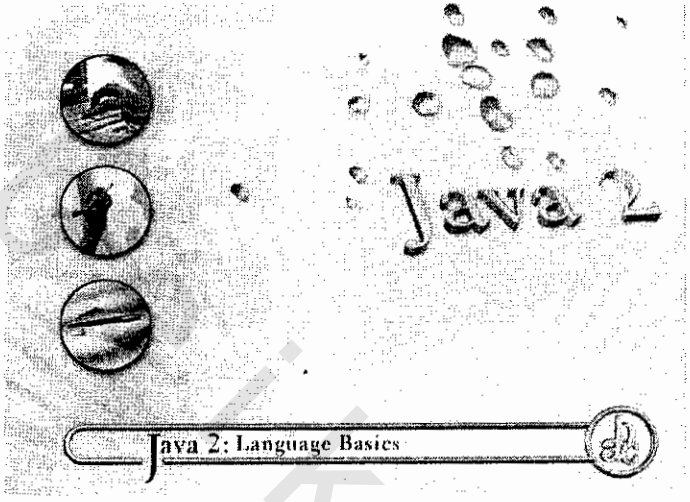
الفصل التاسع عشر



تطبيقات البرمجة موجهة الهدف

Object-Oriented Programming (OOP) Applications

في هذا الفصل نتعرف علي العديد من التطبيقات الهامة للغة Java حيث نتعرف علي العديد من تطبيقات البرمجة موجهة الهدف Object-Oriented Programming (OOP).

Online Kan.com

obeikan.com

مقدمة:

- نتعلم في هذا الفصل مشاريع البرمجة موجهة الهدف Object-Oriented Programming (OOP).

مثال (1): تصميم الفصائل Classes:

أولاً: هدف المثال:

يهدف هذا المشروع إلى إعطائك فكرة عن كيفية التفكير وتصميم وكتابة الفصائل classes ، وكيفية وضع المتغيرات variables والدوال Methods في الفصائل classes. هذا البرنامج يختبر قدرتك على التفكير بطريقة البرمجة موجهة الهدف Object-Oriented Programmıng.

وإليك المطلوب في البرنامج:

فى شركة ما ، يوجد قسمان هما القسم الهندسى وقسم المبيعات. يحتوى القسم الهندسى على 3 مهندسين ويحتوى قسم المبيعات على 2 موظفين.

البيانات المطلوبة من المهندسين هي: الاسم والمدينة والعمر والجامعة التى تخرج منها والقسم الذى تخرج منه فى الكلية ولغته الأم.

البيانات المطلوبة من موظفى المبيعات هي: الاسم والمدينة والعمر والجامعة التى تخرج منها ولغته الأم بالإضافة إلى لغته الأولى.

يجب أن يكون البرنامج قادراً علي طباعة بيانات هؤلاء الموظفين.

وعند ملء بيانات الموظفين ، إذا لم يتم تحديد اللغة الأم فاعتبر أنها العربية.

نريد أيضاً وضع كلمة سر على البرنامج بحيث لا يتم السماح بتنفيذ البرنامج إلا بكلمة السر.

ثانياً: خطوات الحل:

الخطوة الأولى:

- كما تعلم ، فإن أي برنامج مكتوب بلغة Java يتكون من مجموعة من الفصائل classes. إذن فالخطوة الأولى هى تحديد الفصائل classes التي نحتاجها فى البرنامج.

إذا نظرت للمطلوب في البرنامج ، ستجد أنه يوجد فصيلتان classes 2 وهما فصيلة

class للمهندسين وفصيلة class أخري للمبيعات. لن نحتاج في البرنامج أي فصائل classes أخرى. إذن سوف ننشئ فصيلة class للمهندسين وفصيلة class للمبيعات.

الخطوة الثانية:

- فى كل فصيلة class حدد المتغيرات variables والدوال methods التى تحتاجها. بالنسبة للمتغيرات لابد من تحديد نوعها Data Type واسمها ، وبالنسبة للدوال methods فلابد من تحديد المعاملات Parameters التى تستقبلها الدالة method ونوع القيمة التى سترجع من الدالة Return Type بالإضافة إلي اسم الدالة method بالطبع.
- كما ذكرنا فعندنا فصيلة class للمهندس. ما هي المتغيرات variables بالنسبة للمهندس؟ إذا نظرت للمسألة ستجد أننا نحتاج متغيرات للبيانات الآتية: الاسم والمدينة والعمر والجامعة والقسم واللغة الأم. وبعد تحديد المتغيرات المطلوبة نقوم باختيار نوع المتغيرات المناسب Data Type. طبعاً بالنسبة للاسم والمدينة والجامعة والقسم واللغة الأم ، فإن نوع المتغير المناسب هو String لأن هذه المتغيرات تمثل نصاً ، أما بالنسبة للعمر فحيث أنه رقم فإننا سنختار نوع المتغير byte. قد تسأل لماذا لم نختر int؟
- الإجابة هو أننا دائماً نريد أن نقلل من حجم المتغيرات من أجل توفير مساحة فى الذاكرة. بالطبع أنت تعلم أن المتغير من نوع int يحجز 4 بايت Bytes والمتغير من نوع short يحجز 2 بايت Bytes والمتغير من نوع byte يحجز 1 بايت Byte ، فدائماً نحاول اختيار النوع الذى يحجز أقل مساحة ممكنة وفى نفس الوقت يكفى مدي الأرقام التي يمكن أن يأخذها هذا المتغير. إذا نظرت للبايت byte ستجد أن أقصى قيمة له هو 127 ، فهل يعقل أن عمر أى شخص يتجاوز 127!!! طبعاً أطال الله فى عمرك ولكن حتى إذا تجاوز عمرك 127 فإنه لا يمكنك العمل بعد سن الستين. إذن فاختيارك للبايت byte جاء نتيجة أنه صاحب أقل مساحة ممكنة وفى نفس الوقت يعطينا المدى الذى نريده للأرقام التي يمكن أن يأخذها المتغير الخاص بالعمر.

- بالنسبة للفصيلة class الخاصة بالمبيعات ، ستجد أن المتغيرات الخاصة بها هى: الاسم والمدينة والعمر والجامعة واللغة الأم واللغة الأولى. بنفس الطريقة فإن نوع المتغير المناسب للاسم والمدينة والجامعة واللغة الأم واللغة الأولى هو String وبالنسبة للعمر سنختار byte.
- بالنسبة للدوال methods فستجد أننا نحتاج دالة method لطباعة بيانات الموظف ، ولذلك فكل فصيلة class سواء المهندس أو المبيعات تحتاج لدالة method تطبع جميع المتغيرات الموجودة فى الفصيلة class.
- الرسم التالى يوضح ناتج الخطوتين الأولى والثانية.

Engineer	Sales
String name String city byte age String university String nativeLanguage String department	String name String city byte age String university String nativeLanguage String firstLanguage
void displayDetails()	void displayDetails()

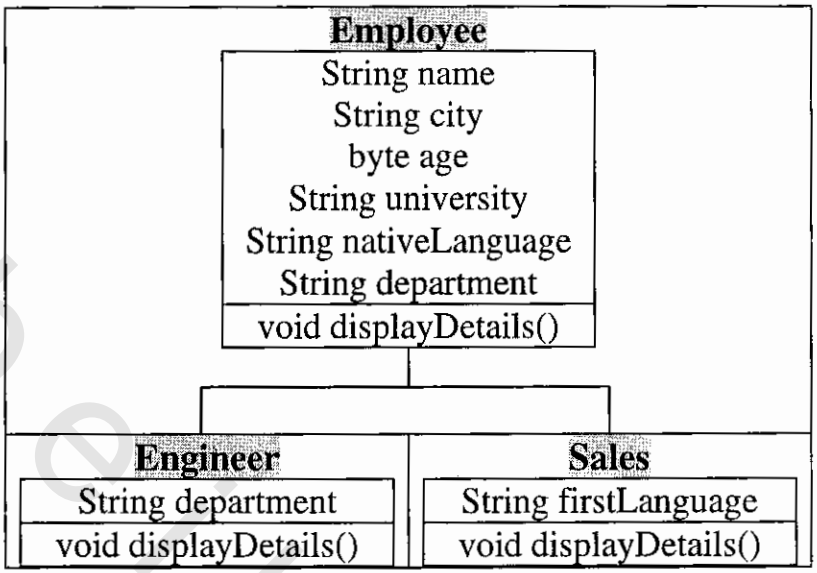
- لاحظ أن اسم الفصيلة class مكتوب في أعلي الجدول ويليه أسماء المتغيرات ونوعها ثم بعد الخط الفاصل تأتي الدوال methods.
- لاحظ أن المطلوب من الدالة displayDetails() هو طباعة بيانات الشخص ولذلك فلا نحتاج لتمرير قيمة لها ولا نحتاج أن نستقبل قيمة منها ، ولذلك فإن نوع القيمة المرتجعة Return Type منها هو void.
- حاول دائماً اختيار أسماء المتغيرات والدوال methods جيداً بحيث تعطى فكرة عن هدف المتغيرات والدوال methods ووظيفتها بمجرد قراءة اسمها ، فمثلاً بالنسبة للاسم اخترنا متغيراً اسمه name ليدل عليه ، فإذا قرأت name فى أى جزء من البرنامج بعد ذلك فستعرف مباشرة أنه متغير خاص بالاسم. أما إذا سميت هذا المتغير s مثلاً فهو لا يدل عليه مطلقاً ، وعندما تقوم بتتبع البرنامج وتجد s فلابد أن

