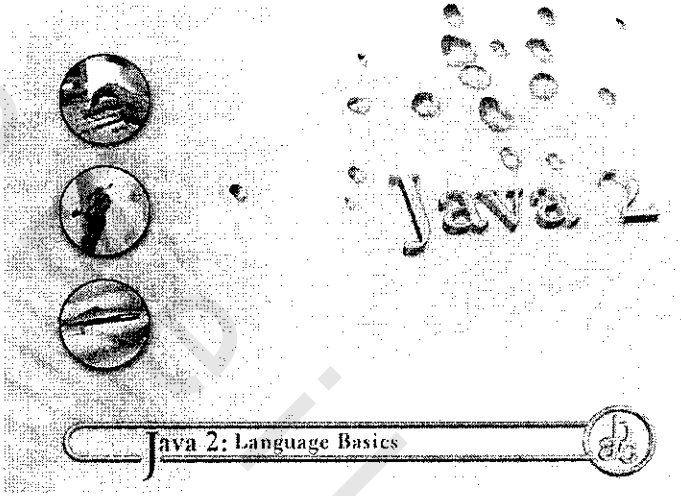
الفصل العاشر



الحزم Packages

في هذا الفصل نتعرف علي كيفية تجميع الفصائل classes معاً فى شكل حزم Packages وذلك من خلال النقاط التالية:

- **الحزم Packages.**
- **إنشاء الحزم Creating Packages.**
- **استخدام الحزم packages.**
- **الاستيراد الثابت Static Import.**

الحزم Packages:

- تخيل أن عندك مكتبة كبيرة فيها عدد ضخم من الكتب ، وتريد أن تبحث عن كتاب معين ، بالطبع ستبذل مجهوداً ضخماً إذا لم يكن عندك تصنيف وترتيب معين للكتب ، والأفضل أن يتم وضع الكتب المتشابهة مع بعضها البعض في مكان واحد لتسهيل عملية البحث.
- بالمثل في لغة Java ، فعندنا عدد ضخم من الفصائل classes ونريد تسهيل عملية التعامل مع هذه الفصائل classes ، ولذلك فإننا نضع كل مجموعة من الفصائل classes المتشابهة مع بعضها البعض في حزمة Package واحدة.
- إذن الحزم Packages هي عبارة عن مجلدات Directories تنظم الفصائل classes بحيث أن كل مجموعة فصائل classes لها وظائف مشتركة فى لغة Java يتم وضعها فى حزمة Package معينة.
- وقد تعاملنا فيما مضى مع الحزم Packages الخاصة بلغة Java مثل الحزمة javax.swing التي تحتوي علي الفصائل classes الخاصة بأدوات مكتبة الـ Swing وبالمثل باقي المكتبات packages التي تعاملنا معها من قبل.
- وهذا الأسلوب لا تخلو منه لغة برمجة لأن بدونه تصبح جميع الفصائل classes مع بعضها البعض في مكان واحد بدون تنظيم ولا تقسيم ، وعملياً يتم وضع جميع الفصائل classes التى تؤدى وظائف متشابهة معاً ، مثل فصائل الرسم Graphics وأدوات مكتبة Swing كما أشرنا.
- وتأتي هذه الحزم Packages مبنية فى لغة Java ، فكيف نستطيع بناء حزم Packages خاصة بنا ولماذا؟
- نريد أن ننشئ الحزم Packages الخاصة بنا للأسباب الآتية.
- عندما تنشئ نظاماً معقداً ، فيفضل وضع مجموعة الوظائف الخاصة بجزء معين من النظام فى حزمة package خاصة بها ، فمثلاً قد يكون النظام يتعامل مع شبكات Networks أو رسومات Graphics أو قاعدة بيانات Database أو واجهة

مستخدم User Interface ، ولذلك يفضل إنشاء حزمة package لكل وظيفة على حدة بأن تضع حزمة package للشبكات وحزمة package للرسومات وهكذا.

