الوحدة الثانية

العناصر الأساسية لبرمجيات الشبكات

Main Components of Network

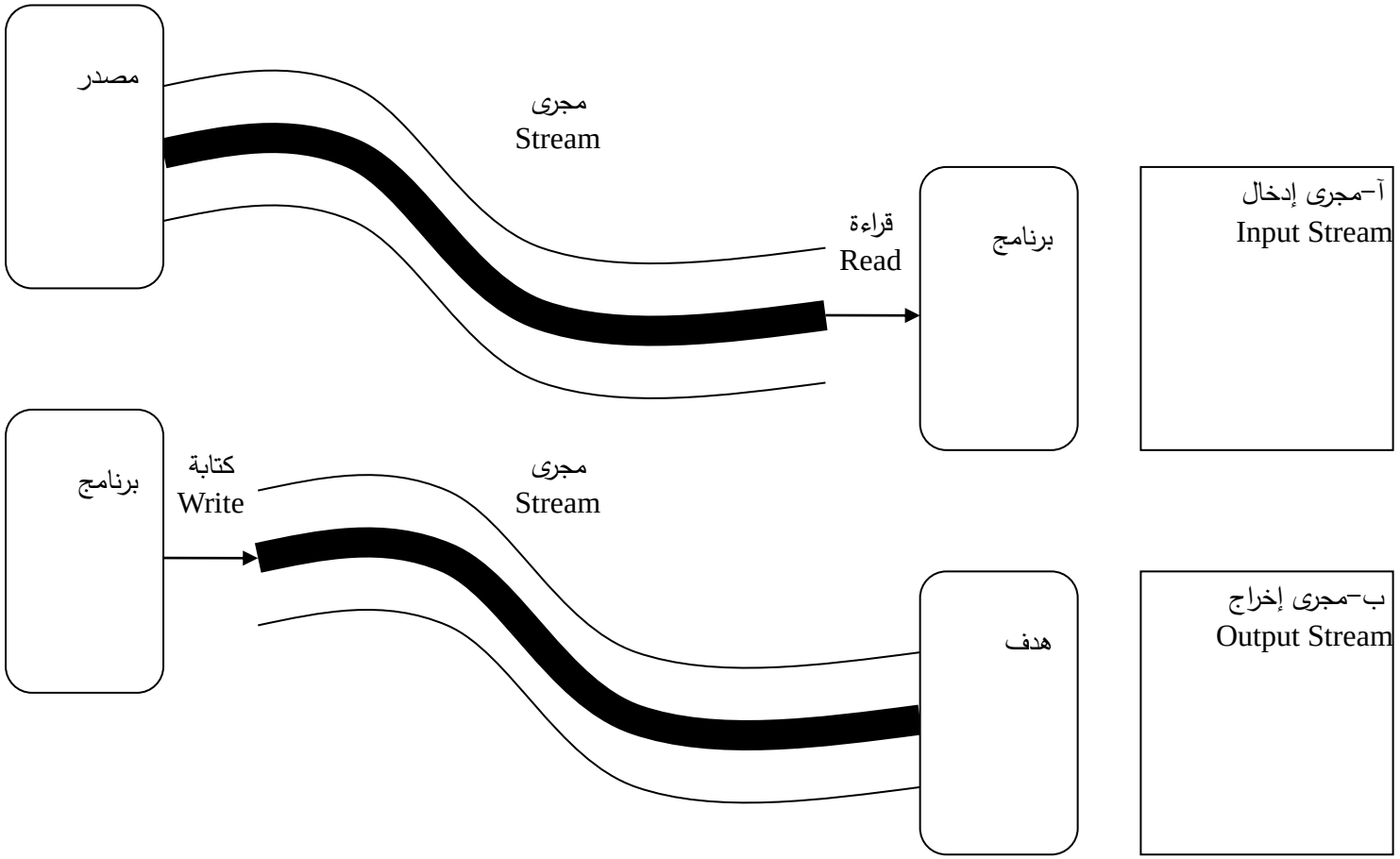
1. مقدمة

- فقد تم تخصيص هذه الوحدة من أجل دراسة العناصر البرمجية الأساسية المستخدمة في برمجة الشبكات
- **2.1 الأهداف**
- يتوقع من الدارس بعد الانتهاء من دراسة هذه الوحدة أن يكون قادرا على أن:

1. يستخدم الملفات التسلسلية وملفات الوصول العشوائي في برمجة الشبكات.
2. يتعامل مع المنافذ Ports في تطبيق بسيط0.
3. يستخدام المقابس Sockets في كل من الخادم والمخدوم.
4. يستخدم البرمجة متعددة المسالك Multi-Threading.
5. يتعرف على آليات توزيع ومعالجة الكينونات في برمجة الشبكات .
6. يكتب برمجيات توضح الفعاليات المذكورة أعلاه.

