Multithreading programming

جامعة طرابلس - كلية تقنية المعلومات

د. عبدالحميد الواعر

Multithreading programming

Multitasking

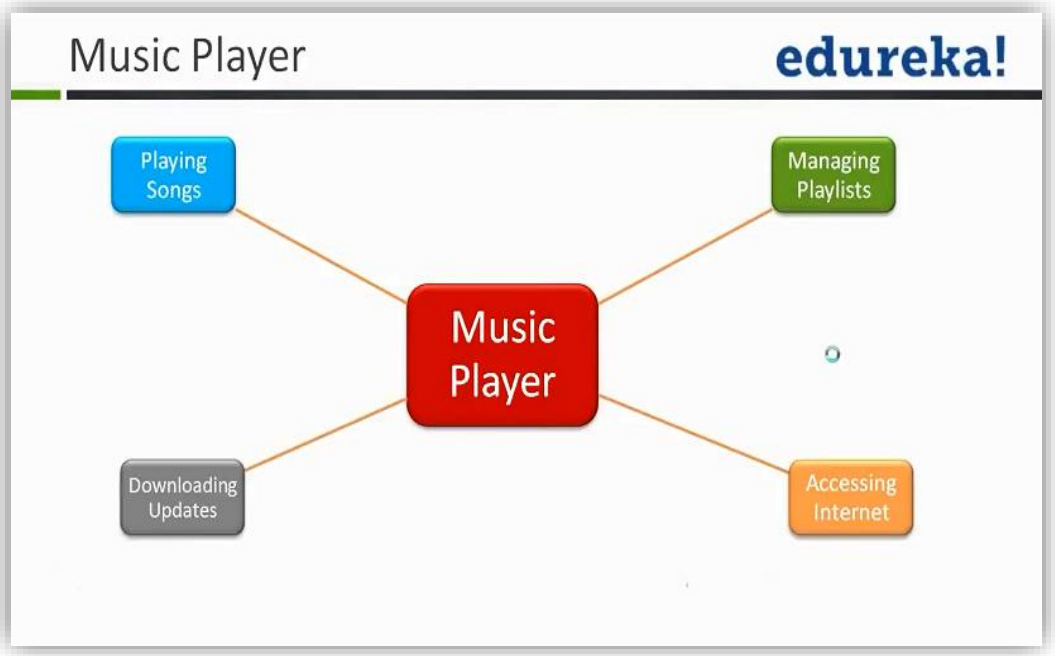
- هي القدرة على تنفيذ مجموعة من البرامج (المهام) في وقت واحد وعلى جهاز حاسوب واحد.

Process

- عبارة عن تنفيذ برنامج (Task) في الحاسوب في بيئة خاصة به.

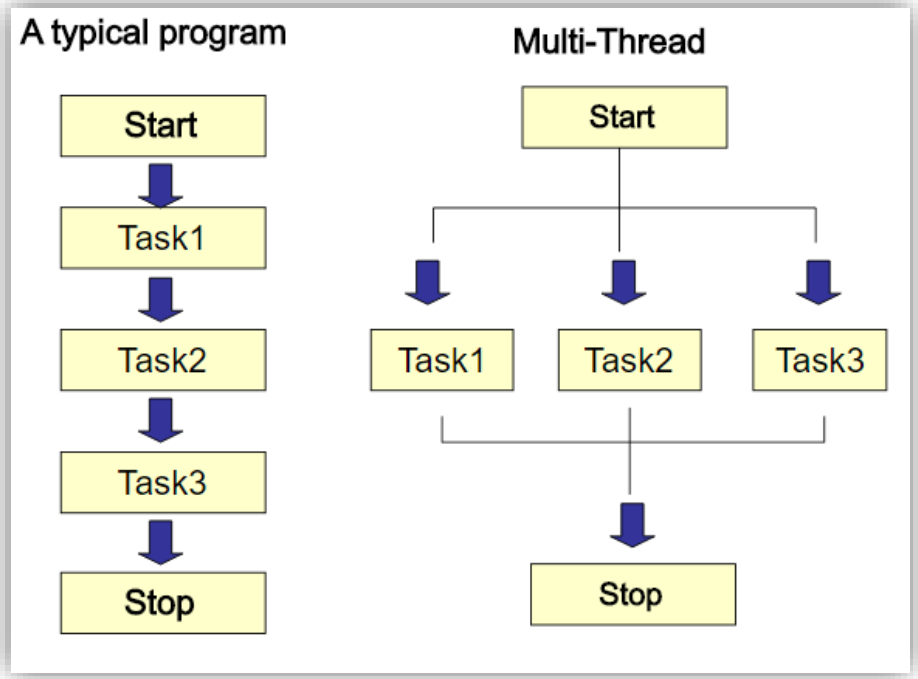
Thread

- عبارة عن سلسلة من التعليمات ضمن برنامج يمكن تنفيذها بشكل مستقل عن باقي أجزاء البرنامج.