تبحث فى تعريفات المتغيرات الموجودة في الفصيلة class علي المتغير s لتعرف ما الذى يدل عليه.

- لاحظ أن تسمية المتغيرات لا تسبب اختلافاً فى تنفيذ البرنامج ولكنها تساعدك فيما بعد عند تتبعك البرنامج أو عند مراجعتك له فيما بعد خاصة إذا كانت هناك مدة طويلة بين كتابتك لكود البرمجة وبين مراجعتك له.

الخطوة الثالثة:

- لابد من معرفة العلاقة بين الفصائلclasses . المقصود بمعرفة العلاقة هو تحديد المتغيرات variables والدوال methods المتشابهة بين الفصائل classes التى أنشأناها. لاحظ أنه فى الخطوة الأولى حددنا الفصائل classes وفى الخطوة الثانية حددنا المتغيرات والدوال methods فى كل فصيلة class. إذن يمكننا الآن تحديد المتغيرات والدوال method المتشابهة.
- إذا نظرت لتكوين الفصائل classes فى الرسم السابق ، فستجد أن المتغيرات الخاصة بالاسم والمدينة والعمر والجامعة واللغة الأم هى متغيرات متشابهة وأيضاً الدالة displayDetails() هى دالة method مشتركة.
- دائماً يهدف البرنامج إلى تقليل إعادة كتابة الكود ولذلك فإن الحل الأفضل هو إنشاء فصيلة class تحتوى على المتغيرات والدوال methods المشتركة ثم الوارثة Inheritance من هذه الفصيلة class بدلاً من إعادة كتابة المتغيرات والدوال methods مرتين في كل فصيلة class.
- لاحظ أن غرض الوراثة Inheritance الأساسي هو تقليل تكرار المتغيرات من أجل توفير حجز مساحة المتغيرات وبالإضافة إلى ذلك ، فإنك إذا قمت بالوراثة Inheritance من فصيلة class فإنك ترث جميع المتغيرات والدوال methods الموجودة فيها ويمكنك الإضافة إليها ، ولذلك فإننا سننشئ فصيلة class تحتوى على المتغيرات والدوال methods المتشابهة وسنطلق عليها اسم Employee.
- وبذلك سنعيد بناء الفصائل classes لتصبح كما هي موضحة في الشكل التالى.



- تذكر أن لغة Java لا تدعم خاصية الوراثة المتعددة Multiple Inheritance بمعنى أنه لا يمكن أن توجد فصيلة class ترث خواصها من أكثر من فصيلة class أخري ، ولكن يمكن أن توجد أكثر من فصيلة class ترث خواصها من نفس الفصيلة class.
- بمعنى أن الفصيلتين Engineer وSales ترثان من الفصيلة Employee وهذا صحيح تماماً ، ولكن كان سيصبح خطأ لو قلنا مثلاً أن الفصيلة Sales ترث من Employee و Engineer فى نفس الوقت.

ثالثاً: كود البرمجة:

يحتوي هذا البرنامج علي ثلاثة فصائل classes هم Sales و Engineer و Employee بالإضافة إلي فصيلة class أخري للدالة الرئيسية ()main.