2. مجاري الإدخال/الإخراج IO Streams

- تتبادل البرامج بياناتها المحرفية والبيانية والصوتية مع المصادر الخارجية مثل الملفات والأقراص المرنة والصلبة والمقابس من خلال ما يطلق عليه مجاري الإدخال/الإخراج IO Streams.
- يبين الشكل (1) تمثيلاً للكيفية التي يتم بها تبادل البيانات عبر المجاري ، إذ أن البرنامج يبدأ أولاً بفتح مجرى له عندما تظهر الحاجة لتبادل البيانات ليقوم
- بالقراءة من المصدر (آ-مجرى إدخال Input Stream) أو
- بالكتابة إلى الهدف (ب-مجرى إخراج Output Stream).



الشكل (1): مجاري الإدخال والإخراج وعمليات القراءة من المصدر أو الكتابة إلى الهدف

- **1.2 المجاري ولغة جافا**
- تتميز لغة جافا بمنظومة متكاملة للتعامل مع المجاري، حيث تم تخصيص الحزمة جافا البرمجية java.io التي تحتوي على العديد من الأصناف التي تخدم القراءة والكتابة عبر المجاري. تتميز لغة جافا بتمكينها التعامل مع نوعين من البيانات:

1. تشكيلات محارف يشغل كل منها 16 خانة ثنائية 16-bit Characters يغطيها الصنفان القارئ Reader والكاتب Writer.
2. تشكيلات ثمانيات (بايتات) يشغل كل منها ثمانية خانات ثنائية 8-bit bytes يغطيها الصنفان مجرى الإدخال InputStream ومجرى الإخراج OutputStream

2.2 الصنفان القارئ والكاتب Readerو Writer

- يعتبر الصنفان القارئ والكاتب صنفان مجردان Abstract Classesلمجاري المحارف في الحزمة البرمجية java.io، يمكن أن يورثا ولا يمكن أن نولد منهما كيانات.
- يقدم هذان الصنفان للمبرمجين واجهة برمجة تطبيقات API مع إمكانية تحقيق قارئات أو كاتبات لمجاري تقرأ أو تكتب محارف يشغل كل منها 16 خانة ثنائية 16-bit Characters.

3.2 الصنفان مجرى الإدخال InputStream ومجرى الإخراج OutputStream

- فيما يتعامل الصنفان Reader و Writer مع محارف بعرض 16 خانة ثنائية 16-bit characters فإن الصنفين مجرى الإدخال InputStream ومجرى الإخراج OutputStream تتعامل مع البايتات المكونة من ثمانية خانات ثنائية 8-bit bytes.

- وفيما يلي أهم الأصناف المشتقة من الصنفين InputStream و OutputStream مع شرح مبسط عن الوظائف التي تقوم بها تلك الأصناف المشتقة:
- **1.3.2 الصنفان مجرى** إدخال الملف **و** مجرى إخراج الملف
- يقرأ مجرى إدخال الملف FileInputStream الدخل من ملف على شكل بايتات متتالية، بينما يسمح مجرى إخراج الملف FileOutputStream بكتابة بايتات إلى الملفات.
- **2.3.2 الصنفان مجرى ادخال إنبوبي و مجرى إخراج إنبوبي**
- تمثل الانابيب Pipes قنوات اتصال متزامنة Synchronized Communication Channels بين المسالك، حيث يفيدنا **مجرى ادخال إنبوبي** PipedInputStreamوهوالصنف المشتق من الصنف مجرى الادخال InputStream للمسلك بقراءة البيانات من قناة الاتصال المتزامنة، بينما يسمح مجرى إخراج إنبوبي PipedOutputStreamبكتابة البيانات إلى المسلك عبر الناقل.

