الدليل العملي لمقرر
برمجة الشبكات

إعداد: د. م. سامر حسني جالودي
جامعة القدس المفتوحة - فرع نابلس

التجربة الأولى

إدخال وإخراج البيانات المتعلقة بالملفات

File Data Input and Output

مقدمة

عزيزي الدارس، هناك عدة طرق لتصنيف (Compile) وتنفيذ (Run) برامج جافا. منها:

1. باستخدام بيئة التطوير (eclipse) والذي يمكن تحميله (download) من الموقع التالي:
http://www.eclipse.org/downloads/
2. باستخدام بيئة التطوير (JBuilder)، ولكن نسخة تجريبية، والذي يمكن تحميله من الموقع التالي:
https://downloads.embarcadero.com/free/jbuilder
3. باستخدام بيئة التطوير (JCreator)، والذي يمكن تحميله من الموقع التالي: http://www.jcreator.com/
4. باستخدام بيئة التطوير (NetBeans) والذي يمكن تحميله من الموقع التالي: https://netbeans.org/downloads/
5. باستخدام موقع شركة أوراكل (Oracle)، يمكن تحميل معدات تطوير جافا (download JDK) :
http://www.oracle.com/technetwork/java/javase/downloads/index.html ومن ثم بعد ذلك يمكن العمل على بيئة JDK بشكل مباشر أو من خلال برنامج (Textpad) للتسهيل أثناء التصنيف والتنفيذ من الموقع:
http://www.textpad.com/download/

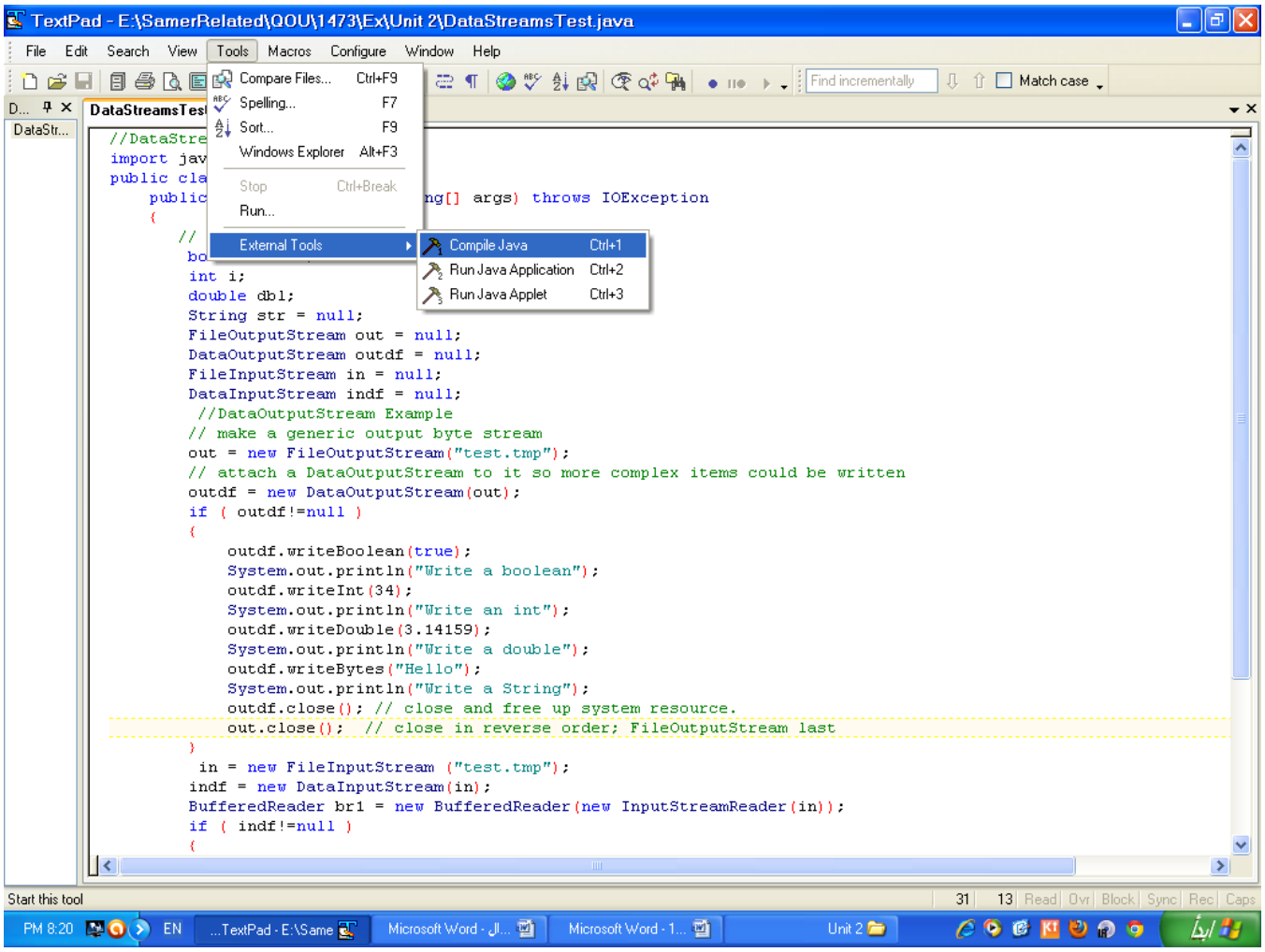
سيتم اعتماد الطريقة الأخيرة لتنفيذ التجارب لهذا الدليل العملي وذلك لغايات تحقيق الأهداف المرجوة من المقرر. مع العلم أنه بإمكانك استخدام أي واجهة تطوير أخرى (IDE) تناسبك، وذلك بالتنسيق مع مشرفك الأكاديمي.

الأهداف

1. استخدام المجريين (DataOutputStream) و (FileOutputStream) بشكل تسلسلي لتحقيق كتابة مجموعة من متغيرات من نوع عدد صحيح (int) ومتسلسلة رموز (String) ومنطقي (Boolean) و مزدوج (double)، وذلك على شكل بايتات ضمن ملف (test.tmp).
2. استخدام المجريين (DataInputStream) و (FileInputStream) لاستعادة المتغيرات (قراءة) من البايتات الموجودة في الملف (test.tmp).
3. استخدام الصنف (RandomAccessFile) من اجل كتابة البيانات نفسها المكتوبة باستخدام المجريين (DataOutputStream) و (FileOutputStream) ثم استرجاع تلك البيانات باستخدام نفس الصنف وهو (RandomAccessFile).
4. تكتب وتنفذ برنامج يستخدم طرق الملف (File).

خطوات عمل الجزء الأول

نبدأ عزيزي الدارس بالهدفين الأول والثاني حيث يمكن تحقيقهما من خلال البرنامج التالي. انقل الكود البرمجي إلى واجهة Textpad ثم احفظه بالامتداد (.java)، انقر الأمر (Compile Java) كما هو موضح في الشكل التالي، ثم انقر الأمر (Run Java Application).




```
//DataStreamsTest.java
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
        }
        in = new FileInputStream ("test.tmp");
        indf = new DataInputStream(in);
        BufferedReader br1 = new BufferedReader(new InputStreamReader(in));
        if ( indf!=null )
        {
            bool = indf.readBoolean();
            i = indf.readInt();
            dbl = indf.readDouble();
            str = br1.readLine(); // we cannot use indf to read a string, because it is deprecated
            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.
            br1.close();
            in.close();  // close in reverse order; FileInputStream last
        }
    }
}
```

سؤال (1): بعد التنفيذ، اشرح آلية عمل البرنامج السابق؟

سؤال (2): جزّء البرنامج السابق إلى جزأين منفصلين (برنامجين)، الأول يعمل على الكتابة في الملف (test1.tmp) وبالتالي يحقق الهدف الأول، والثاني يعمل على قراءة الملف (test1.tmp) وبالتالي يحقق الهدف الثاني؟

خطوات عمل الجزء الثاني

الهدف الثالث يتم تحقيقه من خلال البرنامج التالي:

```
//RandomAccessTest.java
//Write to a file then read it using just
//RandomAccessFile object

import java.io.*;

public class RandomAccessTest
{
   public static void main(String[] args) throws IOException
   {
      // get random access to file "test" (opens at beginning of file)
      RandomAccessFile raf = new RandomAccessFile("test2.tmp","rw");
      if ( raf!=null )
      {
          // write true, 34, 3.14159, "Hello" in binary to the file
         raf.writeBoolean(true);
         System.out.println("Wrote a boolean");
         raf.writeInt(34);
         System.out.println("Wrote an int");
         raf.writeDouble(3.14159);
         System.out.println("Wrote a double");
         raf.writeBytes("Hello");
         System.out.println("Wrote a String");

         raf.seek(0);// rewind the file

         boolean bool = raf.readBoolean();
```

```
        int i = raf.readInt();
        double g = raf.readDouble();
        String str = raf.readLine();
        System.out.println("Read\nBoolean: "+bool+ "\nint: "+i+"\ndouble: "+g+"\nString: "+str);
        if ( i!=34 || g!=3.14159 || !str.equals("Hello") )
        System.out.println("Problem with RandomAccessFile");
        else
           System.out.println("RandomAccessFile Test Complete");
        raf.close();  // close and free up system resource.
     }
   }
}
```

سؤال (3): نفّذ ثم وضح آلية عمل البرنامج السابق؟ ثم بيّن ما يلي:

أ - ما الهدف من استخدام "rw" عند استدعاء الصنف البناء (RandomAccessFile)؟

```
RandomAccessFile raf = new RandomAccessFile("test2.tmp","rw");
```

ب -ما الهدف من استخدام الدالة ()seek؟

سؤال (4): اعمل على تجزئة البرنامج إلى جزئين منفصلين (برنامجين)، البرنامج الأول يعمل على الكتابة على ملف (test3.tmp) والبرنامج الثاني يقرأ القيم المكتوبة في الملف (test3.tmp)؟

خطوات عمل الجزء الثالث

أكتب برنامجا بلغة جافا يعمل على استخدام طرق الملف (File) حيث:

1. يتأكد أولا من وجود الملف أو الدليل الذي تبحث عن مواصفاته.
2. يميز فيما إذا كان الاسم يمثل ملف أو مجلد، وهل هو قابل للقراءة أم لا.
3. يستعرض محتوياته إن كان مجلدا.
4. وإن لم يكن مجلد، فإنه يعطي طول الملف بالبايت.

التجربة الثانية

التطبيقات متعددة المسالك

Multithreaded Applications

مقدمة

عزيزي الدارس، عندما نتكلم عن البرمجة لا بد لنا أن نستعرض تقنيات البرامج وخصائصها من خلال تصنيف يأخذ بعين الاعتبار طريقة تعاملها مع العمليات والمهمات التي تشكل الوحدة الأساسية في البناء البرمجي لتطبيقات الخادم/الزبون والبرمجة الموزعة وأهم هذه البرامج هي:

1. البرنامج التسلسلي Sequential Program : سلسلة متتابعة من الفعاليات (التعليمات والمتغيرات) المولدة لنتيجة، نطلق عليها العملية Process أو المهمة Task أو المسلك Thread.
2. البرنامج التعاوني Concurrent Program : عمليتان Processes أو أكثر تتصلان مع بعضهما البعض لإنجاز عملاً مشتركاً من خلال متغيرات تشاركية Shared Variables وتؤديان (تودي) عملهما من خلال تمرير الرسائل وتحتاجان من أجل ذلك إلى مزامنةٍ Synchronization لفعالياتهما.
3. التطبيقات متعددة المسالك Multithreaded Applications : تنفيذ الفعاليات من خلال استخدام أكثر من مسلك Thread وغالباً ما يتم ذلك من خلال التشارك بزمن المعالجة التابع لوحدة معالجة مركزية واحدة. تعتبر التطبيقات متعددة المسالك وسيلة جيدة لتنظيم التطبيقات البرمجية الحديثة من خلال الخادمات Servers ونظم التقسيم إلى شرائح زمنية Time Sharing systems.
4. التطبيقات المتوازية Parallel Applications : يختص كل معالج في التطبيقات المتوازية بتنفيذ عمليات محددة منوطة به من أجل حل مشاكل ومسائل برمجية كبيرة الحجم بسرعة أكبر.
5. التطبيقات الموزعة Distributed Applications : تتصل العمليات مع بعضها البعض في التطبيقات الموزعة عبر شبكة مما يسمح باستثمار معلومات ومقدرات الحواسيب ذات القدرات الكبيرة مثل الخادمات الكبيرة والمتطورة من خلال الاتصال بها عن بعد وهو موضوع فقرتنا التالية.