فيما يلي كود الفصيلة Employee:

```
1: public abstract class Employee
2: {
3:   public static final String DEFAULT_LANGUAGE =
   "arabic";
4:
```

```
5:   private String name;
6:   private String city;
7:   private String university;
8:   private byte age;
9:   private String nativeLanguage;
10:
11:  protected void displayDetails()
12:  {
13:       System.out.println("Name: "+name);
14:       System.out.println("City: "+city);
15:       System.out.println("University: "+university);
16:       System.out.println("Age: "+age);
17:       System.out.println("Native                Language:
     "+nativeLanguage);
18:  }
19:
20:  Employee(String      name,String      city,String
     university,byte age,String nativeLanguage)
21:  {
22:       this.name = name;
23:       this.city = city;
24:       this.university = university;
25:       this.age = age;
26:       this.nativeLanguage = nativeLanguage;
27:  }
28:}
```

شرح الفصيلة Employee:

- كما ذكرنا في خطوات الحل ، فإننا سنقوم بإنشاء الفصيلة Employee التي تحتوى على الخواص المتشابهة كما سبق لنا شرحه.
- في هذا الملف يتم إنشاء هذه الفصيلة class ولاحظ أن الدالة displayDetails() تقوم بطباعة البيانات المتشابهة فقط (راجع السطر رقم 20).
- لاحظ استخدام المعدل modifier المسمي protected قبل اسم الدالة

displayDetails() وهذا جاء نتيجة أننا نريد أن نجعل هذه الدالة method متاحة للفصائل الصغري child classes وليس لأي فصيلة class أخري.

- لاحظ وضع كلمة abstract كما هو واضح في السطر رقم 1. قد تتعجب من استخدامها ولكننا هنا نوضح المبدأ الخاص بالتجريد abstraction. تذكر أن التجريد abstraction هو النظر إلى الشيء نظرة مجردة ، أي أنه يتم التركيز على خواصه الأساسية دون الاهتمام بالتفاصيل ، ولذلك فإنك تقوم بتكوين فصيلة class غير كاملة التفاصيل.
- إذا طبقت هذا المبدأ هنا فستجد أنك أنشأت فصيلة class تسمى Employee ووضعت فيها الخواص الأساسية (أى المتشابهة بين مجموعة من الفصائل classes) ولم تهتم بالتفاصيل الخاصة مثل المتغير department الخاص بالفصيلة Engineer ومثل أيضاً المتغير firstLanguage الخاص بالفصيلة Sales.
- تذكر أن الفصيلة class إذا كانت مجردة abstract فإنه لا يمكنك إنشاء هدف object منها مباشرة ولكن لابد أن تنشئ فصيلة class جديدة ترث من هذه الفصيلة المجردة abstract class ومن ثم يمكنك إنشاء هدف object من الفصيلة class الجديدة.
- هنا سيتضح لك معنى كلمة abstract ، فتخيل أنك أنشأت هدفاً object من نوع Employee فماذا يمثل لك؟ في الحقيقة أنه لا يمثل شيئاً ملموساً بسبب عدم وجود تفاصيل كاملة له ، إذن فلا معنى لإنشاء هدف object منه.
- كمثال واقعى فإن جميع الثدييات تلد وتتنفس ، ولكن هذه التفاصيل ناقصة ولا تمثل كائن حى بعينه ، ولكن يمكنك القول أن الإنسان يرث من الثدييات جميع صفاتها ويزيد عليها أن له عقل يفكر ، ومن ثم يمكنك القول أن الشخص المسمي أحمد هو إنسان. إذن أحمد هو شئ واقعى ملموس ولكن الثدييات ليست كائناً موجوداً فعلياً فعندما تصف أحمد فستقول أنه إنسان وليس ثدييات.
- قد تسأل سؤالاً: ماذا يحدث إذا لم أضع الكلمة abstract؟
- الإجابة هى أنه لا شئ يحدث ، بمعنى عدم وجود تأثير على تنفيذ البرنامج ولكننا نوضح المبدأ الخاص بالتجريد abstraction وأيضاً نقوم الإشارة إلى أنه إذا كان عندك برنامج كبير جداً ، فمن المهم توضيح وظيفة كل جزء وبيان ما إذا كانت

تفاصيله كاملة أم ناقصة ، ولذلك تعود على وضع كلمة abstract أمام الفصائل classes الناقصة التى لا يمكن إنشاء هدف object منها مباشرة.

- قد تتعجب من وجود دالة بناء Constructor في الفصيلة Employee بالرغم من أنها فصيلة مجردة abstract class ، والحقيقة أن وجود دالة البناء Constructor سببه الأساسي هو الوراثة Inheritance ، تذكر أنه عندما تنشئ هدفاً object من أي فصيلة class ، فإنه يتم تلقائياً استدعاء دالة البناء Constructor الخاصة بالفصيلة الأم Super class ، ولذلك تم وضع دالة البناء Constructor في الفصيلة Employee بالرغم من أنها فصيلة مجردة abstract class.
- لاحظ أيضاً استخدام المعدل modifier المسمي private قبل اسم المغيرات وهذا جاء نتيجة تطبيق مبدأ الكبسلة Encapsulation الذي سبق لنا شرحه في الفصول السابقة.
- الملحوظة الأخيرة هي وجود المتغير النهائي final المسمي DEFAULT_LANGUAGE (راجع السطر رقم 3) ، وهذا المتغير يمثل القيمة الافتراضية للغة الأم التي ستكون اللغة العربية في حالتنا هذه.