- **3.3.2 الصنفان مجرى ترشيح إدخال و مجرى ترشيح إخراج**
- يقصد بالترشيح Filtering بأن هذا المجرى الذي تم انتقاؤه يتمتع بفعاليات إضافية
- **4.3.2 الصنف Line Number Input Stream مجرى إدخال رقم السطر**
- **5.3.2 الصنفان DataInputStreamو DataOutputStreamمجريي إدخال وإخراج البيانات**
- **6.3.2 الصنفان BufferedInputStreamو BufferedOutputStreamمجريي الإدخال والإخراج مع التخزين**

- **7.3.2 الصنف PushbackInputStream مجرى الاعادة**
- الذي يسمح بإعادة آخر بايت تمت قراءتها إلى المجرى كما لو أنها لم تقرأ ويجري استخدام هذه التقنية في أنماط محددة من المحللات القواعدية parsers.
- الصنف PrintStream مجرى طباعة
- **8.3.2 الصنفان ObjectInputStream و ObjectOutputStream مجرى إدخال الكيانات ومجرى إخراج الكيانات**

4.2 ملفات الوصول التسلسلي Serial access files

- يتم تخزين البيانات في ملفات الوصول التسلسلي في مواقع متجاورة فيزيائياً. تستخدم ملفات الوصول التسلسلي غالباً لتخزين كميات قليلة نسبياً من البيانات ولمدة قصيرة، وتتميز ببنية داخلية ثنائية Binary أو نصية Text.
- ولنأخذ مثلاً التعامل مع الملف النصي. يحتاج التعامل مع الملف النصي إلى قارئ ملفات FileReader لعملية إدخال الكيانات، وكاتب ملفات FileWriter لعملية إخراج الكيانات. ويجري تحميل التابع البناء تحميلاً زائداً Overloaded constructor من خلال تمرير إما سلسلة محرفية String(تحمل اسم الملف) أو كيان ملف File Object(يبنى من اسم الملف) لهذا التابع البناء كما يبين المثال التالي:

FileReader fileIn = new FileReader ("filename.dat");
FileWriter fileOut = new FileWriter ("result.dat");
أو إنشاء كيان أولاً:
File inFile = new File ("input.txt");
File outFile = new File ("output.txt");
ومن ثم استخدام قارئات وكاتبات الملفات:
FileReader in = new FileReader (inFile);
FileWriter out = new FileWriter (outFile);
لا يزودنا الصنفين FileReader و FileWriter بكافة الفعاليات التي تسهل علينا تبادل البيانات مع الملفات مما يجعلنا نغلف استخدام القارئات والكاتبات بأصناف أخرى مثل قارئ الخزن BufferedReader وكاتب الطباعة PrintWriter كما يلي:
BufferedReader input = new BufferedReader(new FileReader("in.txt"));
PrintWriter output = new PrintWriter(new FileWriter("out.txt));
لنتمكن من استخدام الطرق التي تخص الأصناف المغلفة مثل readLine من الصنف BufferedReader و print أو println من الصنف PrintWriter كما يلي:
BufferedReader in = new BufferedReader(new FileReader("infile.dat"));
String input = in.readline();
System.out.println("value read: " + input);
File out = new File("outFile.dat");
PrintWriter output = new PrintWriter(new FileWriter(out);
Out.println("Test Output");

مثال (1):