تحتاج هذه التطبيقات المختلفة إلى عتاد صلب Hardware كالحواسيب ذات المعالج الوحيد Single Processor والحواسيب متعددة المعالجات Multi Processor والنظم متعددة الحواسيب Multi-Computer إضافةً إلى الشبكات Networks التي تعد الأبطأ بين التقنيات المذكورة سابقاً والأكثر فعالية في النظم التي تحتاج موارد متعددة.

في حال أردت، عزيزي الدارس، تنفيذ عدة تطبيقات (مثلا تصفح الإنترنت عن طريق المتصفح Browser والاستماع إلى الموسيقى وتشغيل برنامج معالج الكلمات MS Word كل هذه التطبيقات على جهازك الحاسوب في آن واحد)، فإن تقسيم وقت المعالج بين هذه التطبيقات هو من مهمة نظام التشغيل وبالتالي يسمى تعدد المهام (Multitasking). ولكن وجود عدة برامج تعمل في نفس الوقت من نفس التطبيق (مثلا في المتصفح Browser يتم الاستماع إلى موسيقى في الإطار الأول من صفحة الويب، وفي الإطار الثاني من نفس الصفحة يتم عرض رسومات Graphics دعائية، ويعمل الشخص القارئ على انزلاق الصفحة لأعلى وأسفل لقراءة المحتويات) فإن هذا يسمى تعدد المسالك (Multithreaded)، حيث كل برنامج يحتاج إلى مسلك Thread مستقل، وبالتالي فإن المتصفح يجب أن يدعم خاصية تعدد المسالك. في هذه التجربة، سيتم كتابة وتنفيذ مجموعة من البرامج على النوع الثالث المذكور أعلاه وهو التطبيقات متعددة المسالك، بحيث تكتب برامج قادرة على تنفيذ مهمتين أو أكثر في نفس الوقت من خلال نفس التطبيق.

الأهداف

تتعرف على دورة حياة المسلك (thread life cycle) ومراحله:

1. مسلك جديد (new thread): تتعرف على الطرق التي يمكن إنشاء مسلك بواسطتها. حيث تشرح آلية عمل كل برنامج. وهذا في الجزء الأول من التجربة.
2. مفعّل (Runnable): وهذه الحالة عندما يدخل المسلك حيز التنفيذ. سيتم توضيحها في كل برنامج من برامج هذه التجربة.
3. تنفيذ (running): يدخل المسلك التنفيذ عند الإقلاع بالطريقة ()start.ونستطيع أن نسأل عن حالة مسلك من خلال الطريقة ()isAlive التي ترجع true إذا كان المسلك في حالة قابلية التفعيل runnable أو كان مفعلاً running في حالة التشغيل.
4. غير مفعّل (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.

5. منته (dead): يخرج المسلك من التنفيذ عادةً بواسطة الطريقة ()exit التي تخرج المسلك من عملية التنفيذ بشكل نهائي وتنقله إلى حالة الموت dead State. كذلك الأمر بالنسبة للطريقة ()stop والطريقة ()interrupt.
6. إنهاء التنفيذ (run() is terminated): عندها يتوقف المسلك عن التنفيذ.

خطوات عمل الجزء الأول

هناك طريقتين لإنشاء المسالك. الأولى عن طريق وراثة الصنف (Thread)، والثانية عن طريق استخدام الواجهة البينية لقابلية التنفيذ (Runnable). نفذ ثم وضح آلية عمل البرنامجين التاليين وذلك لتحقيق الأهداف الثلاثة الأولى وكذلك الخامس والسادس:

البرنامج الأول:

```
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" );
	}
}
```

البرنامج الثاني:

```
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" );
	}
}
```

خطوات عمل الجزء الثاني

أكتب برنامج توضح فيه كيفية إنشاء مسلكين جديدين وإعطائهما أسماء "Uni" و "Faculty" ومن ثم إدخالهما إلى حالة التشغيل running من خلال الطريقة ()start التي تستدعي الطريقة ()run. تقوم الطريقة بتنفيذ كل مسلك مرتين ثم تنقله إلى حالة الانتهاء (الموت dead). في هذا البرنامج، تعطي اسم لكل مسلك ثم تحصل على اسم المسلك قيد التنفيذ باستخدام الطريقة ()getName والطريقة ()currentTread اللتين تسمحان بتحديد المسلك الذي يتم تنفيذه في الطريقة ()run.

سؤال (1): نفذ البرنامج على عدة أجهزة، عدة مرات، ثم وضح دور المُجدْول (Scheduler) في إخراج النتائج بهذا الشكل؟ وهل هناك عوامل أخرى تحدد نتائج التنفيذ؟

خطوات عمل الجزء الثالث

يتم تحقيق الهدف الرابع في هذا الجزء من خلال مجموعة من الأمثلة توضح طرق انتقال المسلك من حالة التنفيذ إلى حالة التوقف المؤقت بعدة طرق كما أسلفنا، وبعدة أسباب منها:

1. السبب الأول: محجوب بانتظار حدث (Blocked Awaiting Event) حيث نستخدم الطريقة ()sleep لتحقيق ذلك كما في البرنامج الأول من هذا الجزء.
2. السبب الثاني: محجوب بواسطة كيان إقفال (Blocked by Locked Object) حيث نستخدم بلوك التزامن (synchronized) كما في البرنامج الثاني من هذا الجزء.
3. السبب الثالث: تسليم التنفيذ لمسلك آخر باستخدام الطريقة ()yield، وهذا كما في البرنامج الثالث من هذا الجزء.
4. السبب الرابع: محجوب بسبب تعليمة انتظار ()wait. كما في البرنامج الرابع من هذا الجزء.

البرنامج الأول: لديك الصنفان (ThreadTester) و (PrintThread). ضعهما في برنامج جافا واحد باسم (ThreadTester.java) في واجهة (Textpad) ثم نفذ البرنامج، حلل النتائج و آلية عمل البرنامج كذلك؟

```
public class ThreadTester
{
  public static void main( String args[] )
  {
    PrintThread thread1, thread2, thread3, thread4;
    thread1 = new PrintThread( "thread1" );
    thread2 = new PrintThread( "thread2" );
    thread3 = new PrintThread( "thread3" );
    thread4 = new PrintThread( "thread4" );
    System.err.println( "\nStarting threads" );
    thread1.start();
    thread2.start();
    thread3.start();
    thread4.start();
    System.err.println( "Threads started\n" );
  }
}
class PrintThread extends Thread
{
  private int sleepTime;
  public PrintThread( String name )
  {
    super( name );
    sleepTime = (int) ( Math.random() * 5000 ); // sleep between 0 and 5 seconds
    System.err.println( "Name: " + getName() + ";  sleep: " + sleepTime );
  }
  public void run()// execute the thread
  {
    try
    {
      System.err.println( getName() + " going to sleep" );
      Thread.sleep( sleepTime ); // put thread to sleep for a random interval
    }
    catch ( InterruptedException exception ) {System.err.println( exception.toString() );}
    System.err.println( getName() + " done sleeping" );// print thread name
  }
}
```

البرنامج الثاني: لديك الصنف (LockExample) التالي. نفذ البرنامج، حلل النتائج و آلية عمله أيضاً من خلال تحليل عمل بلوك التزامن synchronized؟

```
public class LockExample extends Thread
{
  private Lock myLock;
  public LockExample( Lock aLock )
  {
    myLock = aLock;
  }
  public void run()
  {
    System.out.println( "Start run");
    myLock.enter();
    System.out.println( "End run");
  }
  public static void main( String args[] ) throws Exception
  {
    Lock aLock = new Lock();
    LockExample tester = new LockExample( aLock );
    synchronized ( aLock )
    {
      System.out.println( "In Block");
      tester.start();
      System.out.println( "Before sleep");
      Thread.currentThread().sleep( 5000);
      System.out.println( "End Block");
    }
  }
}
class Lock
{
  public synchronized void enter()
  {
    System.out.println( "In enter");
  }
}
```

البرنامج الثالث: لديك الصنف (YieldThread) التالي. نفذ البرنامج، حلل النتائج و آلية عمله أيضاً ثم وضح سبب استخدام الطريقة ()yield؟ وبيّن سبب استخدام الطريقة ()setPriority؟

```
public class  YieldThread  extends Thread
{
  public  void run()
  {
    for(int count = 0;  count < 4; count++)
    {
      System.out.println( count + " From: " + getName() );
      yield();
    }
  }
  public static void main( String[]  args )
  {
    YieldThread  first  = new YieldThread();
    YieldThread  second = new YieldThread();
    first.setPriority( 1);
    second.setPriority( 1);
    first.start();
    second.start();
    System.out.println( "End" );
  }
}
```

البرنامج الرابع: أكتب الكود البرمجي اللازم لتحقيق السبب الرابع من هذا الجزء، من خلال مثال على طريقتي ()wait و ()notify من خلال مسألة المنتج المستهلك الشهيرة بحيث تطبقها من خلال المسالك. سيقوم أحد المسالك بدور المستهلك الذي يقوم بإزالة عنصر من الصف المشترك shared queue بين المستهلك والمنتج. إذا فرغ الصف المشترك فعليه أن ينتظر حتى يقوم المنتج من خلال أحد المسالك بوضع مدخل جديد في الصف المشترك ومن ثم إشعار ()notify مسالك المستهلك المنتظرة بذلك، والتي بدورها ستستهلك ما تم إنتاجه. وسوف يستمر ذلك في مثالنا إلى ما لانهاية. يتألف البرنامج من أربعة أصناف:

أ - صنف المستهلك consumer.

ب - صنف المنتج producer.

ج - صنف الصف المشترك الذي يجري تقاسمه SharedQueue.

