الفصل الثاني

التطبيق الأول باستخدام J2ME، خطوات بناء التطبيق

عرفنا فيما سبق أن J2ME هي القسم الثالث من أقسام الجافا بعد J2SE و J2EE.

وهذا القسم J2ME يعمل على الأجهزة الصغيرة device Small، والآن سنتعرف على المكتبات الخاصة بهذه اللغة وكذلك أنواع المعطيات المستخدمة وهيكلية بناء البرنامج، ثم ننتقل إلى بناء التطبيق الأول باستخدام J2ME.

❖ المكتبات الأساسية الخاصة بالـ J2ME هي:

javax.microedition.lcdui: تزودنا هذه المكتبة بالعديد من واجهات المستخدم مثل النوافذ والأزرار وصناديق النصوص.

javax.microedition.rms: تزودنا هذه المكتبة بالطرق المستخدمة لخزن البيانات على وحدة الخزن الرئيسية للجوال.

javax.microedition.midlet: تزودنا هذه المكتبة بال MIDP وكذلك بالبيئة التي سيعمل فيها التطبيق.

javax.microedition.io: تزودنا هذه المكتبة بالمناهج الخاصة بعمليات الاتصال GenericConnection framework GCF java.io تزودنا هذه المكتبة بالمناهج الخاصة بعمليات الإدخال والإخراج.

java.lang: تحتوي على المناهج الخاصة بال J2SE وأنواع المعطيات.

java.util: تحتوي على المناهج الملحقة.

كما إن هناك العديد من المكتبات الأخرى التي تعمل على أجهزة معينة فقط بينما هذه المكتبات هي المكتبات الأساسية الخاصة بال J2ME.

❖ **أنواع المعطيات في J2ME:**

CODE

```
java.lang.Boolean
java.lang.Byte
java.lang.Character
java.lang.Integer
java.lang.Long
java.lang.Short
```

ونلاحظ هنا أن J2ME لا تدعم الأعداد ذوات النقطة العائمة float , double وذلك لأن النقطة العائمة تحتاج إلى عتاد كبير وبرمجيات ذات تكلفة عالية

❖ **بناء البرنامج الأول:**

عند قراءة هذا الموضوع يفترض أن يكون عندك معلومات لا بأس بها عن البرمجة الكائنية.

في برنامج الجافا أو السي مثلا نرى أن بداية تنفيذ الكود يبدأ من الدالة main أما بالنسبة لل J2ME فإن الأمر مختلف قليلاً وسيبدأ تنفيذ البرنامج في J2ME من الدالة startApp وذلك لسبب بسيط وهو:

عندما نقوم بتحديد برنامج ما هو البرنامج الرئيسي في الجافا أو السي يكفي أن نقوم بكتابة الدالة Main لنشير للمترجم أن يبدأ من هنا، أما في J2ME فإننا نقوم بإنشاء منهج مشتق من المنهج MIDlet الموجود في الحزمة javax.microedition.midlet ، ويوجد في هذا المنهج الطرق المجردة abstract Methods الثلاث التالية , destroyApp , pauseApp startApp .

ومن أساسيات البرمجة الكائنية أن الطرق المجردة في صنف مشتق منه يجب ذكرها كاملة، ولهذا فإن البرنامج الرئيسي سيكون هو البرنامج الذي سيتم

اشـتقاقه مـن الـصنف MIDlet وسـيبدأ تنفيـذ البرنـامج مـن الدالـة startApp أيضاً هناك الطريقتان pauseApp , destroyApp.

pauseApp: يتم تنفيذ هذه الدالة عند انتقال التطبيق من الحالة النشطة إلى الحالة الغير نشطة.

destroyApp: يتم تنفيذ هذه الدالة عند إغلاق التطبيق.

❖ **دورة حياة البرنامج الرئيسي MIDlet**

يمر التطبيق بثلاثة مراحل أو حالات:

الحالة الابتدائية النشطة ثم الحالة غير النشطة (التوقف المؤقت) ثم حالة الإغلاق وإنهاء التطبيق.