- يبين المثال (1) استخدام المجريين DataOutputStream و FileOutputStream بشكل تسلسلي ضمن ما يدعى بسلسلة المجاري Stream chain لتحقيق كتابة مجموعة متغيرات من نوع int و char و Boolean و double على شكل بايتات ضمن ملف "test.tmp". وفي الجزء الثاني من البرنامج نسلسل DataInputStream و FileInputStream لكي نستعيد المتغيرات من البايتات الموجودة في الملف.

```
//DataStreamsTest.java
//الكتابة إلى ملف باستخدامDataOutputStream
//ومن ثم قراءة الملف باستخدامDataInputStream
import java.io.*;
public class DataStreamsTest {
   public static void main(String[] args) throws IOException
   {
     // create holders to read file input
      boolean bool;
      int i;
      double dbl;
      String str = null;
      FileOutputStream out = null;
      DataOutputStream outdf = null;
      FileInputStream in = null;
      DataInputStream indf = null;
//DataOutputStream Example
      // make a generic output byte stream
      out = new FileOutputStream("test.tmp");
      // attach a DataOutputStream to it so more complex items could be written
      outdf = new DataOutputStream(out);
      if ( outdf!=null )
      {
      outdf.writeBoolean(true);
      System.out.println("Write a boolean");
          outdf.writeInt(34);
          System.out.println("Write an int");
      outdf.writeDouble(3.14159);
      System.out.println("Write a double");
      outdf.writeBytes("Hello");
      System.out.println("Write a String");
      outdf.close(); // close and free up system resource.
      out.close();  // close in reverse order; FileOutputStream last
      }
```

```
if ( indf!=null )
        {
       bool = indf.readBoolean();
            i = indf.readInt();
            dbl = indf.readDouble();
            str = indf.readLine();
  System.out.println("\n Read\n boolean: " + bool + ",\n int: " + i + ",\n double: " + dbl + ",\n and String: " + str);
   indf.close(); // close and free up system resource.
            in.close();  // close in reverse order; FileInputStream last
       }
    }
```

5.2 ملفات الوصول العشوائي Random access Files

- تسمح ملفات الوصول العشوائي Random access Files بالوصول غير التسلسلي أو العشوائي لمحتوى الملفات
- مثلاً أن نفتح ملفاً باسم netpro.txt للقراءة عن طريقة التعليمة التالية:
- new RandomAccessFile("netpro.txt", "r");
- حيث يشير الحرف 'r' إلى أن الملف مفتوح للقراءة فقط، وإذا أردنا أن نفتح الملف للقراءة والكتابة معاً يمكننا استخدام التعليمة:
- new RandomAccessFile("netpro.txt", "rw");
- ويمكننا بعد فتح الملف استخدام التعليمات read و write بأشكالها لكي نقوم بإدخال أو إخراج البيانات من الملف.

تمتلك الطريقة RandomAccessFile ثلاث طرق إضافية ذات ميزات جيدة في التعامل مع الملفات هي:

1. تخطى بايتات skipBytes التي تحرك مؤشر الملف إلى الأمام بعدد البايتات التي يتم تخطيها.
2. البحث seek التي تضع المؤشر قبل البايت المبحوث عنها.
3. الحصول على قيمة مؤشر الملف getFilePointer التي تعيد موقع البايت التي يشير إليها مؤشر الملف.

- **مثال (2) على الطالب 62**

- **طرق الملفات**
- تتعرف طرق الصنف File على:
- مواصفات الملفات مثل اسم الملف وحجمه بالبايت.
- الحالات المختلفة التي يمر بها الملف مثل إمكانية القراءة من الملف أو الكتابة فيه.
- ويحتوي الجدول (1) أهم طرق الصنف File وشرح مبسط عن وظيفتها.
- النظر ص 64