Multithreading program

- هو البرنامج الذي يحتوي على مجموعة من الاجزاء التي يمكن تنفيذها في وقت واحد وكل جزء يقوم بتنفيذ مهمة محددة.



- كل جزء يتم تشغيله في Threadخاص به.
- يعتبر صورة خاصة من multitasking.

مثال :

- تطبيق محرر النصوص يسمح للمستخدم بالكتابة ويقوم يعملية التدقيق الاملائي في نفس الوقت.
- برنامج تحميل الملفات من الانترنت يقوم بتحميل عدة ملفات في نفس الوقت.

Java Threads

- في لغة جافا يمكن أنشاء Thread بأستخدام أحدى الطريقتين التاليتين:
  - استخدام Thread class.
  - استخدام Runnable interface.

أستخدام Thread class

- انشاء Thread يتم أنشاء class تقوم بعمل **extends** ل **Thread class** ومن ثم عمل **override** للدالة **run** الموجودة في Thread class.

```
Class MyThread extends Thread {
        @override
         public void run(){
                .....
        }
}
```

مثال:

- البرنامج التالي يقوم بأنشاء thread باستخدام Thread class يقوم بطباعة جملة Hello world عشر مرات :

```
public class MyThread extends Thread {

    @Override
    public void run() {
        for(int i=1;i<=10;i++){
            System.out.println("Hello world "+i);
        }
    }
}
```

أستخدام Runnable Interface

- لأنشاء Thread يتم أنشاء class تقوم بعمل **implementation** ل **Runnable interface** ومن ثم عمل **override** للدالة **run**.

```
Class MyThread implements Runnable {
        @override
         public void run(){
                .....
        }
}
Thread t = new Thread(new MyThread());
```

مثال:

- البرنامج التالي يقوم بأنشاء thread بأستخدام Runnable interface يقوم بطباعة جملة Hello world عشر مرات :

```
public class MyThreadUsingRunnable implements Runnable {

    @Override
    public void run() {
        for (int i = 1; i <= 10; i++) {
            System.out.println("Hello world " + i);
        }
    }

    public static void main(String[] args) {
        Thread t = new Thread(new MyThreadUsingRunnable());

    }

}
```

Starting Thread

- تستخدم الدالة start() في تنفيذ thread.
- البرنامجين التاليين يقوم بأنشاء thread يقوم بطباعة جملة Hello world عشر مرات وتنفيذه عن طريق الدالة start():

```
public class MyThread extends Thread {

    @Override
    public void run() {
        for(int i=1;i<=10;i++){
            System.out.println("Hello world "+i);
        }
    }

    public static void main(String[] args) {
        MyThread t = new MyThread();
        t.start();
    }

}
```

```
public class MyThreadUsingRunnable implements Runnable {

    @Override
    public void run() {
        for (int i = 1; i <= 10; i++) {
            System.out.println("Hello world " + i);
        }
    }

    public static void main(String[] args) {
        Thread t = new Thread(new MyThreadUsingRunnable());
        t.start();
    }

}
```

Main thread

- هو thread الاول الذي يتم تنقيده عند بداية تنفيذ برنامج جافا .
- Main Thread يقوم بالتالي:
  - تنفيد أي threads أخرى.
  - أخر thread يتم أنهاؤه لأنه المسئول عن أنهاء كل الانشطة في البرنامج قبل أنهاء تنفيد البرنامج.

مثال:

- البرنامج التالي يقوم بأنشاء class بها thread يقوم بطباعة جملة Hello world خمس مرات ومن ثم الحصول على ثلاث objects من هذه class تمثل ثلاث threads ومن ثم تنفيذها عن طريق الدالة start():

```
public class MultiThreads extends Thread {

    @Override
    public void run() {
        for(int i=1;i<=5;i++){
            System.out.println("Hello world "+i);
        }
    }

    public static void main(String[] args) {
        MultiThreads t1 = new MultiThreads();
        MultiThreads t2 = new MultiThreads();
        MultiThreads t3 = new MultiThreads();
        t1.start();
        t2.start();
        t3.start();
    }

}
```