فيما يلي كود الفصيلة Engineer:

```
1:  public class Engineer extends Employee
2:  {
3:    String department;
4:
5:    public void displayDetails()
6:    {
7:        super.displayDetails();
8:        System.out.println("Department: "+department);
9:    }
10:
11:  Engineer(String name,String city,String university,
12:  byte age,String nativeLanguage,String department)
13:  {
14:       super(name,city,university,age,nativeLanguage);
15:       this.department=department;
```

```
16:  }
17:
18:  Engineer(String name,String city,String university,
19:  byte age,String department)
20:  {
21:        this(name,city,university,age,
     DEFAULT_LANGUAGE,department);
22:  }
23:}
```

شرح الفصيلة Engineer:

- في هذا الملف يتم إنشاء الفصيلة Engineer. لاحظ أن الفصيلة Engineer ترث من الفصيلة Employee ولذلك استخدمنا الكلمة extends كما هو واضح في السطر رقم 1.
- في السطر رقم 2 يتم إضافة المتغير الخاص بالفصيلة Engineer وهو المتغير الخاص بالقسم.
- في السطر رقم 5 يتم تعريف دالة البناء constructor لهذه الفصيلة class. سوف تستقبل دالة البناء constructor عدد 6 متغيرات وتستخدمهم لتحديد جميع قيم المتغيرات variables في الفصيلة Engineer مع ملاحظة استخدام كلمة super في السطر رقم 8 ، حيث تستخدم كلمة super هنا لاستدعاء دالة البناء constructor الخاصة بالفصيلة الأم super class أي الفصيلة Employee حيث يتم تحديد قيم 5 متغيرات (راجع الفصيلة Employee في السطور من 11 إلي 18).
- بالنسبة للمتغير الباقي (department) فإنه يتم تحديد قيمته في السطر رقم 9.
- لاحظ أن اسم المتغيرات variables التى تستقبلها دالة البناء constructor لها نفس اسم المتغيرات فى الفصيلة class ، ولذلك لابد من استخدام كلمة this للتفريق بين المتغيرات.
- الآن سنحل مشكلة ذكرناها فى رأس المسألة وهى: أنه فى حالة عدم تحديد اللغة الأم فإنك تعتبرها اللغة العربية. ولحل هذه المشكلة فإننا سننشئ دالة بناء constructor أخري تستقبل خمس قيم فقط لأنها لن تستقبل المتغير الخاص باللغة الأم لأنه في هذه الحالة ليس متغيراً ولكنه قيمة ثابتة وهي اللغة العربية.

- الحل الطويل هو أنه فى دالة البناء constructor الجديدة ، سنستخدم الخمس القيم التى استقبلتها لتحديد قيمة خمس متغيرات فقط من الفصيلة class ثم نضع قيمة المتغير الخاص باللغة الأم بالقيمة عربى. أي يتم كتابة دالة البناء constructor لتصبح كالتالي:

```
Engineer ( String name , String city , byte age , String university ,
String department)
{
this.name = name;
this.city = city;
this.age = age;
this.university = university;
this.nativeLanguage = "Arabic";
this.department = department;
}
```

- يتم استخدام القيم التى استقبلتها دالة البناء constructor لتحديد قيم المتغيرات فى الفصيلة class المسماه Engineer.
- لاحظ أن هذه السطور تتشابه مع دالة البناء constructor الموجودة في الفصيلة الأم Employee مع اختلاف السطر قبل الأخير فقط والذي وضعنا فيه قيمة المتغير الخاص باللغة الأم بقيمة ثابتة وهي العربية. لاحظ أيضاً أن دالة البناء الجديدة constructor تستقبل خمس قيم فقط. إذن فى هذه الحالة ستكرر كتابة الكود الذي كتبته في دالة البناء constructor الخاصة بالفصيلة الأم Employee مرة أخرى مع اختلاف سطر واحد فقط ، وكبديل فإننا من داخل دالة البناء constructor التى تستقبل الخمس قيم سنقوم باستدعاء دالة البناء constructor التى تستقبل ستة قيم. السؤال الآن: إذا كانت دالة البناء constructor تستقبل خمس قيم ، إذن كيف يمكننا استدعاء دالة البناء constrcutor الأخري التي تستقبل ستة قيم؟
- الحل هو أن نستخدم الخمس قيم التي استقبلتها دالة البناء constructor والقيمة الأخيرة سنضعها ثابتة وهى اللغة العربية ثم نمرر تلك الست متغيرات إلي دالة البناء constructor الأولي. إذن نحتاج إلى متغير يشير إلى دالة البناء constructor التى من نفس نوع الهدف object الحالى ولذلك استخدمنا كلمة this (راجع السطر رقم 15).