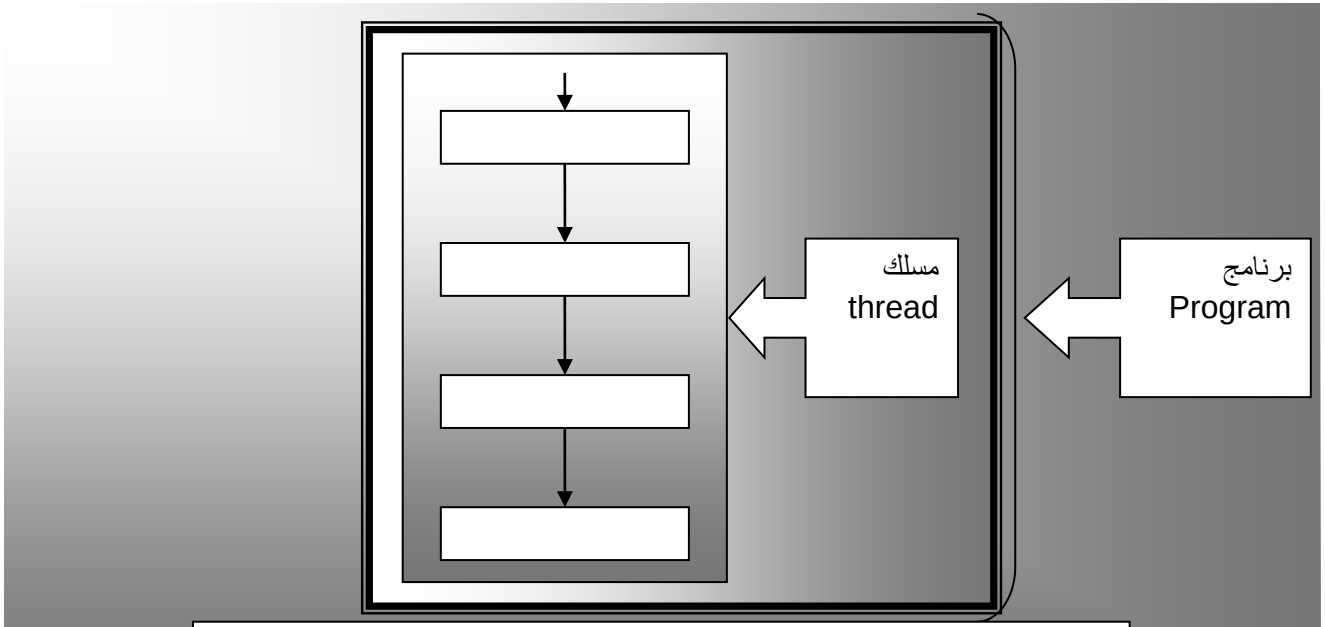
- **3. المنافذ Ports**
- يطرأ على الذهن أول ما يطرأ عند ذكر مصطلح المنفذ أنه مفهوم داراتي مرتبط بالعتاد الصلب المكون للحاسوب وأنها معبر للإشارات عند الاتصال بين العناصر الفيزيائية بين الحواسيب في شبكة.
- وحقيقة الأمر أن مصطلح المنفذ في دراستنا هنا لا يعبر عن عنصر عتادٍ صلبٍ، وإنما عن مفهوم مجرد يسمح للمبرمج بالاستفادة من وصلات الربط التي تبنى بين العقد المختلفة في منظومة الشبكة الحاسوبية.
- والمنفذ Portتعريفاً هي وصلة منطقية يجري تمييزها برقم في المجال من 1 إلى 65535، دون أن يكون لهذا الرقم أي ارتباط بعدد الوصلات المادية مع الحاسوب. فعدد الوصلات المادية التي تربط الحاسوب مع الأجهزة الأخرى هو غالباً وصلة وحيدة، رغم وجود هذا العدد من المنافذ التي توضع تحت تصرف المبرمج. يجري حجز أرقام المنافذ من 1 حتى 1023 للخدمات القياسية الأساسية.

