من اعداد مصعب غيلان- جمهورية اليمن العربية

-4خطوات تحويل البرنامج المصدر (java) الى(jad ,jar),
لمن يستخدمون *j2me toolkits*
وقد اعتمدت في هذا الشرح على كتاب Learning Wireless for Qusay Moh

هذا الشرح بالنسبة للاشخاص الذين لا يستخدمون JBuilder MobileSet
من اجل تحويل البرنامج او الشفرة المصدرية الى برنامج تنفيذي يعمل على الموبايل يجب المرور بهذة المراحل
1- تحويل البرنامج من الشفرة المصدرية الى java byte code باستخدام الامر javac
2 - التحقق من byte code باستخدام الامر preverify
3 - تحزيم الملفات .class الى الشكل .jar
4 - تكوين ملف .jad الذي يعمل مع الملف .jar
كل هذة الاوامر ستعمل عل محث dos

-1 يقوم هذا الامر بترجمة الملف المصدر .java الى .class
والتحقق من الاخطاء القواعدية للغة الجافا
بنية الامر javac

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

العبارات

javac
-g
-d
-bootclasspath
-classpath

هي عبارات ثابته لا تتغير حسب تغير البرنامج
اما بقية العبارات فهي عبارات متغيرة حسب البرنامج والمجلد او الملفات المتضمنه

مكونات الامر javac

%\%jdk1.3.0_02\bin : الدليل الحالي للملف javac.exe غالبا يكون اسمه \jdk1.3.0_02\bin
ويمكنك الاستغناء عن كتابته اذا قمت بكتابة الامر من داخل الدليل للملف javac.exe

none: : تعني عدم اظهار التنقيح
وهذا الخيار غير مهم ويمكن الاستغناء عنه وعدم كتابته

tmpclasses : المجلد الذي ستوضع فيه الملفات .class بعد ترجمتها
ايظا هذا الخيار غير مهم ويمكن الاستغناء عنه وسيتم وضع الملفات .class في المجلد الحالي %
javac.exe للملف %\jdk1.3.0_02\bin

$midpapi : هذا هو package او الكلاسات الاساسية لل J2ME
وتكون مضغوطة على هيئة ملف .zip وغالبا ماتكون في الدليل C:\WTK104\lib\midpapi.zip

$J2MECLASSPATH : الكلاسات الاضافية التي قمت بتضمينها في برنامجك
مثلا لو كان لديك اكثر من كلاس في برنامج واحد يجب عليك تحزيم الكلاسات الاضافية في حزمة واحدة
وبعد ذلك ستقوم بكتابة اسم هذة الحزمة مع المسار بدلا عن الكلمة $J2MECLASSPATH
وغالبا يتم تضمين هذة الكلاسات الى الحزمة C:\WTK104\wtklib\kvem.jar

مثال على ذلك لنفترض انه لدينا البرنامج HelloMidlet والموجود في المجلد C:\java ونريد ترجمته بالعبارة javac
وارسال الملفات الناتجة .class الى الدليل C:\WTK104\bin\Classes\
سنقوم بكتابة الامر

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 الى كتابة الاوامر بالحروف الصغيرة او الكبيرة تبعا لاسماء الملفات
فإنه ستنتج لدينا عبارة خطأ لأن الملف -bootclasspath C:\WTK104\lib\MIDPAPI.zip مثلا لو قمنا بكتابة مكتوب بحروف كبيرة بينما هو في الاصل مكتوب بحروف صغيرة MIDPAPI.zip

قد تدور بذهنك هذة الاسئلة
ماذا لو كان لدينا اكثر من ملف ونريد ترجمة كل هذة الملفات

$J2MECLASSPATH كما ذكرنا سابقا عندما شرحنا
نقوم اولا بترجمة الكلاس المستدعى الاضافي -وليس الذي يستدعي- jar. ونمر بالمراحل الثلاث الى ان نصل الى الشكل
$J2MECLASSPATH وبعد ذلك نقوم بوضع اسمه مع الدليل بدلا عن المتغير