- ما فعلناه هنا هو أنه عند استقبالنا للخمس القيم ، فإننا وضعنا القيمة السادسة بقيمة ثابتة هى اللغة العربية فأصبحوا 6 قيم ، ثم استدعينا دالة البناء constructor الأخري مع تمرير الستة قيم لها.
- إذن نتعلم من هنا أنه إذا أردت استدعاء دالة بناء constructor من داخل دالة بناء constructor أخري فإنك تستخدم this وهذا أحد استخداماتها.
- لاحظ أنه لكى تستدعى دالة بناء constructor من داخل دالة بناء constructor أخري ، فلابد أن يتم الاستدعاء فى أول سطر من دالة البناء constructor وإلا فسيحدث خطأ.
- لاحظ أن القيمة DEFAULT_LANGUAGE ليست موجودة مباشرة فى الفصيلة Engineer ولكنها موجودة فى الفصيلة Employee ، ولكن حيث أن الفصيلة Engineer ترث من الفصيلة Employee ، إذن فجميع متغيرات الفصيلة Employee تكون أيضاً موجودة فى الفصيلة Engineer.
- تذكر أن تمرير القيم لدالة البناء constructorلابد أن يتم بنفس ترتيب القيم التى تستقبلها دالة البناء constructor.
- الآن لابد من تعريف الدالة displayDetails() التى تطبع قيم المتغيرات فى الفصيلة class.
- أيضاً تذكر أن الدالة displayDetails() الموجودة فى الفصيلة Employee (راجع السطر رقم 20) تطبع قيم بعض المتغيرات فى الفصيلة Engineer ، ولذلك بدلاً من كتابة الكود وتكراره ، فإننا سنستدعى الدالة displayDetails() الخاصة بالفصيلة Employee مع وضع الكود الزائد فقط ، ولكي تستطيع استدعاء الدالة displayDetails() الموجودة فى الفصيلة Employee التي تمثل الفصيلة الأم Super class للفصيلة Engineer ، فإنك تستخدم الكلمة super.
- لاحظ أنك لكي تستطيع استدعاء أى متغير أو دالة method من الفصيلة الأم Employee فإنك تشير إليه بالكلمة super ثم تكتب اسم المتغير أو الدالة method.
- إذن يقوم السطر رقم 20 باستدعاء الدالة displayDetails() الموجودة فى الفصيلة

Employee ويقوم السطر رقم 21 بكتابة الكود لطباعة المتغيرات الباقية وهو المتغير الخاص بالقسم.

فيما يلي كود الفصيلة Sales:

```
1:  public class Sales extends Employee
2:  {
3:    String firstLanguage;
4:
5:    Sales(String name,String city,String university,
6:    byte age,String nativeLanguage,String firstLanguage)
7:    {
8:        super(name,city,university,age,nativeLanguage);
9:        this.firstLanguage=firstLanguage;
10:   }
11:
12:   Sales(String name,String city,String university,
13:   byte age,String firstLanguage)
14:   {
15:       this(name,city,university,age,
      DEFAULT_LANGUAGE,firstLanguage);
16:   }
17:
18:   public void displayDetails()
19:   {
20:       super.displayDetails();
21:       System.out.println("first                    language
      "+firstLanguage);
22:   }
23:}
```

شرح الفصيلة Sales:

- في هذا الملف يتم إنشاء الفصيلة Sales التي تتشابه مع الفصيلة Engineer ولـذلك فلن نعيد الشرح ويمكنك تتبع هذه الفصيلة class بلا مشاكل.