4. المقابس Sockets

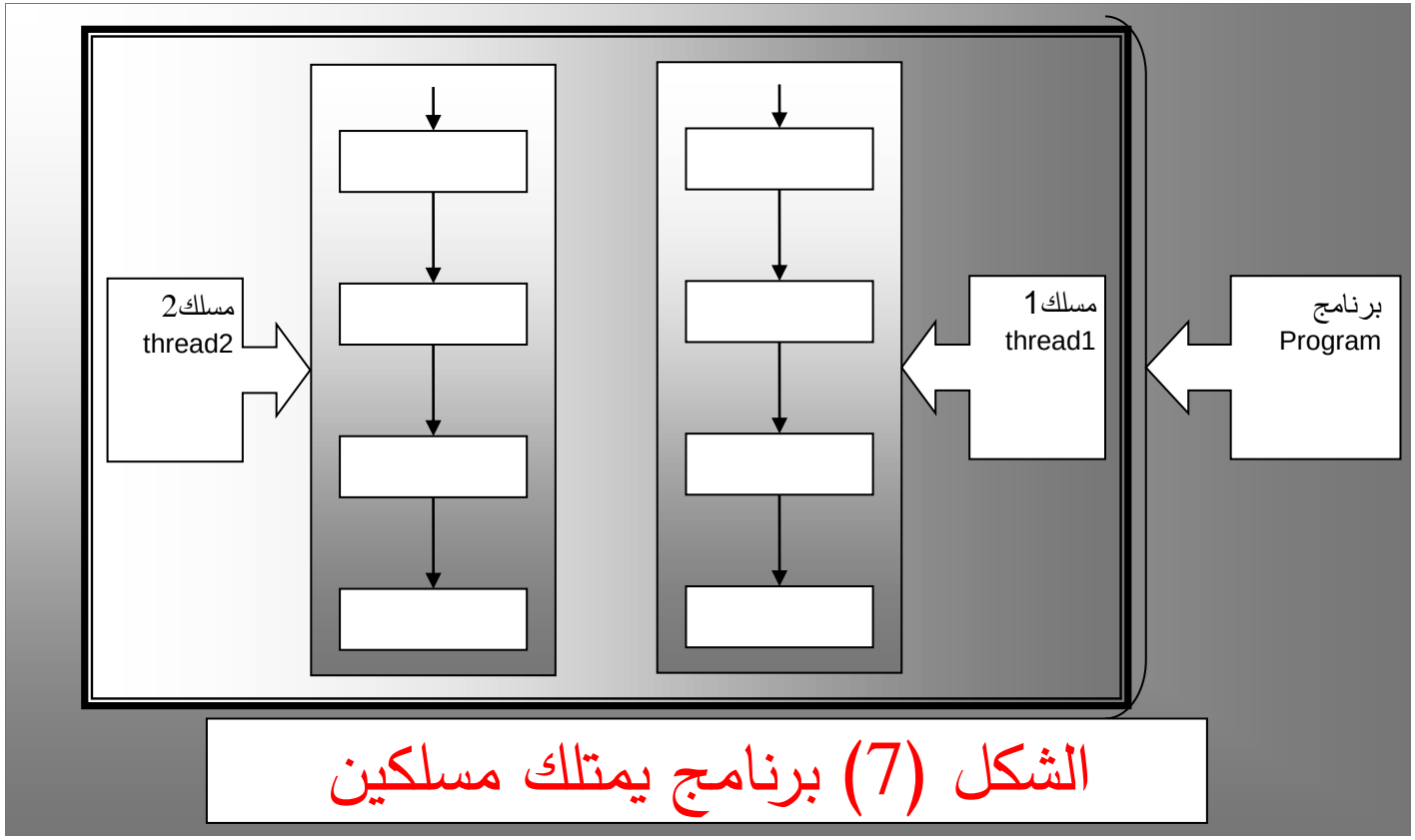
- يعرَف المقبس (Socket) بأنه بنية معطيات مجردة تستخدم لإنشاء قناة اتصال Channel (نقطة وصل) لتبادل المعلومات بين عمليات غير مرتبطة ببعضها البعض.
- والمقابس هي شكل من أشكال الاتصال الداخلي بين العمليات IPC (Inter-Process Communication) الذي يصل بين العمليات في نظام حاسوبي وحيد أو بين العمليات في أنظمة حاسوبية مختلفة.

5. المسلك Thread

- والمسلك Thread يشبه البرنامج التسلسلي من حيث أنه يمتلك بدايةً و خطوات تنفيذية متتالية ونهاية وإمكانية تنفيذ خطوة برمجية وحيدة في لحظة ما. إلا أن المسلك ليس برنامجاً وليس بمقدوره أن يدخل التنفيذ إلا إذا كان ضمن برنامج كما يبين الشكل (6)



الشكل (6): العلاقة بين المسلك الوحيد والبرنامج



الشكل (7) برنامج يمتلك مسلكين

- وهكذا فإننا نستطيع تعريف المسلك Thread على أنه تتابع وحيد للتعليمات المتحكمة بإحدى الفعاليات ضمن برنامج. فإذا كان هنالك حاجة لأكثر من تتابع فإننا نتكلم عن تعدد المسالك Multi-Threading.