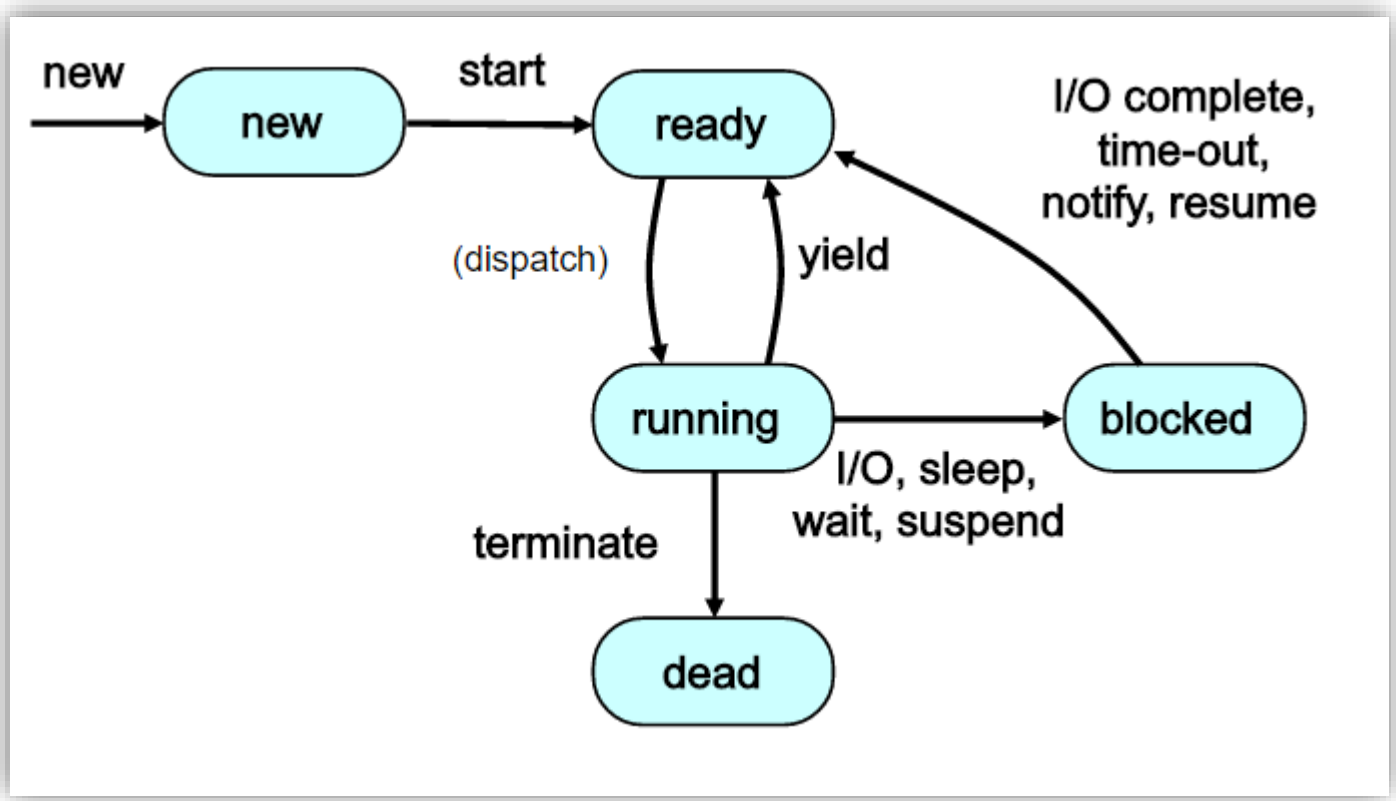
```
run:
Hello world 1
Hello world 1
Hello world 2
Hello world 3
Hello world 4
Hello world 1
Hello world 2
Hello world 5
Hello world 3
Hello world 4
Hello world 2
Hello world 5
Hello world 3
Hello world 4
Hello world 5
```

Thread Live Cycle

- Thread يمر بعدة حالات خلال دورة حياته وهي:

- **Ready**
- وهي الحالة التي يكون فيها مستعد للتنفيذ بمجرد الحصول على CPU.
- **Running**
- وهي الحالة التي يكون حالة تنفيذ (run).
- **Blocked**
- يكون في هذه الحالة عندما ينتظر مورد ما.
- **Dead**
- وهي الحالة التي يكون فيها قد أنهاء عمله.

Live cycle Thread



Thread methods

- Thread class توفر مجموعة من الدوال التي يمكن استخدامها مع threads ومنها :

- **Start()**
- تستخدم لبدء Thread.
- **isAlive()**
- تستخدم لمعرفة thread مازال في حالة تنفيذ.
- **Sleep()**
- تضع thread الحالي في حالة sleep لزمن محدد بعدد من milliseconds.
- **yield()**
- توقف تنفيذ thread الحالي لتسمح ل threads الاخرى بالتنفيذ.