فيما يلي كود الفصيلة Main:

```
1:  public class Main
2:  {
3:   public static void main(String args[])
4:   {
5:       try
6:       {
7:           if(!(args[0].equals("mostafa")))
8:           {
9:               System.out.println("You are not
    authorized");
10:              System.exit(0);
11:          }
12:      }
13:
14:      catch(Exception e)
15:      {
16:          System.out.println("You are not authorized");
17:          System.out.println(e);
18:          System.exit(0);
19:      }
20:
21:      Engineer engineers[] = new Engineer[3];
22:      engineers[0] = new
    Engineer("adham","alex","cairo",(byte)25,"english","el
    ectricity");
23:      engineers[1] = new
    Engineer("ahmed","alex","alex",(byte)22,"french","elec
    tricity");
24:      engineers[2] = new
    Engineer("walid","damanhour","alex",(byte)26,"mecha
    nics");
25:
26:      Sales sales[] = new Sales[2];
```

```
27:     sales[0] = new
    Sales("rana","alex","alex",(byte)21,"spanish");
28:     sales[1] = new
    Sales("hana","cairo","cairo",(byte)28,"english","french"
    );
29:
30:     for(int i = 0;i<engineers.length;i++)
31:     {
32:         engineers[i].displayDetails();
33:
     System.out.println("**********************************
    **");
34:     }
35:
36:     for(int i = 0;i<sales.length;i++)
37:     {
38:         sales[i].displayDetails();
39:
     System.out.println("**********************************
    **");
40:     }
41: }
42:}
```

شرح الفصيلة Main:

- في هذا الملف يتم إنشاء الفصيلة Main المسئولة فقط عن إنشاء الدالة الرئيسية ()main.
- وظيفة الدالة ()main هى تحديد قيم المتغيرات للفصائل classes الموجودة. بالنسبة للمهندسين ، فكما ذكرنا يوجد عندنا 3 مهندسين ولذلك فإننا سننشئ مصفوفة array من الفصيلة class المسماه Engineer ونحدد حجمها بـ 3 أماكن كما فعلنا فى السطر رقم 21.
- في السطر رقم 22 يتم تحديد بيانات المهندس الأول. لاحظ تمرير 6 قيم لدالة البناء constructor. لاحظ أيضاً أن القيمة الرابعة (التي تمثل العمر) تساوي 25 ، ولكننا

إذا كتبنا 25 فذلك يعنى أنها رقم من نوع int وهذا يتعارض مع متغير العمر الذي من نوع byte ، ولذلك استخدمنا عملية التحويل Casting من أجل تحويل الرقم 25 من int إلى byte.

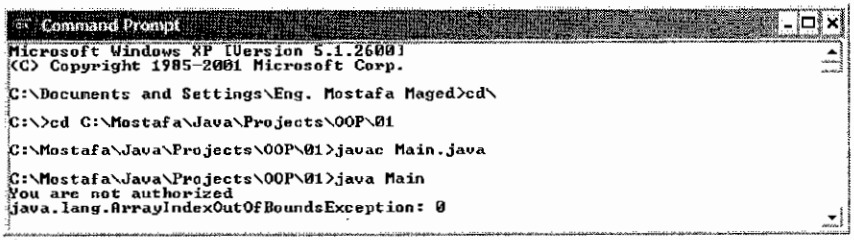
- في السطر رقم 23 يتم تكرار نفس الشيء ويتم تمرير 6 قيم لدالة البناء constructor.
- في السطر رقم 24 يتم تمرير خمس قيم فقط لدالة البناء constructor ، أي أنه سيستدعي دالة البناء constructor الموجودة في السطر رقم 12 في الفصيلة Engineer والتي بدورها تستدعي دالة البناء constructor الأخري المعرفة في السطر رقم 5 في الفصيلة Engineer وتمرر لها الخمس قيم مع إضافة اللغة الأم بقيمة العربية (راجع السطر رقم 15 في الفصيلة Engineer) وهذا يحقق الذى أردناه أنه إذا لم يتم تحديد اللغة الأم فإنك تعتبرها العربية.
- تذكر أن أول عنصر في المصفوفة array يبدأ بصفر وليس بواحد.
- نفس هذا الكلام كررناه بالنسبة للفصيلة Sales مع تذكر أنه يوجد عندنا 2 موظفين فقط فى هذا القسم. يمكنك قراءة السطور من 26 إلي 28 بلا مشاكل.
- بعد أن حددنا جميع قيم المتغيرات ، فإننا نريد أن نطبع هذه القيم ، ويتم ذلك عن طريق إنشاء دوارة for Loop ثم نستدعى الدالة displayDetails() كما فعلنا فى السطر رقم 32. هذه الدوارة Loop تبدأ من صفر حتى أقل من engineers.length. لاحظ أن استدعاء المتغير length الموجود فى المصفوفة array المسماة engineers تعيد لنا حجم المصفوفة array ، وحيث أن حجم هذه المصفوفة array كان 3 (راجع السطر رقم 21) فلذلك هذه الدوارة Loop تبدأ من صفر حتى أقل من 3 (أى صفر وواحد واثنان).
- كررنا نفس هذا العمل بالنسبة لقسم المبيعات كما فى السطور من 36 إلى 40.
- إذا نظرت للمسألة فستجد أننا حققنا جميع متطلبات المسألة عدا واحداً فقط وهو الخاص بوضع كلمة سر على البرنامج.
- تذكر أن تنفيذ البرنامج يبدأ من الدالة الرئيسية main() ويتم تنفيذه عن طريق كتابة السطر التالى فى شاشة الدوس Dos.