والان سيدور بذهنك سؤال اخر ماذا لو كان كل كلاس يقوم باستدعاء الكلاس الاخر
اي ان الكلاسات تستدعي كلاسات اخرى وهذة الكلاسات الاخرى تستدعي نفس الكلاسات الاصلية

وحل هذة المسأله هي ايظا حل اخر للمسألة السابقة -اي حتى لو كان الكلاسات لا يستدعون بعضهم البعض- كلهم في عبارة واحدة java. وهو حل في غاية البساطة فقط سوف تقوم بذكر اسماء الملفات

وكان هذان الكلاسان يستدعيان بعضهم البعض او احدهم test2.java و test1.java مثال لو كان لدينا الملفات على الشكل javac يستدعي الاخر فإنه يمكن كتابة الامر

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

-2 المرحلة التالية مرحلة التحقق من byte code باستخدام الامر preverify

وتتم في هذه المرحلة التحقق من وجود الكلاسات الاخرى في البرنامج وسوف يتم اضافة مجموعة من البيانات على الكلاس .class
ويمكن معرفة ذلك من خلال مراقبة حجم الكلاس قبل تنفيذ هذة المرحلة وبعد تنفيذ هذة المرحلة
وبنية الامر كالتالي

C:\>%WTK104\bin\% preverify -classpath $midpapi;tmpclasses -d classes tmpclasses

العبارات

preverify
-classpath

;
-d

هي عبارات ثابته لا تتغير حسب تغير البرنامج
اما بقية العبارات فهي عبارات متغيرة حسب البرنامج والمجلد او الملفات المتضمنه
%\WTK104\bin% : الدليل الحالي للملف preverify.exe غالبا يكون اسمه \WTK104\bin
ويمكنك الاستغناء عن كتابته اذا قمت بكتابة الامر من داخل المجلد

$midpapi : هو نفسه الملف المشروح في الخظوة 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 الى byte code
وسيتم التحقق من كل الكلاسات الموجودة في هذا الدليل ولن نحدد كلاس واحد .class

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

مثال على ذلك : لنفرض ان الكلاس الذي قمنا بترجمته في الخطوة السابقة وقد قمنا بوضعه في الدليل
\C:\WTK104\bin\Classes
والان نريد استخدام الامر preverify
ووضع الكلاسات الناتجة من هذا الامر في المجلد \C:\jdk1.3.0_02\bin

فإننا سنقوم بكتابة الامر التالي

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

ويجب ان تلاحظ هنا ان هذا الامر يقوم بالتنفيذ على عدد من الكلاسات موجودة في مجلد واحد وليس على كلاس واحد بعينه
مثلا لو كان لدينا الكلاسات test1.class test2.class الموجودين في الدليل \C:\WTK104\bin\Classes
سنقوم بكتابة نفس الامر السابق لجميع الكلاسات test1.class test2.class

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

3 - تحزيم الملفات باستخدام الامر jar
يقوم هذا الامر بضغط الكلاسات وتجميعها في حزمة واحدة يتم تنفيذها على الموبايل
وبنية هذا الامر كالتالي

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

او على الطريقة التالية

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

والفرق بين الطريقتين هي انه في الطريقة الاولى لا نقوم بتضمين الملف MANIFEST.MF وهذا الملف مهم جدا على بعض الاجهزة مثل نوكيا وليس مهما على اجهزة اخرى مثل سيمنس
بينما في الطريقة الثانية سنقوم بتضمين الملف MANIFEST.MF الذي سنقوم بشرح كيفية تكوينه فيما بعد

ويجب الانتباه هنا الى اننا قمنا بالدخول الى الدليل اولا قبل استخدام الامر jar على العكس من المراحل السابقة التي لا تشترط ذلك
والدخول الى الدليل مهم جدا من اجل ان يتم تحزيم الكلاسات .class في الدليل الرئيسي للملف .jar
اما في حالة عدم الدخول للدليل المحتوى على الامر jar فإن تحزيم الكلاسات في الملف .jar سيكون داخل الملف الفرعي الموجود فيها الكلاسات التي تم تحزيمها
لا تهتم اذا لم تفهم ما ذكر سابقا فقط قم بالدخول الى الدليل المحتوى على الامر jar واستخدم الامر من داخل المجلد