د - وأخيراً صنف ProducerConsumerExample الذي يحتوي على برنامج التشغيل ()main. يفعل البرنامج خمسة مسالك اثنان من الصنف consumer وثلاثة من النوع producer. حيث يقوم المنتج (IBM or MAC) بإنتاج منتج يميز برقم إنتاج ابتداءً من الصفر بينما يقوم المستهلك بالحصول على هذا المنتج. وينتظر المستهلك المنتج (حال فراغ الصف المشترك) حتى يقوم بالإنتاج. يشرف على توزيع الفعاليات الصف المشترك الذي يدخل حالة الانتظار ()wait في حال فراغ الصف ويبقى فيها حتى يضيف المنتج جديداً إلى الصف يقوم إثرها الصف بإعلام المستهلك من خلال ()notify.

```
// Consumer.java
class Consumer extends Thread
{
	SharedQueue localMall;
	int sleepDuration;
	public Consumer(String name, SharedQueue input, int speed)
	{
		setName(name);
		localMall = input;
		sleepDuration = speed;
	}
	public void run()
	{
		try
		{
			while (true) // Shop until you drop
			{
				System.out.println(getName()+" got "+localMall.get());
				sleep( sleepDuration );
			}
		}
		catch ( InterruptedException endOfCreditCard ){ return; }
	}
}
//=====================================================
// Producer.java
public class Producer extends Thread
{
	SharedQueue factory;
	int workSpeed;
	public Producer( String name, SharedQueue output, int speed )
	{
		setName(name);
		factory = output;
		workSpeed = speed;
	}
	public void run()
	{
		try
		{
			int product = 0;
			while (true) // work forever
```

```
                        {
                                System.out.println(getName()+" produced "+product);
                                factory.append(getName()+String.valueOf(product));
                                product++;
                                sleep( workSpeed);
                        }
                }
                catch ( InterruptedException WorkedToDeath ){ return; }
        }
}
//=========================================================
// SharedQueue.java
import java.util.ArrayList;
public class SharedQueue
{
        ArrayList <Object> elements = new ArrayList <Object>();
        public synchronized void append( Object item )
        {
                elements.add( item);
                notify();
        }
        public synchronized Object get( )
        {
                try
                {
                        while ( elements.isEmpty() )
                        wait();
                }
                catch (InterruptedException threadIsDone )  {return null;}
                return elements.remove(0);
        }
}
//=========================================================
// ProducerConsumerExample.java
public class ProducerConsumerExample
{
        public static void main( String args[] ) throws Exception
        {
                SharedQueue store = new SharedQueue();
                Producer fact1 = new Producer( "IBM", store, 500 );
                Producer fact2 = new Producer( "MAC", store, 1200 );
                Consumer consum1 = new Consumer( "Ahmad", store, 400);
                Consumer consum2 = new Consumer( "Rami", store, 900);
                Consumer consum3 = new Consumer( "Sami", store, 2200);
                fact1.start();
                fact2.start();
                consum1.start();
                consum2.start();
                consum3.start();
        }
}
```

التجربة الثالثة

برمجة بروتوكولات طبقة الشبكة

Network Layer Protocols

مقدمة

تهتم طبقة الشبكة بإيصال الرزم (packets) من المصدر (source) إلى المقصود (destination) ومن الممكن أن يتطلب ذلك المرور بمجموعة من الأجهزة لتوجيه الرزم نحو الجهاز المقصود. تكمن أهمية هذه الشبكة بأنها تتعامل مع العناوين المنطقية(logical addresses) للأجهزة في الوقت التي تتعامل طبقة ربط البيانات مع العناوين الفيزيائية للشبكة (physical addresses).

حتى تحقق طبقة الشبكة الأهداف المنوط بها يجب أن تعرف مجموعة الموجهات المرتبطة مع الشبكة لاختيار المسار المناسب لتوجيه الرزم وتجنب الموجهات ذات الحمولة الزائدة لتقليل التأخير ما أمكن. إضافة لما سبق فعندما يكون المصدر والمقصود تنتمي لشبكتين مختلفتين في التقنية فان طبقة الشبكة تهتم بالتعامل مع الاختلاف لتامين التواصل بينهما.

يعتبر بروتوكول Internet Protocol (IP) حجر الزاوية في بروتوكولات (TCP\IP) وهو البروتوكول الأكثر استخداما لطبقة الشبكة ومعظم وظائف طبقة الشبكة تعتمد على إمكانيات بروتوكول الانترنت (IP) وقد تتغير بروتوكولات طبقة ربط البيانات عدة مرات بما يلائم هذه الشبكات إلا أن بروتوكول طبقة الشبكة يظل نفسه.

وظائف بروتوكول طبقة الشبكة

1. العنونة : تتضمن الترويسة التي يضيفها بروتوكول طبقة الشبكة حقلين يوضح فيهما عنوان المصدر وعنوان الوجهة النهائية للرزمة الذي يختلف عن عنوان الوجهة الذي يأتي في ترويسة بروتوكول طبقة ربط البيانات ويعتمد بروتوكول IP عنوانا على شكل قيمة 32 بت من قبل مسؤول الشبكة أو يعين آليا.
2. التجزئة : قد يتوجب على المخططات البيانية التي تنشئها الشبكة عبور الكثير من الشبكات المختلفة في طريقها إلى وجهتها وقد يكون بروتوكول طبقة ربط البيانات التي تصادفها هذه المخططات البيانية خصائص وإمكانيات مختلفة ومن هذه الإمكانيات الحجم الأقصى للرزمة التي يستطيع البروتوكول حملها.
3. التوجيه هي عملية توجيه المخطط البياني من مصدره عبر شبكة جامعة وصولا إلى وجهته النهائية عبر أفضل مسار ممكن ترتبط الشبكات المحلية مع بعضها لتشكل شبكة واسعة بواسطة موجهات وعمل الموجه هو استلام البيانات الواردة من إحدى الشبكات وإرسالها إلى وجهة معينة على شبكة محلية أخرى وتحتفظ الموجهات بمعلومات عن الشبكة ضمن جداول تخزن في ذاكرتها.
4. التعرف على بروتوكول طبق النقل : كما أن ترويسة بروتوكول طبقة ربط البيانات تحدد بروتوكول طبقة الشبكة الذي ولّد البيانات التي ينقلها تميز ترويسة بروتوكول طبقة الشبكة بروتوكول طبقة النقل الذي استلمت منه البيانات التي ينقلها هذه المعلومات يستطيع النظام المستقبل تمرير البيانات الواردة إلى بروتوكول طبقة النقل الصحيح.

عزيزي الدارس، سوف تحلل مجموعة من البرامج وتكتب أخرى، وذلك من أجل فهم بروتوكولات طبقة الشبكة.

الأهداف

1. تتعرف إلى بروتوكولات طبقة الشبكة.
2. تكتب برنامجا بلغة جافا لإيجاد عنوان (IPv4) لجهاز محلي في شبكة محلية.
3. تكتب برنامجا يعمل على إعداد قائمة بجميع مداخل الشبكة.
4. تكتب برنامجا يعمل على إيجاد اسم المضيف من عنوان الجهاز، والعكس أيضاً.
5. تكتب برنامجا بلغة جافا تستعرض خواص عنوان (IP) معين.

خطوات عمل الجزء الأول

نفذ البرامج الثلاثة التالية وأوجد ناتج التنفيذ؟ ثم اشرح آلية عملها؟
البرنامج الأول: MyDottedQuadAddress.java

```
import java.net.*;
public class MyDottedQuadAddress
{
 public static void main (String[] args)
 {
            try
            {
   InetAddress me = InetAddress.getLocalHost();
   String dottedQuad = me.getHostAddress();
   System.out.println("My address is " + dottedQuad);
  }
  catch (UnknownHostException e)
  {
   System.out.println("I'm sorry. I don't know my own address.");
  }
 }
}
```

البرنامج الثاني: InterfaceLister.java

```
import java.net.*;
import java.util.*;
public class InterfaceLister
{
  public static void main(String[] args) throws Exception
  {
   Enumeration interfaces = NetworkInterface.getNetworkInterfaces();
   while (interfaces.hasMoreElements())
                {
                        NetworkInterface ni = (NetworkInterface) interfaces.nextElement();
                        System.out.println(ni);
                }
        }
}
```

البرنامج الثالث: ReverseTest.java

```
//Given the address, find the hostname
import java.net.*;
public class ReverseTest
{
 public static void main (String[] args)
 {
  try
  {
   InetAddress ia = InetAddress.getByName("192.168.1.100"); // you must give an existing IP address in the network
   System.out.println(ia.getHostName());
  }
  catch (Exception ex)
  {
   System.err.println(ex);
  }
 }
}
```

ملاحظة: عزيزي الدارس، يجب عليك أن تعطي عنوان (IP) موجود في الشبكة وإلّا فإن الناتج سيكون نفس العنوان المعطى.

<u>سؤال (1):</u> استبدل العنوان (IP) بالاسم الذي حصلت عليه من ناتج تنفيذ البرنامج السابق، ثم أعد تنفيذ الخطوات السابقة؟

<u>خطوات عمل الجزء الثاني</u>

أكتب برنامجا بلغة جافا يعمل على إيجاد خواص عنوان (IP)؟

<u>سؤال (2):</u> قارن بين هذا البرنامج والأمر (ipconfig/all) والمستخدم في نافذة الدوس (DOS) لإدارة الشبكة المحلية؟

التجربة الرابعة

برمجة طبقة النقل

Transport Layer Programming

مقدمة

طبقة النقل (Transport Layer) هي الطبقة التي تنقل البيانات وتكون مسؤولة عن تسليمها بشكل سليم خالي من الأخطاء و تقوم بتقسيم المعلومات إلى أجزاء صغيرة، كما تقوم بالتجميع في الجهاز المستقبل وهي المسؤوله عن إشعار الاستلام من الحاسوب المستقبل بأن المعلومات تم استلامها بدون أخطاء. وهي مسئولة كذلك عن تحويل البيانات إلي النقطة المطلوبة باستخدام العنوان الخاص بكل نقطة في الشبكة.

تحوي هذه الطبقة على مجموعة من البروتوكولات والتي تستخدم لتوفير جلسات الاتصال بين الأجهزة على الشبكة وهي مسئولة عن صيانة جودة و دقة المعلومات المنقولة بين الأجهزة، و من أمثلتها :

- الجزء الناقل من بروتوكول ميكروسوفت .NWLink
- الجزء الناقل من بروتوكول NetBEUI
- Sequenced Packet Exchange (SPX) تبادل الرزم المتسلسلة
- Transmission Control Protocol(TCP) بروتوكول التحكم بالنقل

هناك بروتوكولان أساسيان في مستوى طبقة النقل، البروتوكول الأول هو بروتوكول التحكم بالنقل TCP (Transmission Control Protocol)، و هو بروتوكول ذو مستوى عالٍ، يسمح بإعادة إرسال المعطيات التالفة أو المفقودة من قبل المرسل، و هو المسؤول أيضاً عن تسليم الرزم بنفس الترتيب الذي أرسلت به. والبروتوكول الثاني هو بروتوكول مخطط طرود المستخدم UDP (User Datagram Protocol)، يسمح هذا البروتوكول للمستقبل أن يكتشف الرزم التالفة ولكنه لا يضمن وصولها بنفس بالترتيب التي أرسلت به. عادةً ما يكون بروتوكول UDP أسرع بكثير من بروتوكول TCP. يطلق أحيانا على TCP البروتوكول الموثوق (reliable protocol)، أما UDP فهو بروتوكول غيرموثوق (unreliable)، وعمليا فإن البروتوكولات غير الموثوقة ذات فائدة كبيرة لا يمكن الاستهانة بها.

في هذه التجربة، ستعمل عزيزي الدارس على برمجة وفهم آلية عمل بروتوكوليّ TCP وكذلك UDP.