لا تهتم حالياً بالحالة pauseApp ولكن سنركز على الحالتين , startApp destroyApp.

يجب ملاحظة إنه عند تنفيذ أي تطبيق فإنه سيقوم أولاً بتنفيذ الكود الموجود في المشيد Constructor والذي يحمل نفس اسم المنهج، وبعد ذلك سينتقل التنفيذ إلى الدالة startApp.

❖ والآن سنقوم بكتابة المثال التالي:

قم بكتابة هذا الكود مع ملاحظة إنه يجب حفظ هذا الملف بنفس اسم المنهج HelloMidlet

CODE

```
import javax.microedition.midlet.*;

import javax.microedition.lcdui.*;


public class HelloMidlet extends MIDlet
{
```

```
private Display display;


TextBox box = null;

public void HelloMidlet(){}

public void startApp(){


display =Display.getDisplay(this);

box =new TextBox("Simple Example ","Hello
",20,0);

display.setCurrent(box);
}

public void pauseApp(){ }

public void destroyApp(boolean unconditional){}
}
```

❖ لتحويل هذا البرنامج إلى تطبيق على الجوال سنتبع الخطوات التالية:

خطوات تحويل البرنامج المصدر (java) إلى (jar, jad) لمن يستخدمون j2me)

١ - يقوم هذا الآمر بترجمة الملف المصدر java إلى class والتحقق من الأخطاء القواعدية للغة الجافا بنية الآمر javac.

CODE

```
C:\> %jdk1.3.0_02\bin\%javac -g:none -d
tmpclasses -bootclasspath $midpapi -classpath
$J2MECLASSPATH MyProgram.java
```

العبارات

CODE

```
javac
-g
-d
-bootclasspath
-classpath
```

هي عبارات ثابتة لا تتغير حسب تغير البرنامج، أما بقية العبارات فهي عبارات متغيرة حسب البرنامج والمجلد أو الملفات المتضمنة مكونات الأمر javac.

٪jdk1.3.0_02\bin\% : الدليل الحالي للملف javac.exe غالباً يكون اسمه jdk1.3.0_02\bin\ .

ويمكنك الاستغناء عن كتابته إذا قمت بكتابة الآمر من داخل الدليل للملف javac.exe .

none: تعني عدم إظهار التنقيح.

وهذا الخيار غير مهم ويمكن الاستغناء عنه وعدم كتابته.

tmpclasses : المجلد الذي ستوضع فيه الملفات class بعد ترجمتها.

أيضاً هذا الخيار غير مهم ويمكن الاستغناء عنه وسيتم وضع الملفات .class في المجلد الحالي ٪jdk1.3.0_02\bin٪ للملف javac.exe .

$midpapi : هذا هو package أو الكلاسات الأساسية لل J2ME وتكون مضغوطة على هيئة ملف .zip وغالباً ما تكون في الدليل C:\WTK104\lib\midpapi.zip .

$J2MECLASSPATH : الكلاسات الإضافية التي قمت بتضمينها في برنامجك.

مثلاً لو كان لديك أكثر من كلاس في برنامج واحد يجب عليك تحزيم الكلاسات الإضافية في حزمة واحدة، وبعد ذلك ستقوم بكتابة اسم هذه الحزمة مع المسار بدلاً عن الكلمة $J2MECLASSPATH وغالباً يتم تضمين هذه الكلاسات إلى الحزمة C:\WTK104\wtklib\kvem.jar .

مثال على ذلك لنفترض أنه لدينا البرنامج HelloMidlet والموجود في المجلد C:\java ونريد ترجمته بالعبارة javac وإرسال الملفات الناتجة .class إلى الدليل C:\WTK104\bin\Classes\ .

