY) .
    - Java allows users to change priority:

      ```
      ThreadName.setPriority(int Number)
      ```

    - Priority constants are defined:
      - **Thread.MIN_PRIORITY --> 1**
      - **Thread.NORM_PRIORITY --> 5**
      - **Thread.MAX_PRIORITY --> 10**
    - New threads inherit priority of thread that created it.
    - Informally, higher-priority threads are more important to a program and should be allocated processor time before lower-priority threads.
    - Thread priorities cannot guarantee the order in which threads execute.
  - Timeslicing
    - Most operating systems support timeslicing, which enables threads of equal priority to share a processor.
    - Each thread assigned time on the processor (called a quantum)
    - An operating system's thread scheduler determines which thread runs next.
    - One simple thread-scheduler implementation keeps the highest-priority thread running at all times and, if there's more than one highest-priority

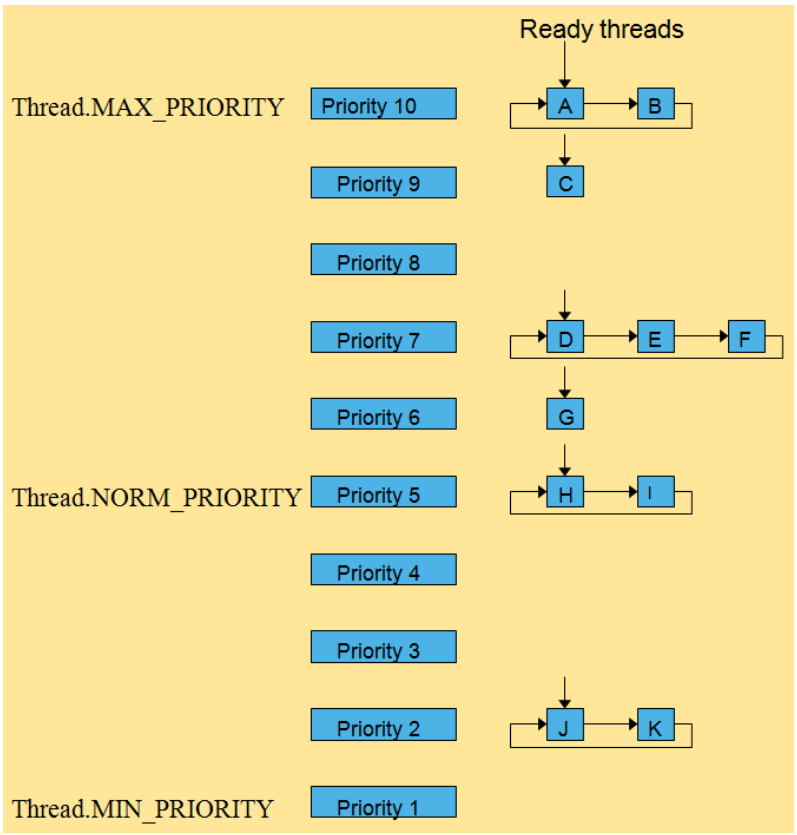
Reference:

Java How to Program, Deitel and Deitel, Edition 9. Publisher: Prentice Hall, New Jersey 2012.

thread, ensures that all such threads execute for a quantum each in round-robin fashion. This process continues until all threads run to completion.

- Thread scheduling is platform dependent - the behavior of a multithreaded program could vary across different Java implementation.

- Thread priority scheduling example:



- **Two ways to define a thread:**
  - Create a class that extends the Thread class (java.lang.Thread)
  - Create a class that implements the Runnable interface (java.lang.Runnable)

*extend: class
*implements: interface

- **Extending Thread class**
  - Create a class by extending (inheriting) Thread class and override run() method:

```
class MyThread extends Thread  {
    public void run() {
        // thread body of execution
```

Reference:

Java How to Program, Deitel and Deitel, Edition 9. Publisher: Prentice Hall, New Jersey 2012.

run: abstract , i must override it

*اول ما أنشئ الثريد واعمل start لحاله بنفذ الrun ، يعني ممنوع أنا استدعيها

*البودي تبع الrun فيه الtask اللي بدّي أنفذها

```
        }
    }
```

- Create a thread:

```
MyThread thr1 = new MyThread();
```

object
من الكلاس اللي عاملة اكستند للثريد

- Start Execution of threads:

```
thr1.start();
```

- Or Create and Execute:

```
new MyThread().start();
```

- **Implementing Runnable interface**
  - Java does not support multiple inheritance (cannot extend more than one class). Instead, use interfaces.
  - The Runnable interface declares the single method run, which contains the code that defines the task that a Runnable object should perform.
  - When a thread executing a Runnable is created and started, the thread calls the Runnable object's run method, which executes in the new thread.