الأهداف

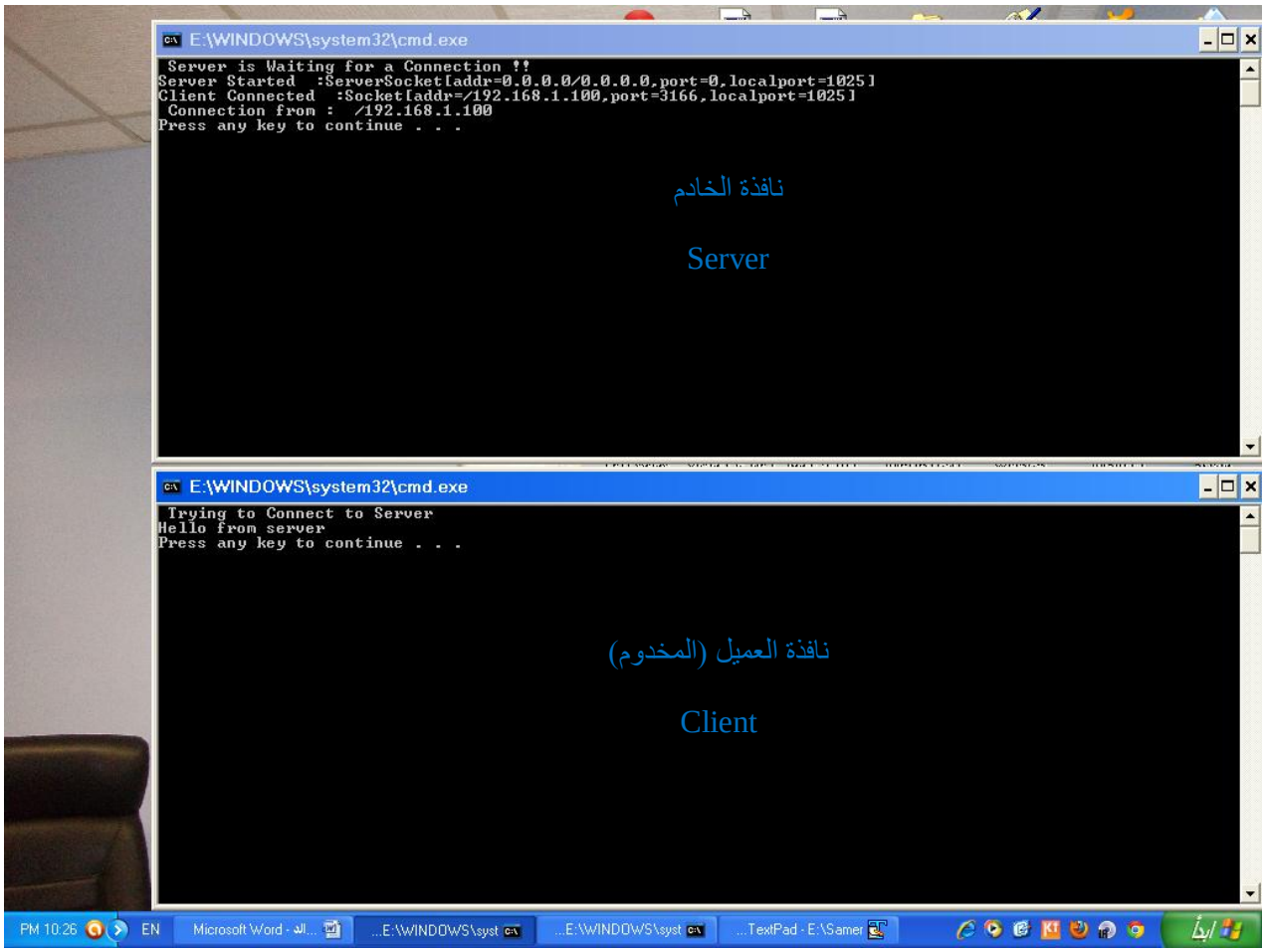
1. تكتب وتنفذ برنامجين عبر بروتوكول (TCP) يمثلان خادم/عميل (Client/Server).
2. تكتب وتنفذ برنامجين عبر بروتوكول (UDP) يمثلان خادم/عميل (Client/Server).
3. تكتب برنامجا بلغة جافا يعمل على إيجاد أول منفذ غير مستعمل.
4. تكتب برنامجا يعرض البوابات التي تستضيف خدمات (TCP).
5. تكتب برنامجا يعمل على إيجاد جميع أسماء واجهات الشبكة (Network Interfaces) والعناوين المنطقية (IP) المرتبطة بها.
6. تكتب وتنفذ برنامجين عبر بروتوكول UDP يمثلان خادم/مخدوم لطريقة الاتصال Multicast.

خطوات عمل الجزء الأول

في هذا الجزء، ستعمل عزيزي الدارس على تصنيف (Compile) وتنفيذ (Run) برنامجين يمثلان عميل/خادم (Client/Server) عبر بروتوكول TCP. البرنامج الأول (Client.java) يمثل العميل، وبالتالي يتم فتحه في نافذة منفصلة عن نافذة البرنامج الثاني (Server.java) والذي يمثل الخادم. في هذه الحالة يجب أن يتم تنفيذ (Run) برنامج الخادم (Server.class) أولا ووضعه في حالة إنصات (Listening)، لأن العميل يسأل (Request) الخادم ويتوقع أن يكون دائما في حالة الجهوزية للرد (Response).

```
import java.lang.*;
import java.io.*;
import java.net.*;
import java.net.InetAddress;

class Client
{
	public static void main(String args[])
	{
		Socket sock=null;
		DataInputStream dis=null;
		PrintStream ps=null;
		System.out.println(" Trying to Connect to Server");
		try
		{
			// to get the ip address of the server by the name
			InetAddress ip =InetAddress.getByName("192.168.1.100"); // "localhost" for example
			// Connecting to the port 1025 declared in the Serverclass creates a socket with the server bind to it.
			sock= new Socket(ip, Server.PORT);
			ps= new PrintStream(sock.getOutputStream());
			ps.println(" Hi from client");
			//DataInputStream is = new DataInputStream(sock.getInputStream());
			//the method readLine is deprecated in the new version, so use BufferedReader
			BufferedReader br1 = new BufferedReader(new InputStreamReader(sock.getInputStream()));
			System.out.println(br1.readLine());
		}
		catch(SocketException e)
		{
			System.out.println("SocketException " + e);
		}
		catch(IOException e)
		{
			System.out.println("IOException " + e);
		}
		// Finally closing the socket from the client side
		finally
		{
			try
			{
				sock.close();
			}
			catch(IOException ie)
			{
				System.out.println(" Close Error :" + ie.getMessage());
			}
		}// finally
  } // main
 }// Class Client
```

```
import java.net.*;
import java.lang.*;
import java.io.*;

public class Server
{
	// port number should be more than 1024
	public static final int PORT = 1025;
	public static void main( String args[])
	{
		ServerSocket sersock = null;
		Socket sock = null;
		System.out.println(" Server is Waiting for a Connection !! ");
		try
		{
			//  Initialising the ServerSocket
			sersock =  new ServerSocket(PORT);
			//Gives the Server Details Machine name, Port number
			System.out.println("Server Started  :"+sersock);
			try
			{
				// makes a socket connection to particular client after  which two way communication take place
				sock = sersock.accept();
				System.out.println("Client Connected  :"+ sock);
				// Receive message from client i.e Request from client
				DataInputStream ins = new DataInputStream(sock.getInputStream());
				// Send message to the client i.e Response
				PrintStream ios = new   PrintStream(sock.getOutputStream());
				ios.println("Hello from server");
				ios.close();
				// Close the Socket connection
				sock.close();
			}
			catch(SocketException se)
			{
				System.out.println("Server Socket problem " + se.getMessage());
			}
			catch(Exception e)
			{
				System.out.println("Couldn't start  " + e.getMessage()) ;
			}
			// Usage of some methods in Socket class
			System.out.println(" Connection from :  " + sock.getInetAddress());
		} // try
		catch(Exception e)
		{
			System.out.println("Port is Busy  " + e.getMessage()) ;
		}
	}// main
}// Server class
```



ملاحظة: في حال أردت عزيزي الدارس تنفيذ البرنامجين على نفس الجهاز، فعليك استبدال عنوان (IP) بالعنوان: "127.0.0.1" وذلك في برنامج العميل Client.java كما يلي:

```
InetAddress ip =InetAddress.getByName("127.0.0.1"); // "localhost" for example
```

سؤال (1): اشرح آلية عمل البرنامجين السابقين (Client.java, Server.java) ؟

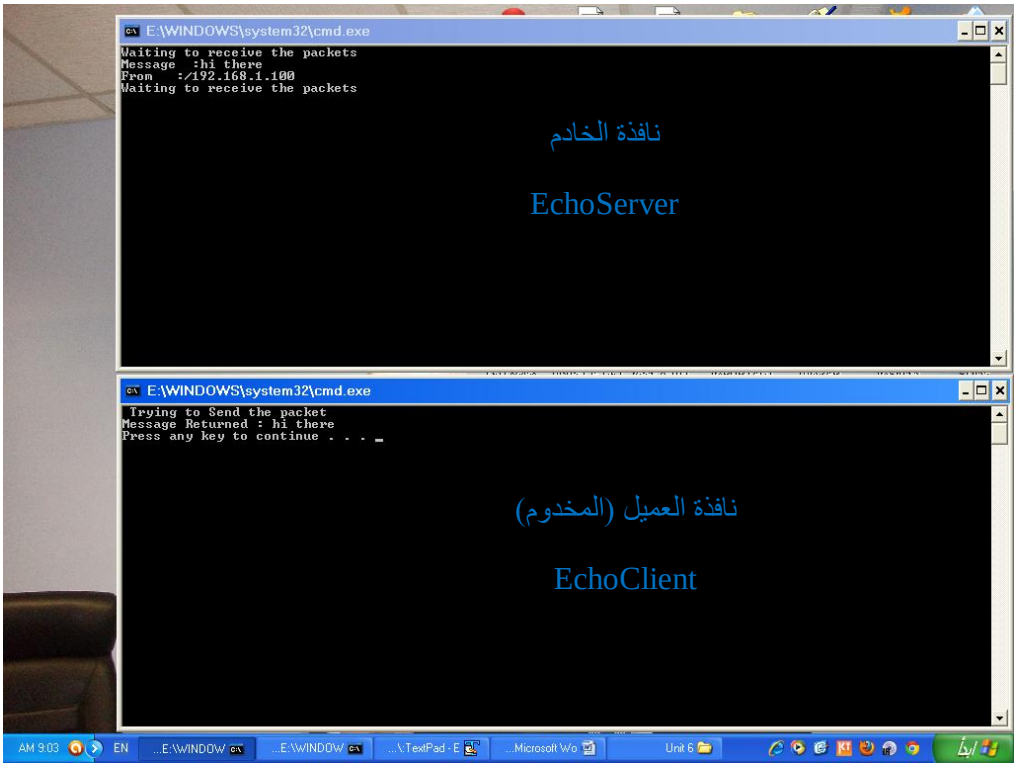
سؤال (2): من خلال حلقة غير منتهية ((while(true)، لكلا البرنامجين الخادم والعميل، عدّل على البرنامجين بحيث يعملان دائما على غرار برنامج الدردشة (Chatting) يخرج العميل من الخدمة إذا كتب المستخدم كلمة (Exit)؟