- من الممكن إعادة استخدام الحزم Packages فى برامج أخرى ، فإذا كنت أنشأت حزمة Package توفر وظيفة عامة فستطيع إعادة استخدامها مباشرة.
- ولا بد أن يتم تعريف المترجم Compiler بالحزم Packages التي نستخدمها في كل فصيلة class حتي يمكننا استخدام أي فصيلة class تنتمي إلي هذه الحزمة Package.
- فمثلاً:

```
import java.util.*;
```

- فى هذا السطر يتم استيراد (الإشارة إلى) import جميع فصائل classes الحزمة util الموجودة فى الفهرس java ، وبالتالى يمكن استعمال الفصائل classes الموجودة بها كما فى السطر التالي:

```
Date today = new Date();
```

- فى هذا السطر يتم تعريف هدف object بالاسم today من نوع الفصيلة Date ، وإذا لم يتم استيراد import الحزمة Package التى تحتوى على الفصيلة Date ، فلن يتعرف المترجم Compiler علي هذه الفصيلة class وسيعطى خطأ.
- ويمكن استيراد import فصيلة class محددة موجودة بالحزمة Package كما في السطر التالي:

```
import java.util.Date;
```

- فى هذا السطر تم تحديد استيراد import الفصيلة Date من الحزمة java.util.
- لاحظ أن استيراد import جميع فصائل الحزمة Package لا يزيد من حجم البرنامج.

ملحوظة:

لا يمكن استيراد import جميع حزم Packages فهرس محدد ، فمثلاً من غير المقبول كتابة السطر التالي للاشارة إلى جميع حزم Packages الفهرس java:

```
 import java.*
or
 import java.*.*
```

- يجب مراعاة عدم الإشارة الى حزم Packages تحتوى على فصائل classes بـنفس الاسم وإلا سيحدث تداخل Conflict بينهما كما فى السطور التالية:

```
import java.util.*;
import java.sql.*;
```

- المشكلة فى هذين السطرين هو احتـواء الحـزمتين Packages علـى فصـيلة class بالاسم Date مما يسبب تداخلاً.
- ولو افترضنا أنك تريد استعمال كلا الفصيلتين Date من كلا الحزمتين Packages ، فيتم الإشارة إليهما صراحة كما فى السطور التالية:

```
java.util.Date deadline = new java.util.Date();
java.sql.Date today = new java.sql.Date(...);
```

إنشاء الحزم Creating Packages:

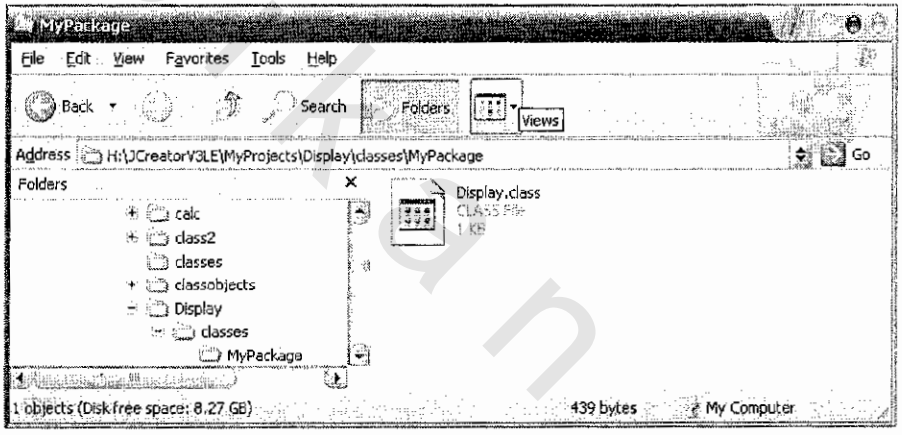
- لنفترض الآن أننا سننشىء فصيلة class بها دالـة Method اسمهـا printData() مهمتها طباعة رسالة على الشاشة ، ونريـد أن نسـتخدم هـذه الدالـة Method فـى فصيلة class أخري علي أن يتم تنظيم الفصائل classes في حزم Packages.
- سنقوم في الخطوات التالية بتوضيح كيفية تنفيذ ذلك.
- قم بإنشاء تطبيق جديد واكتب فيه السطور التالية كما فى الشكل (10-1).

```
1  package MyPackage;
2  public class Display {
3
4      public void printData ()
5       {
6  System.out.println("This is class Display");
7
8      }
9  }
10
```