  - Create a class that implements the interface Runnable and override run() method:

```
class MyThread implements Runnable{
    .....
    public void run() {
        // thread body of execution
    }
}
```

  - Creating Object:

```
MyThread myObject = new MyThread();
```

  - Creating Thread Object:

```
Thread thr1 = new Thread( myObject );
```

  - Start Execution:

Reference:

Java How to Program, Deitel and Deitel, Edition 9. Publisher: Prentice Hall, New Jersey 2012.

```
thr1.start();
```

Reference:

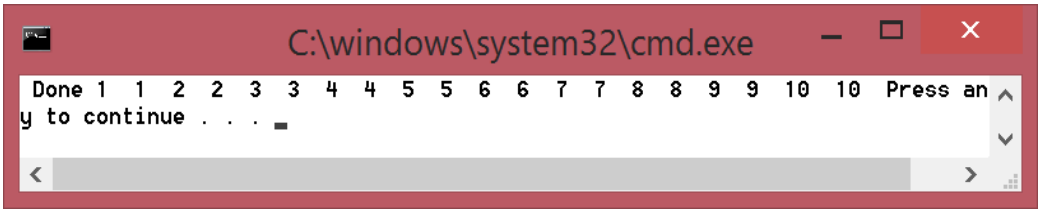
Java How to Program, Deitel and Deitel, Edition 9. Publisher: Prentice Hall, New Jersey 2012.

Example -1-

```
public class Thread1 extends Thread{
   public void run(){
            for(int i=1; i<=10; i++)
                     System.out.printf("%d  ",i);
   }
   public static void main(String args[]){
             Thread1 t1 = new  Thread1();
             t1.start();
             Thread1 t2 = new  Thread1();
             t2.start();
             System.out.print(" Done ");
  }}
```

Example -2-

```
public class Thread2 implements Runnable{
   public void run(){
             for(int i=1; i<=10; i++)
                     System.out.printf("%d  ",i);
   }
   public static void main(String args[]){
             Thread t1 = new Thread(new Thread2());
             t1.start();
             Thread t2 = new Thread(new Thread2());
             t2.start();
             System.out.print(" Done ");
  } }
```



- Thread Constructor
  - **Thread() -** Creates a thread with an auto-numbered name of format **Thread-1**, **Thread-2**...
  - **Thread(String threadName) -** Creates a thread with name
  - **Thread(Runnable runnableObject)**

Reference:

Java How to Program, Deitel and Deitel, Edition 9. Publisher: Prentice Hall, New Jersey 2012.

- **An Overview of the Thread Methods:**
  - **void run**()
    - Does "work" of a thread.
    - Can be overridden in subclass of **Thread** or in **Runnable** object.
  - **void start()**
    - Begins a thread running (calls **run)**
    - This method can be called only once (Error to call **start** twice for same thread).
  - **void stop()**
    - The thread is being terminated.
  - **static** void **sleep(long milliseconds)**
    - Thread static method sleep places a thread in the timed waiting state for the specified amount of time.
    - Can throw a checked exception of type InterruptedException if the sleeping thread's interrupt method is called.
    - Why might we want a program to invoke sleep?
      - Can give lower priority threads a chance to run.
  - **int getPriority()**
    - Returns this thread's priority
  - **void setPriority(int priorityNumber)**
    - Change the priority of the thread (1 to 10)
  - **String getName()**
    - Return the thread name.
  - **void suspend()**
    - Suspend the execution of the thread until resume is called.
  - **void resume()**
    - Resume the suspended thread.
    - It has no effect on a thread that is not suspended.

Reference:

Java How to Program, Deitel and Deitel, Edition 9. Publisher: Prentice Hall, New Jersey 2012.

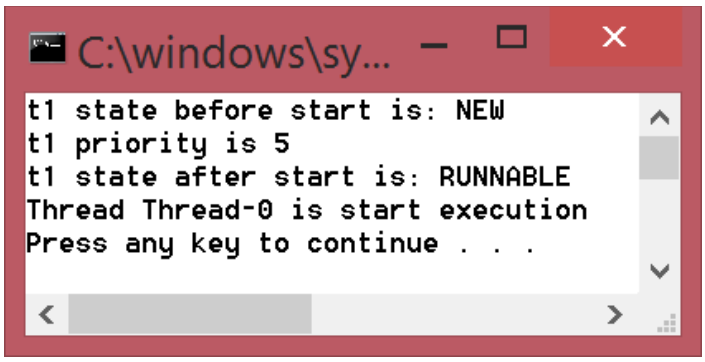
- **static Thread currentThread()**
- **getState()**

Reference:

Java How to Program, Deitel and Deitel, Edition 9. Publisher: Prentice Hall, New Jersey 2012.