خطوات عمل الجزء الثاني

في هذا الجزء، ستعمل عزيزي الدارس على تصنيف (Compile) وتنفيذ (Run) برنامجين يمثلان عميل/خادم (Client/Server) عبر بروتوكول UDP. البرنامج الأول (EchoClient.java) يمثل العميل، وبالتالي يتم فتحه في نافذة منفصلة عن نافذة البرنامج الثاني (EchoServer.java) والذي يمثل الخادم. في هذه الحالة يجب أن يتم تنفيذ (Run) برنامج الخادم (EchoServer.class) أولا ووضعه في حالة إنصات (Listenning)، لأن العميل يسأل (Request) الخادم ويتوقع أن يكون دائما في حالة الجهوزية للرد (Response).

```
import java.net.*;
import java.io.*;

public class EchoClient
{
	static final int serverPort  = 1026;
	static final int packetSize = 1024;
	public static void main(String args[]) throws UnknownHostException, SocketException
	{
		DatagramSocket socket; // How we send packets
		DatagramPacket packet; // what we send it in
		InetAddress address = InetAddress.getLocalHost();//"127.0.0.1"; Where to send
		String messageSend = "hi there"; // Message to be send
		String messageReturn; // What we get back from the Server
		byte[] data;
		// Gets the IP address of the Server
		socket = new DatagramSocket();
		data = new byte[packetSize];
		data = messageSend.getBytes();
		// remember datagrams hold bytes
		packet = new DatagramPacket(data,data.length,address,serverPort);
		System.out.println(" Trying to Send the packet ");
		try
		{
			// sends the packet
			socket.send(packet);
		}
		catch(IOException ie)
		{
			System.out.println("Could not Send :"+ie.getMessage());
			System.exit(0);
		}
		//packet is reinitialized to use it for recieving
		packet = new DatagramPacket(data,data.length);
		try
		{
			// Receives the packet from the server
			socket.receive(packet);
		}
		catch(IOException iee)
		{
			System.out.println("Could not receive : "+iee.getMessage() );
			System.exit(0);
		}
		// display message received
		messageReturn = new String (packet.getData());
		System.out.println("Message Returned : "+messageReturn.trim());
	}// main
}// Class EchoClient
```

```
import java.net.*;
import java.io.*;

public class EchoServer
{
	// Initialize  Port number and Packet Size
	static final int serverPort = 1026;
	static final int packetSize = 1024;

	public static void main(String args[]) throws SocketException
	{
		DatagramPacket packet;
		DatagramSocket socket;
		byte[] data;    // For data to be Sent in packets
		int clientPort;
		InetAddress address;
		String str;
		socket = new DatagramSocket(serverPort);

		for(;;)
		{
			data = new byte[packetSize];
			// Create packets to receive the message
			packet = new DatagramPacket(data, packetSize);
			System.out.println("Waiting to receive the packets");
			try
			{
				// wait infinetely for arrive of the packet
				socket.receive(packet);
			}
			catch(IOException ie)
			{
				System.out.println(" Could not Receive:"+ie.getMessage());
				System.exit(0);
			}
			// get data about client in order to echo data back
			address = packet.getAddress();
			clientPort = packet.getPort();
			// print string that was received on server's console
			str = new String(data,0,packet.getLength());
			System.out.println("Message  :"+ str.trim());
			System.out.println("From   :"+address);
			// echo data back to the client
			// Create packets to send to the client
			packet = new DatagramPacket(data,packetSize,address,clientPort);
			try
			{
				// sends packet
				socket.send(packet);
			}
			catch(IOException ex)
			{
				System.out.println("Could not Send "+ex.getMessage());
				System.exit(0);
			}
		} // for loop
	} // main
} // class EchoServer
```



ملاحظة: تمت برمجة العميل على أن يربط (connect) مع الخادم على نفس الجهاز من خلال الجملة البرمجية:

InetAddress address = InetAddress.getLocalHost();//"127.0.0.1"; Where to send

وبالتالي، إن أردت عزيزي الدارس، تنفيذ الخادم على جهاز أخر على الشبكة، فعليك أن تختار عنوان (IP) مختلف عن عنوان الجهاز المحلي (Local Host).

خطوات عمل الجزء الثالث

يتكون هذا الجزء من 3 برامج. عليك كتابتها، وإخراج النتائج وتحليل البرامج وكذلك تحليل النتائج.

أكتب برنامجا بلغة جافا يعمل على إيجاد أول منفذ غير مستعمل؟

أكتب برنامجا يعرض البوابات التي تستضيف خدمات (TCP)؟

أكتب برنامجا يعمل على إيجاد جميع أسماء واجهات الشبكة (Network Interfaces) والعناوين المنطقية (IP) المرتبطة بها؟

خطوات عمل الجزء الرابع

إن عناوين الشبكة (IP addresses) التي تبدأ بأحد الأرقام المحصورة بين 224 و 239، هي عناوين تستخدم في التراسل من نوع البث المتعدد Multicast. هذه التقنية قائمة على الإرسال من نوع واحد-إلى- واحد one-to-one وكذلك عديد-إلى-عديد many-to-many. وتستخدم لإرسال البيانات عبر الشبكة بطريقة فعالة إلى أكثر من جهة، مثلا في بث الفيديو القائم على الدفق video streaming، وإرسال بيانات مالية، بالإضافة لتطبيقات أخرى. في هذا الجزء، سنستخدم العنوان 230.0.0.1 لتوضيح مثال في البث المتعدد.

يتكون هذا الجزء من برنامجين؛ وهما يمثلان الخادم والمخدوم. الخادم في حالة انصات، والمخدوم يرسل رسالة Hello فيرد عليه الخادم بنفس الرسالة echo، ثم التاريخ والوقت الحالي.

البرنامج الأول: UDPTimeServer.java

```
/**
 * Multicast Server: echo data + date
 * UDPTimeServer.java
 * Created: Sun Jul 22 19:21:13 2001
 * @author <a href="mailto: ">Jan Newmarch</a>
 * @version
 **/
import java.io.*;
import java.net.*;
import java.util.Date;
public class UDPTimeServer
{
   public static final String MCAST_ADDR = "230.0.0.1";
   public static final int MCAST_PORT = 9013;
   public static final int DGRAM_BUF_LEN = 512;
   public static void main(String[] args)
   {
                InetAddress group = null;
                try
                {
                        group = InetAddress.getByName(MCAST_ADDR);
                }catch(UnknownHostException e){e.printStackTrace();System.exit(1);}
                MulticastSocket socket = null;
                try
                {
                        socket = new MulticastSocket(MCAST_PORT);
                        socket.joinGroup(group);
                }catch (IOException e){e.printStackTrace();System.exit(3);}
                while (true)
                {
                        try
                        {
                                byte[] buf = new byte[DGRAM_BUF_LEN];
                                DatagramPacket packet = new DatagramPacket(buf, buf.length);
                                socket.receive(packet);
                                System.out.println("Received: " + new String(buf));
                                String date = new Date().toString();
                                buf = date.getBytes();
                                // get client info
                                InetAddress clientAddr = packet.getAddress();
                                int port = packet.getPort();
                                // prepare packet for return to client
                                packet = new DatagramPacket(buf, buf.length, clientAddr, port);
                                System.out.println("Sending: " + new String(buf));
                                socket.send(packet);
                        }catch(IOException e){e.printStackTrace();}
                }
        }
} // UDPTimeServer
```

```
/**
 * MulticastClient.java
 * Created: Sun Jul 22 19:21:13 2001
 * @author <a href="mailto: ">Jan Newmarch</a>
 * @version
 */
import java.io.*;
import java.net.*;
public class MulticastClient
{
	public static final String MCAST_ADDR = "230.0.0.1";
	public static final int MCAST_PORT = 9013;
	public static final int DGRAM_BUF_LEN = 512;
	public static void main(String[] args) throws Exception
	{
		String msg = "Hello";
		InetAddress group = null;
		try
		{
			group = InetAddress.getByName(MCAST_ADDR);
		}catch(UnknownHostException e){e.printStackTrace();System.exit(1);}
		try
		{
			MulticastSocket socket = new MulticastSocket(MCAST_PORT);
			socket.joinGroup(group);
			DatagramPacket hi = new DatagramPacket(
				msg.getBytes(), msg.length(), group, MCAST_PORT);
			System.out.println("Sending: " + msg);
			socket.send(hi);
			// get their responses!
			while (true)
			{
				byte[] buf = new byte[DGRAM_BUF_LEN];
				DatagramPacket recv = new DatagramPacket(buf, buf.length);
				socket.receive(recv);
				byte[] data = recv.getData();
				System.out.println("Received: " + new String(data));
				Thread.sleep(10);
			}
		}catch(IOException e){e.printStackTrace();System.exit(2);}
	// OK ،I'm done talking - leave the group...
	// s.leaveGroup(group); System.exit(0);
	}
}// MulticastClient
```

التجربة الخامسة

برمجة بروتوكولات طبقة التطبيقات

Application Layer Protocols

مقدمة

تتضمن الطبقات الثلاث العليا في بروتوكول OSI- طبقات الجلسة والتمثيل والتطبيقات Session, Presentation and Application Layers- والتي تكافئ طبقة التطبيقات في البروتوكول TCP/IP، الفعاليات الأساسية التي لها ارتباط مباشر بالمستخدم، وتتنوع برمجياتها لتشمل كافة أشكال التطبيقات. تمثل هذه الطبقات بشكل أو بآخر الواجهة البينية بين المستخدم والطبقات الأربعة الدنيا وتتميز بأنها تشمل نطاق واسع من التطبيقات وتستفيد من خدمات كافة الطبقات الدنيا. لما كانت برمجيات هذه الطبقة هي الأقرب للمستثمر وبسبب تنوع تطبيقات الشبكة فإننا سنقتصر في تجربتنا هذه على بروتوكولات منتقاة توضح لنا فعاليات مختلف طبقات OSI الثلاث العليا. سندرس في البداية بروتوكول نقل الملفات File Transfer Protocol (FTP) الذي يسمح للمستخدم بإرسال واستقبال الملفات من وإلى حاسوب مضيف بعيد. ثم بروتوكول نقل البريد البسيط Simple Mail Transfer Protocol (SMTP). ولهذين التطبيقين صلة مباشرة مع الإنترنت مما يمكن الدارس من فهم طبيعة صلة هذه الطبقات مع برمجة الإنترنت.

الأهداف

1. تستخدم الصنف InetAddress والتابع العضو getByName وكذلك getAllByName لطباعة عنوان أو جميع عناوين (IP) مستخدما اسم النطاق.
2. تتعرف إلى فلسفة بروتوكول (FTP) وتكتب برنامجا يحقق تلك الفلسفة.
3. تتعرف إلى آلية عمل بروتوكول (SMTP) وتكتب برنامجا يحقق تلك الآلية.

خطوات عمل الجزء الأول

أكتب برنامجين بلغة جافا:

1. يعمل الأول على طباعة عنوان التشبيك الداخلي لأسم نطاق "www.google.com" باستخدام الصنف InetAddress والتابع العضو getByName؟
2. ويعمل الثاني على طباعة جميع عناوين النطاق "www.gmail.com" مستخدما التابع العضو getAllByName؟

خطوات عمل الجزء الثاني

تعتمد برمجة بروتوكول FTP على برمجة الاتصال بين الزبون Client والخادم Server، وقد خصصت لغة Java الحزمة net البرمجية للتعامل مع الشبكات ، والرزمة الخاصة org.apache للتعامل مع الخادمات. وسنذكر مثالاً على الكيفية التي تتم بها برمجة بعض أوامر هذا البروتوكول من خلال الحزمة net والحزمة io.

```
improt java.io.*
import java.net.*
```

إذ يمكننا انطلاقاً من الحزمة البرمجية net إنشاء اتصال عبر الطريقة connect على البوابة 21 المخصصة لبروتوكول نقل الملفات FTP والاتصال مع خادم FTP كمستخدم مجهول anonymous. ويمكننا تعريّف الطريقتين sendLine و readLine لإرسال أو استقبال أمر من أوامر FTP. ويمكننا بعد ذلك برمجة الاتصال مع الخادم والدخول باسم مستخدم username وكلمة مرور password من خلال المقبس والقارئات readers والكاتبات writers وانتظار الاستجابة من الخادم مع مراعاة ما يلي:

المقبس موصول مسبقاً socket =! null يقتضي إرسال رسالة بأنه قد سبق الاتصال مع البرووكول FTP ويجب أن ينقطع الاتصال قبل معاودة الاتصال. أما إذا لم يكن المقبس موصول مسبقاً فعلينا لإنشاء وصل عن طريق إنشاء مقبس ومن ثم رصد الإجابة القادمة من الطرف الآخر فإذا كانت الإجابة بالرقم 220 على سبيل المثال فإن ذلك يعني تحقيق الربط مع خادم FTP بينما الإجابات التي تأخذ الرقم

3 في الخانتين 2 و 3 مثل الإجابة بالرقم 331 فإن ذلك يعني استجابة غير محددة وليس هنالك ضمان لحدوث الاتصال كما يظهر البرنامج التالي.