سنقوم بكتابة الآمر :

CODE

```
C:\jdk1.3.0_02\bin\javac.exe -g:none -d
C:\WTK104\bin\Classes\ -bootclasspath
C:\WTK104\lib\midpapi.zip -classpath
C:\WTK104\wtklib\kvem.jar
C:\Java\HelloMidlet.java
```

يجب الانتباه إلى كتابة الأوامر بالحروف الصغيرة أو الكبيرة تبعاً لأسماء الملفات مثلاً لو قمنا بكتابة -C:\WTK104\lib\MIDPAPI.zip bootclasspath فإنه ستنتج لدينا عبارة خطأ لأن الملف MIDPAPI.zip مكتوب بحروف كبيرة بينما هو في الأصل مكتوب بحروف صغيرة.

قد تدور بذهنك هذه الأسئلة:

ماذا لو كان لدينا أكثر من ملف ونريد ترجمة كل هذه الملفات؟

كما ذكرنا سابقاً عندما شرحنا J2MECLASSPATH$ نقوم أولاً بترجمة الكلاس المستدعى الإضافي -وليس الذي يستدعي- ونمر بالمراحل الثلاث إلى أن نصل إلى الشكل .jar وبعد ذلك نقوم بوضع اسمه مع الدليل بدلاً عن المتغير J2MECLASSPATH$.

والآن سيدور بذهنك سؤال آخر ماذا لو كان كل كلاس يقوم باستدعاء الكلاس الآخر؟

أي أن الكلاسات تستدعي كلاسات أخرى وهذه الكلاسات الأخرى تستدعي نفس الكلاسات الأصلية، وحل هذه المسألة هو أيضاً حل آخر للمسألة السابقة -أي حتى لو كانت الكلاسات لا تستدعي بعضها البعض- وهو حل في غاية البساطة فقط سوف تقوم بذكر أسماء الملفات .java كلها في عبارة واحدة.

مثال لو كان لدينا الملفات test1.java و test2.java وكان هذان الكلاسان يستدعيان بعضهم البعض أو أحدهم يستدعي الآخر فإنه يمكن كتابة الأمر javac على الشكل:

CODE

```
C:\jdk1.3.0_02\bin\javac.exe -g:none -d
C:\WTK104\bin\Classes\ -bootclasspath
C:\WTK104\lib\midpapi.zip -classpath
C:\WTK104\wtklib\kvem.jar test1.java test2.java
```

٢ - المرحلة التالية مرحلة التحقق من byte code باستخدام الأمر perversity ويتم في هذه المرحلة التحقق من وجود الكلاسات الأخرى في البرنامج وسوف يتم إضافة مجموعة من البيانات على الكلاس .class ويمكن معرفة ذلك من خلال مراقبة حجم الكلاس قبل تنفيذ هذه المرحلة وبعد تنفيذ هذه المرحلة، وبنية الآمر كالتالي:

CODE

```
C:\>%WTK104\bin\% perversity -class path
$midpapi;tmpclasses -d classes tmpclasses
```

العبارات

CODE

```
perversity
-class path
;
-d
```

هي عبارات ثابتة لا تتغير حسب تغير البرنامج، أما بقية العبارات فهي عبارات متغيرة حسب البرنامج والمجلد أو الملفات المتضمنة.

%WTK104\bin\: الدليل الحالي للملف preverify.exe غالباً يكون اسمه WTK104\bin\ ويمكنك الاستغناء عن كتابته إذا قمت بكتابة الأمر من داخل المجلد.

$midpapi: هو نفسه الملف المشروح في الخطوة السابقة عبارة عن جميع الكلاسات الخاصة باللغة J2ME.

tmpclasses : هو نفسه المجلد المشروح في الفقرة السابقة الدليل التي تتواجد فيه الكلاسات .class بعد تحويلها من الشفرة المصدرية إلى byte code وسيتم التحقق من كل الكلاسات الموجودة في هذا الدليل ولن نحدد كلاس واحد .class.

classes: هذا الدليل أو المجلد هو المجلد الذي سيوضع فيه الملفات .class بعد التحقق منها.

مثال على ذلك: لنفرض أن الكلاس الذي قمنا بترجمته في الخطوة السابقة قد قمنا بوضعه في الدليل C:\WTK104\bin\Classes\ .