(الشكل 10-1)

شرح السطور:

- فى السطر رقم 1 يتم إنشاء حزمة Package اسمها MyPackage ، حيث ينتج عن ذلك إنشاء مجلد Directory فى نفس مسار الملف كود البرمجة.
- ولكى تستطيع استخدام الفصيلة Display واستدعاءها من حزمة package أخرى ، فلابد أن يتم استخدام المعدل العام public لكى تستطيع استخدامها.
- فى السطر رقم 4 يتم تعريف الدالة ()printData التى تطبع رسالة على الشاشة.
- وبعد كتابة البرنامج بدون أخطاء وترجمته من داخل البرنامج JCreator ، يتم إنشاء المجلد MyPackage كما فى الشكل (10-2).



(**الشكل 10-2**)

- فى هذا الشكل تلاحظ إنشاء مجلد Directory بالاسم MyPackage وبداخله نجد الملف Display.class.
- ولعمل ترجمة Compile لهذا البرنامج من خلال شاشة الدوس Dos ، فقم بفتح شاشة الدوس Dos وقم بتحويل المسار إلى مسار ملف كود البرمجة.
- قم بترجمة Compile البرنامج ليتم إنشاء الحزمة package التى سميناها MyPackage ويتم وضع الفصيلة class الناتجة فى المجلد MyPackage ، ولتنفيذ ذلك نكتب الأمر التالى.