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

cvf او cmf او umf : يعبر عن الطريقة المستخدمة في تنفيذ الامر
وسنقوم بشرحها لاحقا

MyProgram.jar : اسم البرنامج .jar الذي تريد تحزيمه

MyProgram.class : اسم الكلاس او الكلاسات التي تريد تحزيمها

مثال سنقوم بتحزيم المثال السابق بالشكل

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

الناتج من تطبيق الامر السابق Hello.jar سيتكون لدينا البرنامج

والان سيتبادر للذهن ماذا لو كان لدينا اكثر من كلاس واحد ونريد تحزيمها في حزمة واحدة
test.jar ونريد تحزيمها الى test1.class test2.class مثلا الكلاسات
سنقوم بإستخدام الامر التالي

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

يجب ان نذكر هنا ان استخدام البارامتر umf يستخدم لاضافة الملحقات المستخدمة في البرنامج مثل الصور او ايقونات التطبيق
مثلا لو اردنا اضافة الصورة test.png الى البرنامج Hello.jar
طبعا هذا اذا كنا قد استخدمنا هذة الصورة داخل الكلاس HelloMidlet.class في شفرة البرنامج
سوف نستخدم الامر التالي

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

4- يتم انشاء وتكوين ملفات jad و MANIFEST.MF باستخدام اي محرر للنصوص
ويتم كتابة عدة حقول فيهما منها ما هو ضروري ومنها ما هو ليس ضروري

وهذا الجدول المرفق يشرح هذة الحقول

JAR manifest attributes		
اسم الخاصية	مطلوب	شرح الخاصية
MicroEdition-Configuration	نعم	رقم النسخة من ال **J2ME** مثلا **Configuration CLDC-1.0**
MicroEdition-Profile	نعم	رقم النسخة من ال **J2ME** مثلا **Profile MIDP-1.0**
MIDlet-n	نعم	اسم التطبيق, اسم الأيقونة, اسم الكلاس الرئيسي والرقم **n-** يعبر عن رقم التطبيق تبدأ بـ **1**
MIDlet-Data-Size	لا	العدد الأقصى من البايتات الذي سيستخدمه التطبيق في التعامل مع الملفات
MIDlet-Description	لا	يحتوي على وصف للتطبيق
MIDlet-Icon	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		
اسم الخاصية	مطلوب	شرح الخاصية
MIDlet-Name	نعم	اسم التطبيق
MIDlet-Version	نعم	رقم نسخة التطبيق
MIDlet-Vendor	نعم	معلومات منتج البرنامج
MIDlet-Jar-URL	نعم	اسم الكلاسات المحزمة **Jar.**
MIDlet-Jar-Size	نعم	حجم الملف **- jar.** بالبايتات
MIDlet-Description	لا	يحتوي على وصف للتطبيق
MIDlet-Icon	لا	المسار مع الملف لأيقونة التطبيق
MIDlet-Info-URL	نعم	عنوان الإنترنت الذي يحتوي على وصف للتطبيق
MIDlet-Data-Size	لا	العدد الأقصى من البايتات الذي سيستخدمه التطبيق في التعامل مع الملفات

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

وسوف تظهر للمستخدم الرسالة التالية **:** صيغة الملف غير مدعومة

والان اصبح لدينا برنامج جاهز للعمل على اي موبايل

لتجربة هذا التطبيق قبل تنزيله على الموبايل

اولا قم بنقل التطبيق الى الدليل **apps** الموجود في الدليل الخاص بال **J2ME** وليكن **C:\WTK104** اي **C:\WTK104\apps**

بعد ذلك قم بفتح الملف **jad** باستخدام **emulatorw.exe** الموجود في الدليل الخاص بال **J2ME** وليكن **C:\WTK104** اي **C:\WTK104\bin**

تعتبر هذة 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