والآن نريد استخدام الأمر perversity ووضع الكلاسات الناتجة من هذا الأمر في المجلد C:\jdk1.3.0_02\bin\ فإننا سنقوم بكتابة الأمر التالي:

CODE

```
C:\WTK104\bin\preverify.exe -class path
C:\WTK104\lib\midpapi.zip;C:\WTK104\bin\Classes\
-d C:\jdk1.3.0_02\bin\ C:\WTK104\bin\Classes\
```

ويجب أن تلاحظ هنا أن هذا الأمر يقوم بالتنفيذ على عدد من الكلاسات موجودة في مجلد واحد وليس على كلاس واحد بعينه مثلاً لو كان لدينا الكلاسات test1.class , test2.class الموجودين في الدليل C:\WTK104\bin\Classes\ .

سنقوم بكتابة نفس الأمر السابق لجميع الكلاسات test2.class test1.class .

CODE

```
C:\WTK104\bin\preverify.exe -class path
C:\WTK104\lib\midpapi.zip;C:\WTK104\bin\Classes\
-d C:\jdk1.3.0_02\bin\ C:\WTK104\bin\Classes\
```

٣ - تحزيم الملفات باستخدام الأمر jar:

يقوم هذا الأمر بضغط الكلاسات وتجميعها في حزمة واحدة يتم تنفيذها على الموبايل، وبنية هذا الأمر كالتالي:

CODE

```
%C:\jdk1.3.0_02\bin%> jar cvf MyProgram.jar
MyProgram.class
```

أو على الطريقة التالية:

CODE

```
%C:\jdk1.3.0_02\bin%> jar cmf MANIFEST.MF
MyProgram.jar MyProgram.class
```

والفرق بين الطريقتين هي أن في الطريقة الأولى لا نقوم بتضمين الملف MANIFEST.MF وهذا الملف مهم جداً على بعض الأجهزة مثل نوكيا وليس مهما على أجهزة أخرى مثل سيمنس بينما في الطريقة الثانية سنقوم بتضمين الملف MANIFEST.MF الذي سنقوم بشرح كيفية تكوينه فيما بعد.

ويجب الانتباه هنا إلى أننا قمنا بالدخول إلى الدليل أولاً قبل استخدام الأمر jar على العكس من المراحل السابقة التي لا تشترط ذلك. والدخول إلى الدليل مهم جداً من أجل أن يتم تحزيم الكلاسات .class في الدليل الرئيسي للملف jar .

أما في حالة عدم الدخول للدليل المحتوى على الأمر jar فإن تحزيم الكلاسات في الملف jar سيكون داخل الملف الفرعي الموجود فيه الكلاسات التي تم تحزيمها.

لا تهتم إذا لم تفهم ما ذكر سابقاً فقط قم بالدخول إلى الدليل المحتوي على الأمر jar واستخدم الأمر من داخل المجلد.

C:\jdk1.3.0_02\bin%: الدليل الحالي للملف jar.exe .

cvf أو cmf أو umf: يعبر عن الطريقة المستخدمة في تنفيذ الأمر، وسنقوم بشرحها لاحقاً.

MyProgram.jar: اسم البرنامج .jar الذي تريد تحزيمه.

MyProgram.class: اسم الكلاس أو الكلاسات التي تريد تحزيمها.

مثال سنقوم بتحزيم المثال السابق بالشكل:

CODE

```
C:\jdk1.3.0_02\bin\jar.exe cvf Hello.jar
 HelloMidlet.class
```

سيتكون لدينا البرنامج Hello.jar الناتج من تطبيق الأمر السابق.
والآن سيتبادر للذهن ماذا لو كان لدينا أكثر من كلاس واحد ونريد تحزيمها في حزمة واحدة؟
مثلاً الكلاسات test1.class test2.class ونريد تحزيمها إلى test.jar سنقوم باستخدام الأمر التالي:

CODE

```
C:\jdk1.3.0_02\bin\jar.exe cvf test.jar test1.class test2.class
```

يجب أن نذكر هنا أن استخدام البارامتر umf يستخدم لإضافة الملحقات المستخدمة في البرنامج مثل الصور أو أيقونات التطبيق.
مثلاً لو أردنا إضافة الصورة test.png إلى البرنامج Hello.jar طبعاً هذا إذا كنا قد استخدمنا هذه الصورة داخل الكلاس HelloMidlet.class في شفرة البرنامج، سوف نستخدم الأمر التالي:

CODE

```
C:\jdk1.3.0_02\bin\jar.exe umf MANIFEST.MF Hello.jar Test.png
```

٤ - يتم إنشاء وتكوين ملفات jad و MANIFEST.MF باستخدام أي محرر للنصوص، وتتم كتابة عدة حقول فيهما منها ما هو ضروري ومنها ما هو ليس ضروري.
وهذا الجدول المرفق يشرح هذه الحقول:
يمكنك فتح أي ملفات jad لبرامج أخرى لمعرفة كيفية كتابتها

JAR manifest attributes		
اسم الخاصية	مطلوب	شرح الخاصية
MicroEdition-Configuration	نعم	رقم النسخة من J2ME Configuration مثلاً CLDC-1.0
MicroEdition-Profile	نعم	رقم النسخة من J2ME Profile مثلاً MIDP-1.0
MIDlet-n	نعم	اسم التطبيق, اسم الأيقونة , اسم الكلاس الرئيسي والرقم n- يعبر عن رقم التطبيق تبدأ بـ ١
Data-Size-MIDlet	لا	العدد الأقصى من البايتات الذي سيستخدمه التطبيق في التعامل مع الملفات
MIDlet-Description	لا	يحتوي على وصف للتطبيق
Icon-MIDlet	لا	المسار مع الملف لأيقونة التطبيق
MIDlet-Info-URL	نعم	عنوان الإنترنت الذي يحتوي على وصف للتطبيق
MIDlet-Name	نعم	اسم التطبيق
MIDlet-Vendor	نعم	معلومات منتج البرنامج
MIDlet-Version	نعم	رقم نسخة التطبيق

JAD attributes		
شرح الخاصية	مطلوب	اسم الخاصية
اسم التطبيق	نعم	MIDlet-Name
رقم نسخة التطبيق	نعم	MIDlet-Version
معلومات منتج البرنامج	نعم	MIDlet-Vendor
اسم الكلاسات المحزمة Jar.	نعم	MIDlet-Jar-URL
حجم الملف - jar. - بالبايتات	نعم	MIDlet-Jar-Size
يحتوي على وصف للتطبيق	لا	MIDlet-Description
المسار مع الملف لأيقونة التطبيق	لا	Icon-MIDlet
عنوان الإنترنت الذي يحتوي على وصف للتطبيق	نعم	MIDlet-Info-URL
العدد الأقصى من البايتات الذي سيستخدمه التطبيق في التعامل مع الملفات	لا	Data-Size-MIDlet

وهنا يجب ملاحظة أن البيانات الموجودة في الملف MANIFEST.MF والبيانات الموجودة في الملف jad يجب أن تتطابق وإلا فإن التطبيق لن يعمل في بعض الأجهزة مثل أجهزة نوكيا.

وسوف تظهر للمستخدم الرسالة التالية : صيغة الملف غير مدعومة.

والآن أصبح لدينا برنامج جاهز للعمل على أي موبايل.

لتجربة هذا التطبيق قبل تنزيله على الموبايل:

قم أولاً بنقل التطبيق إلى الدليل apps الموجود في الدليل الخاص بال J2ME وليكن C:\WTK104 أي C:\WTK104\apps.
بعد ذلك قم بفتح الملف jad باستخدام emulatorw.exe الموجود في الدليل الخاص بال J2ME وليكن C:\WTK104 أي C:\WTK104\bin .
تعتبر هذه المراحل في ترجمة البرنامج مملة وكبيرة ومعقدة ولكنها مهمة في بداية تعلم اللغة لمعرفة المراحل التي يمر بها البرنامج.

للسهولة يمكنك عمل قوالب جاهزة بملفات batche file.bat ومن ثم تغييرها عند تغيير اسم البرنامج