وسوف نستعرض فيما يلي بعض أوامر FTP:

أمر PASS: يمكننا بعد حدوث الاتصال إرسال الأمر PASS باستخدام الطريقة sendline وانتظار الإجابة. فإذا كانت الإجابة برقمين 2 و 3 في الخانتين الثانية والثالثة (230 مثلاً) فإن ذلك يعني أن بروتوكول FTP غير قادر على الدخول من خلال كلمة السر المرسلة. وبخلاف ذلك فإن ذلك يعني أنه قد تم الدخول.

أمر QUIT: عند الرغبة بفصل الاتصال نستخدم الطريقة disconnect مرفقةً بإرسال الأمر QUIT وجعل المقبس socket=null. ينتج عن استخدام هذه الطريقة قطع الاتصال مع خادم FTP.

أمر PWD: يعيد هذا الأمر الدليل الفعال لخادم FTP المتصل في المتغير المحلي response من النمط String والطريقة indexOf التي تعيد موقع المحرف '\' الفاصل بين دليل وآخر يتم تشكيل السلسلة dir التي تمثل الدليل الفعال للخادم.

وبنفس الطريقة يمكننا تحقيق جميع أوامر الخدمة والتحكم بالوصول لبروتوكول FTP.

ستقوم عزيزي الدارس في هذا الجزء ببرمجة مخدوم بروتوكول (FTP). ولذلك لا بد من تحميل (download) خادم (FTP Server) من أحد المواقع على الإنترنت، وليكن WingFTPerver من موقع (http://www.wftpserver.com/)، حيث أنه سهل التنصيب (Install) ، والتعامل أيضا في بيئته سهل جدا.

ستعمل على إرسال ملفين من جهاز العميل (Client) إلى جهاز الخادم (Server). الأول ملف نصّي من نوع (txt) والثاني ملف صورة على شكل ثنائيات (Binary) باستخدام البرنامج التالي والذي ينقصه مجموعة من الطرق (Methods):

```
// Written by Dr. Samer Jaloudi For Network Programming course.
// This is an FTP client that helps you upload ASCII/Binary files to remote computers
// You can download any FTP server, such as Wing FTP server, with anonymous username and password
// Configure the server and put it in listening mode (just start it) and run this client

import java.io.*;
import java.net.*;
import java.util.*;

public class MyFTPclient
{
	private Socket socket = null;
	private BufferedReader reader = null;
	private BufferedWriter writer = null;
	private static boolean DEBUG = false;

	public static void main (String args[]) throws Exception
	{
		MyFTPclient ftp1 = new MyFTPclient();
		// connect with remote server: host, port, un, pw
		ftp1.connect("127.0.0.1", 21, "anonymous", "anonymous"); // local host

		File f1 = new File ("myText.txt"); // text file to be transferred
		// change to ASCII mode
		ftp1.ascii();
		// copy this file from my PC to the server default directory (previously configured to use E:/)
		ftp1.stor(f1);

		File f2 = new File ("myImg.jpg"); // image file to be transferred
		// change to binary mode
		ftp1.bin();
		// change the working directory on the ftp server to E:/NewFolder1
		ftp1.cwd("NewFolder1");
```

```
		ftp1.stor(f2);
		// disconnect the connection with the remote server
		ftp1.disconnect();

	}

	public synchronized void connect(String host, int port, String user, String pass) throws IOException
	{
		if (socket != null)
		{
			throw new IOException("FTP is already connected. Disconnect first.");
		}
		socket = new Socket(host, port);
		reader = new BufferedReader(new InputStreamReader(socket.getInputStream()));
		writer = new BufferedWriter(new OutputStreamWriter(socket.getOutputStream()));
		String response = readLine();
		System.out.println("Response:" + response.toString());
		if (!response.startsWith("220 "))
		{
			throw new IOException("unknown response received when connecting to the FTP server: "+ response);
		}
		sendLine("USER " + user);
		response = readLine();
		if (!response.startsWith("331 "))
		{
			throw new IOException("SimpleFTP received an unknown response after sending the user: "+ response);
		}
		sendLine("PASS " + pass);
		response = readLine();
		if (!response.startsWith("230 "))
		{
			throw new IOException("FTP was unable to log in with the supplied password: "+ response);
		}
		// Now logged in.
	}


	private void sendLine(String line) throws IOException
	{
		if (socket == null)
		{
			throw new IOException("FTP is not connected.");
		}
		 try
		{
			writer.write(line + "\r\n");
			writer.flush();
			if (DEBUG)
			{
				System.out.println("> " + line);
			}
		}
		catch (IOException e)
		{
			socket = null;
			throw e;
		}
	}

	private String readLine() throws IOException
	{
		String line = reader.readLine();
		if (DEBUG)
		{
			System.out.println("< " + line);
```

```
            }
            return line;
}

////////////// Method disconnect //////////////
public synchronized void disconnect() throws IOException
{
            try
            {
                        sendLine("QUIT");
            }
            finally
            {
                        socket = null;
            }
}
////////////// Method pwd          //////////////
public synchronized String pwd() throws IOException
{
            sendLine("PWD");
            String dir = null;
            String response = readLine();
            System.out.println("myftp> PWD command response is: " + response.toString());
            if (response.startsWith("257 "))
            {
                        int firstQuote = response.indexOf('\"');
                        int secondQuote = response.indexOf('\"', firstQuote + 1);
                        if (secondQuote > 0)
                        {
                                    dir = response.substring(firstQuote + 1, secondQuote);
                        }
            }
            return dir;
}

////////////// Method CWD        //////////////
public synchronized boolean cwd(String dir) throws IOException
{
   sendLine("CWD " + dir);
   String response = readLine();
   System.out.println("CWD command response is: " + response.toString());
   return (response.startsWith("250 "));
}
/**
* Sends a file to be stored on the FTP server. Returns true if the file
* transfer was successful. The file is sent in passive mode.
*/
public synchronized boolean stor(File file) throws IOException
{
            if (file.isDirectory())
            {
                        throw new IOException("FTP cannot upload a directory.");
            }
            String filename = file.getName();
            return stor(new FileInputStream(file), filename);
}

public synchronized boolean stor(InputStream inputStream, String filename) throws IOException
{
            BufferedInputStream input = new BufferedInputStream(inputStream);
            sendLine("PASV");
            String response = readLine();
            System.out.println("PASV command response is: " + response.toString());
            if (!response.startsWith("227 "))
            {
```

```
                throw new IOException("FTP could not request passive mode: " + response);
        }

        String ip = null;
        int port = -1;
        int opening = response.indexOf('(');
        int closing = response.indexOf(')', opening + 1);
        if (closing > 0)
        {
                String dataLink = response.substring(opening + 1, closing);
                StringTokenizer tokenizer = new StringTokenizer(dataLink, ",");
                try
                {
                        ip = tokenizer.nextToken() + "." + tokenizer.nextToken() + "." + tokenizer.nextToken() + "." +
                        tokenizer.nextToken();
                        port = Integer.parseInt(tokenizer.nextToken()) * 256 + Integer.parseInt(tokenizer.nextToken());
                }
                catch (Exception e)
                {
                        throw new IOException("FTP received bad data link information: " + response);
                }
        }

        sendLine("STOR " + filename);

        Socket dataSocket = new Socket(ip, port);

        response = readLine();
        System.out.println("STOR command response is: " + response.toString());
        if (!response.startsWith("150 "))
        {
                throw new IOException("FTP was not allowed to send the file: " + response);
        }

        BufferedOutputStream output = new BufferedOutputStream(dataSocket.getOutputStream());
        byte[] buffer = new byte[4096];
        int bytesRead = 0;
        while ((bytesRead = input.read(buffer)) != -1)
        {
                output.write(buffer, 0, bytesRead);
        }
        output.flush();
        output.close();
        input.close();

        response = readLine();
        System.out.println("STOR command response when finishing is: " + response.toString());
        return response.startsWith("226 ");
}

 /**
  * Enter binary mode for sending binary files.
  */
public synchronized boolean bin() throws IOException
{
        sendLine("TYPE I");
        String response = readLine();
        System.out.println("Binary, TYPE I, command response is: " + response.toString());
        return (response.startsWith("200 "));
}

 /**
  * Enter ASCII mode for sending text files. This is usually the default mode.
  * Make sure you use binary mode if you are sending images or other binary
  * data, as ASCII mode is likely to corrupt them.
```

```
     */
    public synchronized boolean ascii() throws IOException
    {
        sendLine("TYPE A");
        String response = readLine();
        System.out.println("ASCII, TYPE A, command response is: " + response.toString());
        return (response.startsWith("200 "));
    }

}
```

سؤال (1): اشرح باختصار آلية عمل البرنامج السابق؟

سؤال (2): وضح كيف يتعامل العميل (Client) مع الأمر (PASS)؟

سؤال (3): أكتب الأوامر التي تصدر من العميل (FTP Client) والرد المناسب من السيرفر (FTP Server) على شكل جدول؟

سؤال (4): اشرح بالتفصيل آلية عمل الطريقة (المنهج) ()stor؟

خطوات عمل الجزء الثالث

سنناقش برمجة هذا البروتوكول من خلال توضيح طريقة برمجة بعض أوامره التي تعتمد بشكلٍ أساسي على المقابس. وسنوضح الكيفية التي تعمل بها الأوامر الأربعة الأولى الأساسية للبريد الإلكتروني من خلال زبون SMTP الذي يسمح للمستخدم بإرسال رسالة نصية، وسنستخدم بشكلٍ أساسي الحزمتين البرمجيتين net و io.

```
improt java.io.*
import java.net.*
```

نعرّف في البداية البيانات والكيانات التي ستدخل معنا في صياغة البريد الإلكتروني محددين البوابة بالرقم 25 المخصصة لبروتوكول SMTP ومضيف افتراضي هو localhost ومجموعة متغيرات من النمط String إضافة إلى الكيانات من الأصناف BufferedReader و PrintWriter و Socket التي ستضمن تبادل البيانات.

بعدها يمكننا انطلاقاً من الرزمة البرمجية net إنشاء اتصال من خلال المقابس Socket عبر البوابة 25 المخصصة لبروتوكولات البريد الإلكتروني. ويمكننا بعد ذلك برمجة الاتصال مع الخادم من خلال المقبس والقارئات Readers والكاتبات Writers وانتظار الاستجابة من الخادم. وفي ذلك كله تلعب الطريقتان ()getInput و ()sendEmail الدور الرئيس في عمل البروتوكول.

المنهج ()sendEmail:

تعتمد هذه الطريقة على فتح الاتصال عبر المقبس وتبادل البيانات عبر أصناف الحزمة البرمجية io.

BufferReader, PrintWriter, OutputStreamWriter

ويظهر كذلك الأمر في جزء البرنامج أدناه كيفية تطبيق الأوامر HELO, MAILFROM, RCPT TO, DATA. بينما تحصل الطريقة ()getInput على معلومات الدخل من المستخدم من خلال كيانات من الأصناف ,BufferReader والطريقة ()readLine و مراقبة ورود نهاية الرسالة أوعدم صلاحيتها. فإذا حققت الرسالة شروط الإرسال يجري تخزينها في عازل Buffer وتهيئتها بالصياغة المناسبة.