- وغيرها من الدوال الاخرى مثل setPriority(int p),join(), notifyall() , notify()وغيرها.

مثال:

- البرنامج التالي يستخدم الدالة **sleep()**.

```
public class ThreadSleepExample implements Runnable{

    @Override
    public void run() {
        for (int i = 1; i <= 3; i++) {

            System.out.println(Thread.currentThread().getName() + "  " + i);
            try {
                // thread to sleep for 1000 milliseconds
                Thread.sleep(1000);
            } catch (Exception e) {
                System.out.println(e);
            }
        }
    }

    public static void main(String[] args) throws Exception {
        Thread t = new Thread(new ThreadSleepExample());
        t.start();

        Thread t2 = new Thread(new ThreadSleepExample());
        t2.start();
    }
}
```

```
run:
Thread-0  1
Thread-1  1
Thread-0  2
Thread-1  2
Thread-1  3
Thread-0  3
```

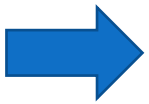
مثال:

- البرنامج التالي يستخدم الدالة **yield()**.

```
public class ThreadYieldExample extends Thread {

    @Override
    public void run() {
        for (int i = 1; i <= 5; i++) {
            try {
                Thread.yield();
            } catch (Exception e) {
                System.out.println(e);
            }
            System.out.println(i);
        }
    }

    public static void main(String args[]) {
        ThreadYieldExample t1 = new ThreadYieldExample();
        ThreadYieldExample t2 = new ThreadYieldExample();
        ThreadYieldExample t3 = new ThreadYieldExample();
        t1.start();
        t2.start();
        t3.start();
    }
}
```



```
run:
1
2
3
4
5
1
1
2
3
4
5
2
3
4
5
```

مثال:

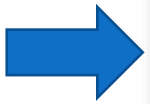
- البرنامج التالي يستخدم الدالة **join()**.

```
public class ThreadJoinExample extends Thread{

    @Override
    public void run() {
        for (int i = 1; i <= 3; i++) {

            System.out.println(Thread.currentThread().getName() + "  " + i);
            try {
                // thread to sleep for 1000 milliseconds
                Thread.sleep(1000);
            } catch (Exception e) {
                System.out.println(e);
            }
        }
    }

    public static void main(String[] args) throws Exception {
        Thread t = new Thread(new ThreadSleepExample());
        t.start();
        t.join();
        Thread t2 = new Thread(new ThreadSleepExample());
        t2.start();
    }
}
```



```
run:
Thread-0  1
Thread-0  2
Thread-0  3
Thread-1  1
Thread-1  2
Thread-1  3
```

Thread synchronization

- Thread synchronization هي عملية التحكم في الوصول إلى أي مصدر مشترك بين مجموعة من threads.
- تستخدم synchronized keyword للسماح ل thread واحد فقط بتنفيذ الجمل الموجودة في synchronized block.

مثال:

- البرنامج التالي يستخدم عدد من threads تقوم بطباعة حاصل ضرب عدد ما في الاعداد من واحد إلى خمسة.

```java
public class PrintThread extends Thread {
    PrintData pd;
    int value;

    PrintThread(PrintData p, int v) {
        this.pd = p;
        this.value = v;
    }
    @Override
    public void run() {
        pd.printValues(value);
    }
    public static void main(String args[]) {
        PrintData obj = new PrintData();
        PrintThread t1 = new PrintThread(obj, 5);
        PrintThread t2 = new PrintThread(obj, 100);
        t1.start();
        t2.start();
    }

}
```

```java
public class PrintData {

        void printValues(int n) {
         for (int i = 1; i <= 5; i++) {
             System.out.println(n * i);
             try {
                 Thread.sleep(500);
             } catch (Exception e) {
                 System.out.println(e);
             }
         }

    }
}
```

النتيجة

```
run:
5
100
10
200
15
300
20
400
25
500
```

مثال:

- تعديل البرنامج واستخدام thread synchronization للدالة printValues الموجودة ب PrintData class.

```
public class PrintData {

    synchronized void printValues(int n) {
        for (int i = 1; i <= 5; i++) {
            System.out.println(n * i);
            try {
                Thread.sleep(500);
            } catch (Exception e) {
                System.out.println(e);
            }
        }

    }
}
```

النتيجة

```
5
10
15
20
25
100
200
300
400
500
```

شكراً