Example -3-

```
public class Thread3 extends Thread{
  public void run(){
       System.out.printf("Thread %s is start execution\n",getName());
       try{  Thread.sleep(1000); }
       catch(Exception e){   }
  }
  public static void main(String args[]){
            Thread3 t1 = new  Thread3();
            System.out.println("t1 state before start is: "+t1.getState());
            System.out.printf("t1 priority is %d\n", t1.getPriority());
            t1.start();
            System.out.println("t1 state after start is: "+t1.getState());
 }}
```

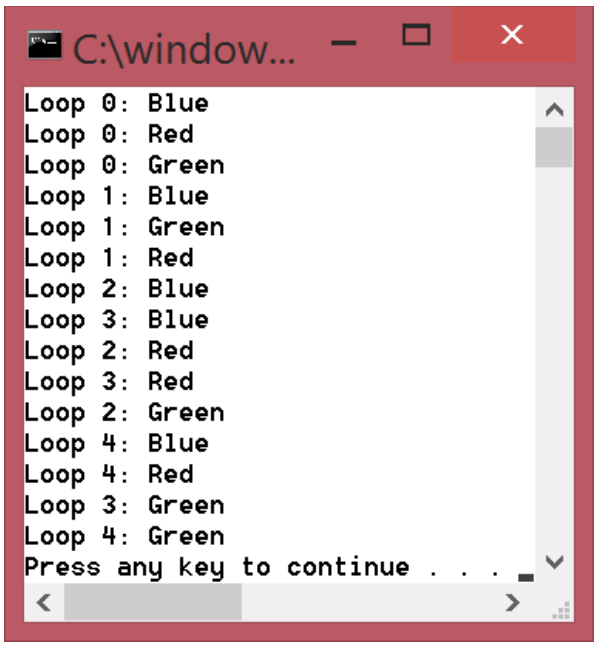


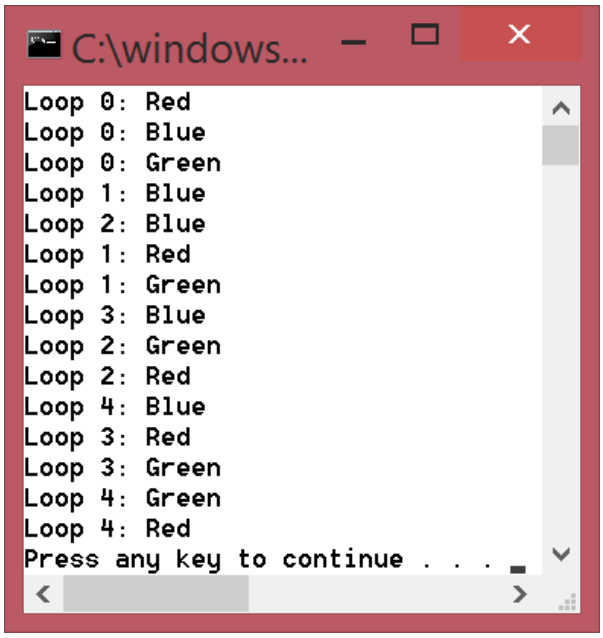
Example -4-

```
public class Thread4 {
        public static void main(String args[]) {
                new ThreadTest("Red").start();
                new ThreadTest("Green").start();
                new ThreadTest("Blue").start();
        }
}
class ThreadTest extends Thread {
        public ThreadTest(String str) {
                super(str);  the first statement in a subclass constructor is to to call the superclass
        }                    constructor
        public void run() {
                for (int i = 0; i < 5; i++) {
                        System.out.println("Loop " + i + ": " + getName());
                        try {
                                sleep((int) (Math.random() * 2000));
                        }
                        catch (InterruptedException e) {
                        }
                }
  }}
```


Reference:

Java How to Program, Deitel and Deitel, Edition 9. Publisher: Prentice Hall, New Jersey 2012.





Reference:

Java How to Program, Deitel and Deitel, Edition 9. Publisher: Prentice Hall, New Jersey 2012.