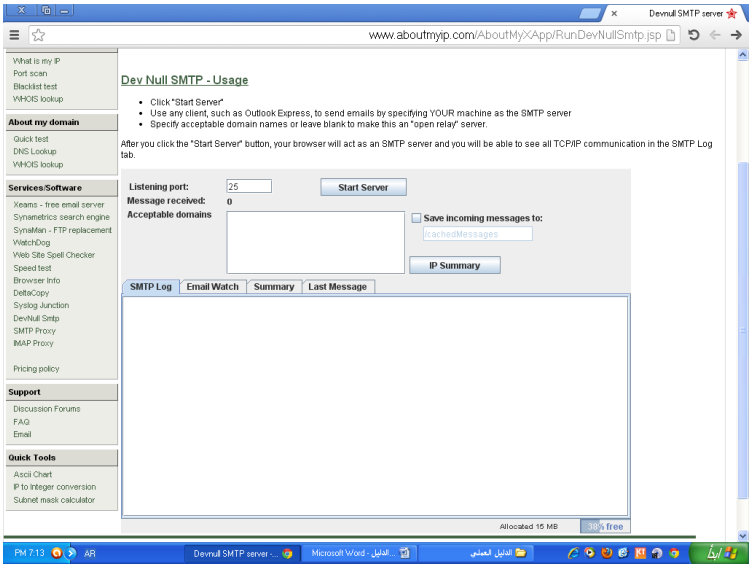
وبشكل عام يمكننا تلخيص ما سبق باختصار في أننا ننشئ ارتباط مباشر مع خادم بريد Mail Server يعتمد تبادل البيانات عبر مقابس TCP المفتوحة على البوابة 25 المخصصة لبروتوكول نقل البريد البسيط من خلال مجموعة من الأوامر التي تحدد جزئيات التفاصيل الرسائل الإلكترونية اعتماداً على برنامج وحيد المسلك ولن نتطرق إلى موضوع تعدد المسالك الذي تعتمده العديد من منظومات البريد

الإلكتروني الذي تعتمده العديد من المنظومات الأكثر تعقيداً. ستقوم هذه المنظومة بمجرد الاستفسار من المستخدم عن وجود بيانات يريد إرسالها ومن ثم ستقوم بعملية الإرسال.

في هذا الجزء، ستتعرف –عزيزي الدارس– على آلية عمل بروتوكول (SMTP) من خلال برنامج بسيط يظهر لك الأوامر التي يرسلها العميل (Client) إلى الخادم (Server). من أجل ذلك ستعمل على كتابة برنامج العميل (Client) المذكور أدناه والذي يعمل بنفس مبدأ أوتلوك إكسبرس (Outlook Express). وبالتالي يمكنك الاستعانة بأحد سيرفرات البريد الإلكتروني مثل (Microsoft Exchange Server) أو سيرفر بسيط يعمل على محاكاة السيرفرات الحقيقية. أحد هذه السيرفرات البسيطة هو برنامج (DevNull SMTP) الذي يمكنه أن يعمل من خلال المتصفح على شكل آبلت (Applet) أو كبرنامج تنفيذي على شكل (JAR File). هذا السيرفر سيعمل على محاكاة تشابه عمل السيرفر الحقيقي، ولكنه بدلا من إرسال البريد إلى المستلم، فإنه يخزن مضمون الرسالة بشكل مؤقت في ذاكرة ثم يتم حذف الرسالة. ولكنه يوضح كيف يتصرف السيرفر الحقيقي بحيث يعطيك عزيزي الدارس صورة واضحة عن أوامر (SMTP).

أولا: اعمل على تحميل برمجية DevNull SMTP من الموقع التالي ثم أنقر الزر (Start Server):

http://www.aboutmyip.com/AboutMyXApp/DevNullSmtp.jsp

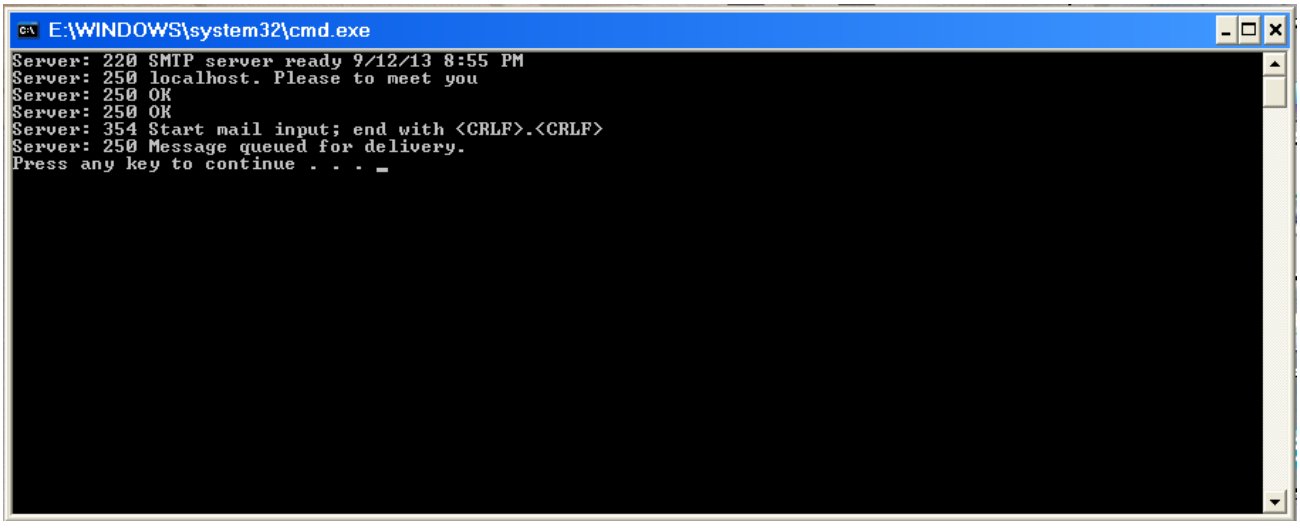


ثانيا: في برنامج Textbad، أكتب البرنامج التالي (smtpClient.java)، ثم صنفه (Compile).

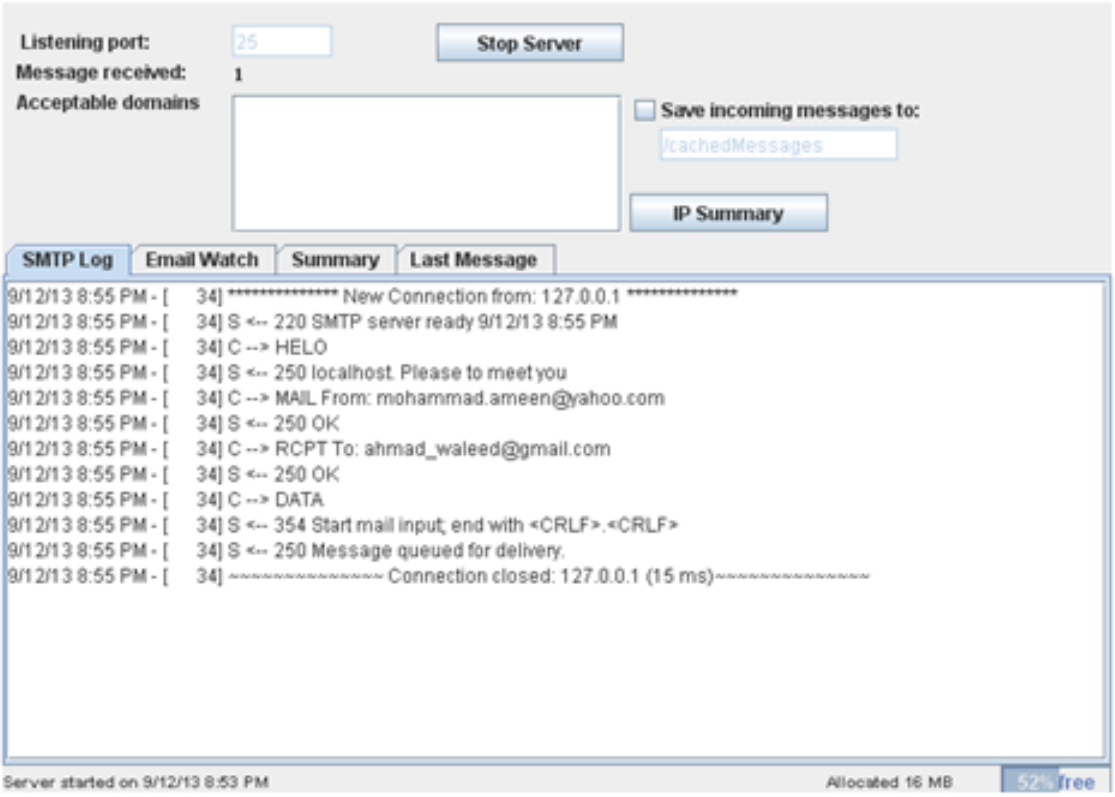
```
/*
Written by Dr. Samer Husni Jaloudi, Al Quds Open University, 12.09.2013
This is a simple SMTP client. It shows how to use the main commands of SMTP.
Download the SMTP server "DevNull SMTP" from: http://www.aboutmyip.com/AboutMyXApp/DevNullSmtp.jsp
The DevNull therefore works as an exchange Email server, however it does not forward the Emails,
but the Email gets deleted, and this is why it is Null.
*/
import java.io.*;
import java.net.*;
public class smtpClient
{
        public static void main(String[] args)
{
Socket smtpSocket = null;
DataOutputStream os = null;
BufferedReader br = null;
try
{
smtpSocket = new Socket("localhost", 25); // because DevNull is running in an applet in the Browser
os = new DataOutputStream(smtpSocket.getOutputStream());
br = new BufferedReader(new InputStreamReader(smtpSocket.getInputStream()));
```