1.5 دورة حياة المسلك Thread life cycle

- لا يستطيع المسلك أن يقوم بتنفيذ تتالي تعليماته من نفسه، فهو يخضع لسيطرة العملية الأم Parent Process التي تحدد تفاصيل تتالي التنفيذ.
- ولا بد لنا قبل الشروع بتفصيل دراستنا للمسالك من التنويه إلى أن لغة جافا تستخدم آلية هامة جداً هي المزامنة synchronize لكي نضمن بأن مسلكاً واحداً في لحظة ما، هو فقط الذي سينفذ. وأن المزامنة في لغة جافا يمكن أن تشمل الطرق أو بلوك من التعليمات. ولكي نحدد الأجزاء الذي نرغب بمزامنتها فإننا نستخدم الكلمة المفتاحية synchronized كما سنرى في أمثلتنا التالية.

- **2.5 مسلك جديد new Thread يدخل التنفيذ Running**
- يدخل المسلك حالة مسلك جديد new عند إنشاء الكيان مسلك thread. ويدخل التنفيذ عند الاقلاع بالطريقة ()start، وهنالك طريقتان لإنشاء مسلك جديد هما:
- **1.2.5 وراثة الصنف Thread**
- يبين المثال (4) حالة وراثة الصنف Thread. يعتبر الصنف Thread الصنف الأساس Super Class والصنف FirstMethod الصنف المشتق أو الوارث للصنف الأساس. تمثل الطريقة ()run نقطة بداية تنفيذ أو ما يمكن أن نطلق عليه main المسلك. يجري

```
class FirstMethod extends Thread
{
public void run() {
for ( int count = 0; count < 4; count++)
System.out.println( "Message " + count +
" From: Uni" );
}
public static void main( String[] args )
{
FirstMethod parallel =
new FirstMethod();
System.out.println( "Create the thread");
parallel.start();
System.out.println( "Started the thread" );
System.out.println( "End" );
}
}
```

Output

```
Create the thread
Message 0 From: Uni
Message 1 From: Uni
Message 2 From: Uni
Message 3 From: Uni
Started the thread
End
```

المثال (4): إنشاء مسلك عن طريق وراثة الصنف thread

- إنشاء كيان من الصنف FirstMethod باسم Parallel وتطبيق الطريقة start عليه، والتي بدورها تستدعي الطريقة ()run وتقلع بتنفيذ المسلك المنشأ.
- ولا بد من ملاحظة أن تطبيق الطريقة ()run للإقلاع بالبرنامج لا يُدخل المسلك دورة حياته المذكورة آنفاً وإنما يعتبر تنفيذ اعتيادي لهذه الطريقة.
- **2.2.5 إنشاء كيان من الصنف Thread**
- يبين المثال (5) استخدام الواجهة البينية لقابلية التنفيذ Runnable Interfaceالتي تمتلك طريقة وحيدة هي ()run التي تلعب نفس الدور التي تلعبه ()run في الصنف المشتق في المثال السابق، ومن ثم يجري بناء كيان من الصنف Thread وتمريره إلى إلى التابع البناء في الصنف المستخدم للواجهة، وأخيراً يجري تطبيق الطريقة ()start على الكيان المنشأ.

```
class SecondMethod implements Runnable
{
public void run() {
for ( int count = 0; count < 4; count++)
System.out.println( "Message " + count + " From: Faculty");
}
public static void main( String[] args )
{
SecondMethod notAThread = new SecondMethod();
Thread parallel = new Thread( notAThread );
System.out.println( "Create the thread");
parallel.start();
System.out.println( "Started the thread" );
System.out.println( "End" );
}
}
```

Output

```
Create the thread
Message 0 From: Faculty
Message 1 From: Faculty
Message 2 From: Faculty
Message 3 From: Faculty
Started the thread
End
```

المثال (5) إنشاء مسلك عن طريق استخدام الواجهة البينية

- **3.5 الانتقال إلى حالة متوقف (مؤقتاً) Not running**
- ينتقل المسلك إلى حالة التوقف المؤقت من حالة التنفيذ وذلك بحجبه من خلال مجموعة طرق أهمها هي **join()**، **sleep()**، **wait()**و **yield()** والتي تسمح بحجب المسلك بعدة طرق ولأسباب مختلفة نوجزها فيما يلي:
- **1.3.5 محجوب بانتظار حدث Blocked awaiting event**
- **المثال (7): على الطالب**
- **2.3.5 محجوب بواسطة كيان إقفال Blocked by locked object**
- **المثال (8): على الطالب**
- **4.3.5 محجوب بسبب تعليمة انتظار Blocked on wait()**