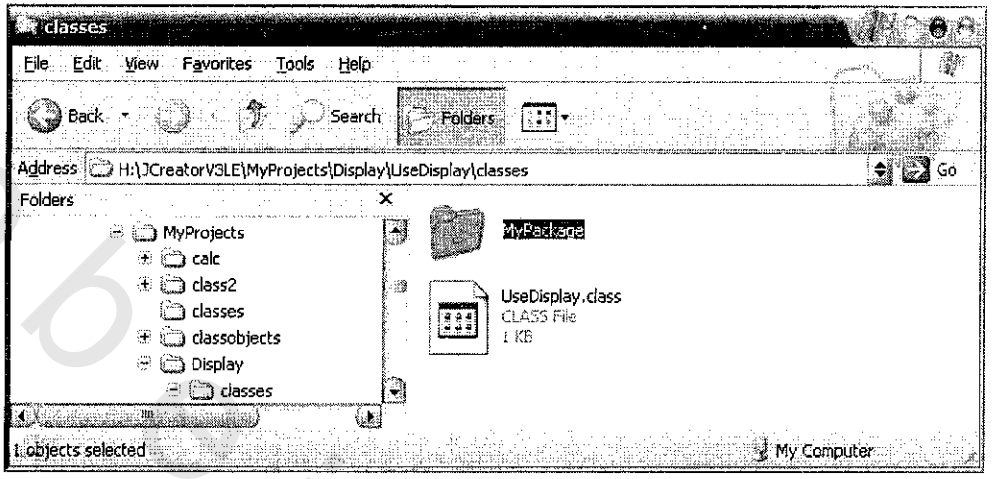
```
javac –d . Display.java
```

- الاختيار d– يقوم بإنشاء مجلد Directory مع ملاحظة وجود مسافة بعد d– ثم يتم وضع نقطة ثم ترك مسافة ثم اكتب Display.java – علي افتراض أن اسم الملف هو Display.java– ، وجود النقطة هنا ضروري لأنها تشير إلى المسار الحالى لأنك تريد أن تنشيء الحزمة MyPackage فى هذا المسار.
- قم بفتح مسار ملف كود البرمجة للملف Display.java وستلاحظ أنه تم إنشاء المجلد MyPackage ، قم بفتح هذا المجلد Directory وستلاحظ وجود الملف Display.class فى هذا المجلد Directory.
- قم بالرجوع إلى مسار الملف Display.java ثم قم بإنشاء ملف جديد باسم UseDisplay.java والذي يقوم باستخدام الحزمة package التي أنشأناها في البرنامج السابق. لاحظ أن هذا الملف لابد أن يكون فى نفس مسار المجلد MyPackage.
- الآن قم بإنشاء تطبيق جديد لاستعمال الحزمة MyPackage التى تم إنشاؤها فى الخطوة السابقة وذلك كما فى الشكل (10–3).

```
1  import MyPackage.Display;
2  public class UseDisplay {
3
4      public static void main(String[] args) {
5        Display d = new Display ();
6            d.printData ();
7
8      }
9  }
```

(الشكل 10–3)

- لاحظ أن مسار المجلد MyPackage تم وضعه فى نفس مسار البرنامج كما فى الشكل (10–4).



(الشكل 10-4)

شرح السطور:

- في الملف UseDisplay.java بدأ البرنامج باستيراد import الفصيلة class المسماه Display الموجودة في الحزمة MyPackage.
- في السطر رقم 5 يتم إنشاء هدف object من نوع الفصيلة Display. لاحظ أن الفصيلة Display تم تعريفها في الحزمة MyPackage.
- في السطر رقم 6 يتم استدعاء الدالة ()printData التي هي أحد دوال Methods الفصيلة Display.
- ترجم Compile هذا الملف بالطريقة العادية كالتالي:

```
javac UseDisplay.java
```

- قم بتنفيذ البرنامج عن طريق الأمر java UseDisplay وستلاحظ طباعة الرسالة عن طريق الدالة ()printData أو قم بتنفيذ البرنامج من خلال أحد البرامج المخصصة لذلك كما أشرنا من قبل.

الاستيراد الثابت Static Import:

من المزايا التى أضيفت ابتداء من الاصدار JDK 5 إمكانية إضافة الكلمة static بعد الأمر import مما يوفر فى استدعاء الدوال Methods وذلك كما فى السطور التالية:

```
1.  import static java.lang.System.*;
2.  class x
3.  {
4.  public staic void main String args[p])
5.  {
6.  out.println("Goodbye, World!"); // i.e., System.out
7.  exit(0); // i.e., System.exit
8.  }
```

شرح السطور:

فى السطر رقم 1 تم إضافة الكلمة static بعد كلمة import مما سهل استعمال دوال Method الفصيلة System –التي تم استيرادها import– بدون الإشارة إلى اسم الفصيلة System كما فى السطر رقم 6.

وتتضح فائدة ذلك عند التعامل مع بعض الدوال Methods التى لا تحتاج أسماء الفصائل classes كما فى السطور التالية ، وذلك بعد استعمال الاستيراد الثابت static import للفصيلة Math:

```
sqrt(pow(x, 2) + pow(y, 2))
```

وبالتالى يكون ذلك أوضح وأبسط من السطر التالى:

```
Math.sqrt(Math.pow(x, 2) + Math.pow(y, 2))
```

وكذلك بالنسبة للثوابت كما فى السطر التالي:

```
if (d.get(DAY_OF_WEEK) == MONDAY)
```

بدلاً من

```
if (d.get(Calendar.DAY_OF_WEEK) == Calendar.MONDAY)
```

ملخص الفصل:

- تعلمنا في هذا الفصل كيفية استخدام وإنشاء الحزم Packages.
- في الفصل القادم نتعلم - بإذن الله- كيفية استخدام الـ Interfaces والفصائل الداخلية Inner Classes ، فتابع معنا الفصل القادم.