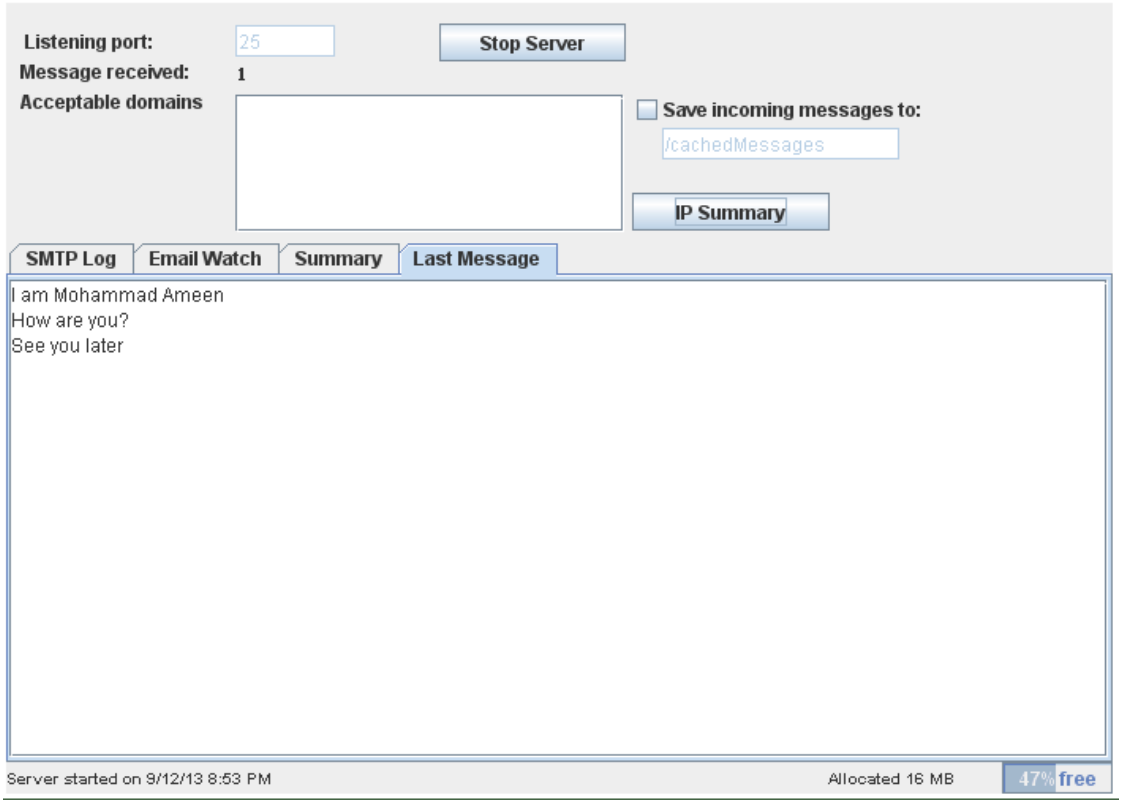
```
}
catch (UnknownHostException e){System.err.println("Host not Known");}
catch (IOException e){System.err.println("Couldn't get I/O for the connection to host");}
if (smtpSocket != null && os != null && br != null)
{
try
{//SMTP specification, RFC1822/3
                                        os.writeBytes("HELO\n");
                                        receiveMsg(br);
                                        os.writeBytes("MAIL From: mohammad.ameen@yahoo.com\n");
                                        receiveMsg(br);
                                        os.writeBytes("RCPT To: ahmad_waleed@gmail.com\n");
                                        receiveMsg(br);
                                        os.writeBytes("DATA\n");
                                        receiveMsg(br);
                                        os.writeBytes("I am Mohammad Ameen\nHow are you?\nSee you later\n.\n");//must end with \n.\n
                                        receiveMsg(br);
                                         receiveMsg(br);
                                        os.close();
                                        br.close();
                                        smtpSocket.close();
                                }
                                catch (UnknownHostException e)
                                {
                                        System.err.println("Trying to connect to unknown host: " + e);
                                }
                                catch (IOException e){System.err.println("IOException:  " + e);}
}
}//main
        static void receiveMsg(BufferedReader br)
        {
                String responseLine;
                try
                {
                        responseLine = br.readLine();
                        System.out.println("Server: " + responseLine);
                }
                catch (Exception e){System.out.println("Error");}
        }//receiveMsg
}//class
```

ثالثا: نفّذ (Run) البرنامج وتأكد من أن نتائج برنامج العميل (smtpClient) هي كما تظهر في الشكل التالي:



خامسا: تأكد من أن نتائج السيرفر في المتصفح هي كما تظهر في الشكلين التاليين:





سؤال (5): اشرح آلية عمل برنامج العميل (Client) السابق؟

سؤال (6): اشرح آلية ردّ الخادم (Dev Null SMTP Server)؟

سؤال (7): صمم جدولا مكون من عمودين، بحيث تضع الأوامر التي تخرج من العميل (smtpClient.java) في العمود الأول وفي العمود الثاني الأرقام التي تصدر من الخادم (Server) في إطار الرد على أوامر العميل (Client)؟

خطوات عمل الجزء الرابع

في هذا الجزء، ستتعرف على آلية عمل بروتوكول (HTTP) من خلال برنامج بسيط يظهر لك طريقة الربط مع موقع website معين من خلال الصنف المجردة URLConnection جهة المخدوم (Client) إلى خادم (Server) الموقع.

سؤال (8): أكتب برنامج المخدوم Client بلغة جافا تستخدم فيه الصنف المجردة URLConnection لتتصل بخادم (Server) الموقع "http://portal.qou.edu/portalLogin.do" وتقرأ محتوياته؟

التجربة السادسة

برمجة شبكات الوسائط المتعددة

Multimedia Network Programming

مقدمة

تعتبر جافا أهم التقنيات للوسائط المتعددة على الإنترنتِ، خصوصاً لتَوزيع المحتوى المتعدد الأوساطِ عند الطّلب. إحدى الطرق الشائعة لتزويدنا بمعلومات الوسائط المتعددة ووسائط الترفيه على الانترنت هي باستخدام جافا آبلت (java applets) حيث لعبت دوراً هاماً لشيوع استخدام لغة جافا علماً بأنه ليس من الضروري استخدام آبلت (applets) لنقل الوسائط المتعددة على الانترنت.

إن بيانات المراقبة عند بناء صفحات الويب وكذلك قيود الأمن الموضوعة في الآبلت (applets) تجعل نقل الملفات بواسطة تطبيقات java مفضلة عند استخدام جافا آبلت Applets. في البداية تم بناء Java لصيغة ملفات (sun) الصوتية فقط. أما في الوقت الحاضر فإنها تستخدم صيغ ملفات (wav files.) وصيغ ماكنتوش (aif files) وكذلك صيغ MIDI كل هذه الصيغ مدعومه من قبل Java. أول انطلاقة لنقل ملفات الصور كان لـ جافا لصيغ GIF فقط (gif.) وقد تم إضافة دعم صيغ JPEG (.jpg & .jpeg) في إصدار JDK1.1 للتمكن من تشغيل فيديو كليب (video clips) وغيرها من الصيغ.

توفر جافا الميزّات التالية للوسائط المتعددة:

- الدعم الشامل للرسم (primitives and bitmapped)
- الدعم الصوتي الرقمي الأساسي
- بيئة متعددة المسالك (Multithreaded)، التي تُساعدُ في توقيت الصور المتحركةِ
- متابع الوسائط وذلك لمتابعة محتوى الوسائط الموزعة.

ملاحظة: عليك عزيزي الدارس بتحميل java media framework (JMF) لتتمكن من التعامل مع ملفات الوسائط المتعددة.

الأهداف

يتوقع منك عزيزي الدارس بعد الانتهاء من هذه التجربة أن يكون قادرا على أن:

1 - تناقش الطرق التي يمكن من خلالها نقل الصور عبر الشبكة.
2 - تعرف الكيفية التي يتم بها التعامل مع الصوت عبر الشبكة.
3 - تستخدم الـ API للتعامل مع الصوت والصورة.
4 - تعد وتشغل ملفات الـ Video والـ Audio.

خطوات عمل الجزء الأول

أكتب برنامجين بلغة جافا بحيث يشكلان زوج خادم/عميل (Client/Server) بحيث يعملان على:

1. إرسال ملف صور من خادم إلى زبون يمكن الاتصال به؟
2. استقبال ملف صور مرسل من خادم إلى زبون؟

خطوات عمل الجزء الثاني

اكتب برنامجا بحيث يستقبل الخادم اتصالا من زبائن ويقوم الخادم بإرسال:

1. ملف صور باسم "*beesting.jpg*" إذا أرسلَ الزبونَ طلبَ image أو
2. ملف صوت باسم "*cucko.au*" إذا أرسلَ الزبونَ طلبَ sound.

كما يقوم الزبون بحفظ الملفات حسب شكلها؛ ملف صور باسم "*image.jpg*"، وملف صوت باسم "*sound.au*". يتم إرسال الملف (سواء ملفَ صورةَ أو صوت) باستخدام الطريقة *sendFile*، بينما يتم استقبال الملف باستخدام الطريقة *getFile*؟

خطوات عمل الجزء الثالث

عزيزي الدارس، لديك البرنامج التالي. اعمل على كتابته، وتصنيفه (Compile) ثم تنفيذه (Run)؟

```
import java.applet.*;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.net.*;

public class SimpleSound extends JFrame implements ActionListener
{
	private AudioClip clip1;
	private JButton play1, stop1, loop1;
	private JPanel buttonPanel;

	public static void main(String[] args)
	{
		SimpleSound frame = new SimpleSound();
		frame.setSize(300,200);
		frame.setVisible(true);
		frame.setDefaultCloseOperation(EXIT_ON_CLOSE);
	}

	public SimpleSound()
	{
		setTitle("Simple Sound Demo");
		try
		{
			//Obviously, the path given below is simply an example and could be anywhere in the user's file system.
			//clip1 = Applet.newAudioClip(new URL("file:///C:/sounds/hi.au"));
			clip1 = Applet.newAudioClip(new URL("file:hi.au"));
		}
		catch(MalformedURLException muEx)
		{
			System.out.println("*** Invalid URL! ***");
			System.exit(1);
		}

		play1 = new JButton("Play");
		play1.addActionListener(this);
		stop1 = new JButton("Stop");
		stop1.addActionListener(this);
		loop1 = new JButton("Loop");
		loop1.addActionListener(this);
		buttonPanel = new JPanel();
		buttonPanel.add(play1);
		buttonPanel.add(stop1);
		buttonPanel.add(loop1);
		add(buttonPanel,BorderLayout.SOUTH);
	}

	public void actionPerformed(ActionEvent event)
	{
		if (event.getSource() == play1)
		clip1.play();
		if (event.getSource() == stop1)
		clip1.stop();
		if (event.getSource() == loop1)
		clip1.loop();
	}
}
```

سؤال (1) : اشرح آلية عمل البرنامج، ثم حلل النتائج؟

خطوات عمل الجزء الرابع

لإنجاز هذا الجزء، عليك عزيزي الدارس أن تعمل على تحميل إطار وسائط جافا (Java Media Framework – JMF) من موقع شركة أوراكل. في هذا الجزء ستكتب آبلت (Applet) تعمل من خلال المتصفح من خلال الخطوات التالية:

أولا: أنقل البرنامج التالي إلى واجهة TextPad ثم اعمل على تصنيفه (Compile Java) ثم تنفيذه (Run Java Applet).

```
import java.applet.*;
import java.awt.*;
import java.net.*;

import javax.media.*; // from JMF

public class PlayerApplet extends Applet
{
	Player player1 = null;
	public void init()
	{
		setLayout( new BorderLayout() );
		String mediaFile = getParameter( "FILE" );
		try
		{
			URL mediaURL = new URL( getDocumentBase(), mediaFile );
			player1 = Manager.createRealizedPlayer( mediaURL );
			if (player1.getVisualComponent() != null)
				add("Center", player1.getVisualComponent());
			if (player1.getControlPanelComponent() != null)
				add("South", player1.getControlPanelComponent());
		}
		catch (Exception e)
		{
			System.err.println( "Got exception " + e );
		}
	}

	public void start()
	{
		player1.start();
	}

	public void stop()
	{
		player1.stop();
		player1.deallocate();
	}

	public void destroy()
	{
		player1.close();
	}
}
```

ثانيا: أكتب ما يلي في ملف نصّي وخزنه باسم (htmlPage.html) :

```
<Applet code=PlayerApplet width=320 height=300>
	<PARAM name=FILE value="DELTA.MPG">
</Applet>
```

ثالثا: يمكنك عزيزي الدارس تحميل (download) ملف فيديو تجريبي من الموقع التالي:

http://www.fileformat.info/format/mpeg/sample/

وليكن الملف الأول في الصفحة (DELTA.MPG)

رابعا: تأكد من أن الملفات التالية جميعها على نفس المجلد: (PlayerApplet.class, htmlPage.html, DELTA.MPG)، ثم افتح الملف (htmlPage.html) بالنقر نقرتين متتاليتين على الملف نفسه. في هذه الحالة، يجب أن يتم تشغيل الآبلت (PlayerApplet.class) في صفحة (htmlPage.html) كما يلي:

ملاحظة: في حال لم يتم تشغيل الملف في المتصفح، يمكن تشغيله من خلال مشغل الآبلت appletviewer. إذا كنت تستخدم برمجية TextPad، فيمكن تصنيف الملف أولا بالضغط على مفتاح التحكم CTRL بالتزامن مع مفتاح 1. ثم لتشغيل الأبلت بالضغط على مفتاح التحكم CTRL بالتزامن مع مفتاح 3.

سؤال (2): اشرح آلية عمل البرنامج السابق (PlayerApplet.class) ؟