java Main

- هنا نريد تمرير قيمة للدالة ()main بحيث أن هذه القيمة تمثل لنا كلمة السر التي نريدها. لاحظ أن الدالة ()main تستقبل متغيراً يمثل مصفوفة array من نوع String ، إذن لتنفيذ البرنامج فإننا سنكتب java Main ثم كلمة السر المطلوبة. كلمة السر هذه ستكون هى القيمة [0]args.
- إذن سنختبر القيمة التى نمررها للدالة ()main ، فإذا كانت هى كلمة السر التى حددناها نفذنا البرنامج وإلا فلن يتم تنفيذ البرنامج.
- ولذلك ستجد أننا وضعنا الشرط الخاص بجملة if فى بداية الدالة الرئيسية ()main (راجع السطر رقم 7) فإذا كانت كلمة السر التى ستمررها وتستقبلها الدالة الرئيسية ()main في القيمة [0]args تساوي كلمة السر التي حددناها بكلمة mostafa أمكنك تنفيذ البرنامج وإلا فستظهر لك رسالة تبين عدم إمكانك تشغيل البرنامج ثم يتم إنهاء البرنامج عن طريق الدالة ()exit.
- لاحظ أنك إذا لم تدخل كلمة السر فذلك يعنى أنه لا توجد قيمة للمتغير [0]args ولذلك ستظهر لنا رسالة تفيد حدوث استثناء Exception ، وطالما أنك تتعامل مع مصفوفة array فإنه من الممكن حدوث استثناء Exception من نوع ArrayIndexOutOfBoundsException ، أى أنك حاولت استخدام عدد عناصر أكبر من عدد عناصر المصفوفة array ، ولذلك استخدمنا try وcatch كما هو واضح فى السطر رقم 5 و 14 لمعالجة هذا الاستثناء Exception حيث يتم أيضاً إنهاء البرنامج في حالة حدوثه.
- إذن لتنفيذ البرنامج فلابد أن نكتب السطر التالى في شاشة الدوس Dos:

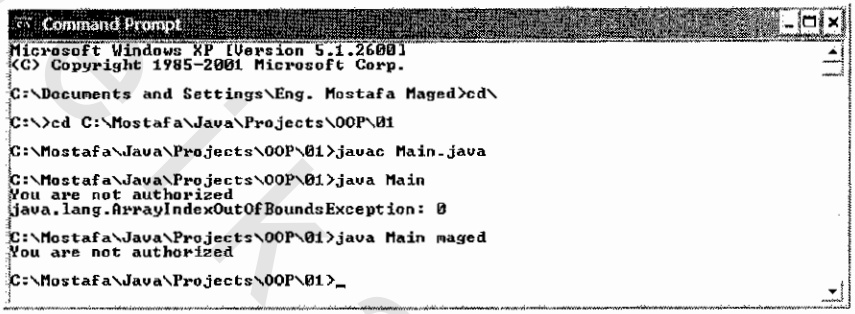
java Main mostafa

- إذا كانت كلمة السر صحيحة فسيتم تنفيذ البرنامج ، أما إذا كانت خاطئة فلن يمكن تنفيذ البرنامج وهذا يحقق هدفنا الذى أردناه.
- قم بترجمة البرنامج وتنفيذه ثم قم مرة بعدم إدخال كلمة سر ثم إدخال كلمة سر خاطئة مرة ثم إدخال كلمة سر صحيحة مرة أخري ولاحظ تنفيذ البرنامج كما أردنا.
- في الشكل (19-1) يتم توضيح شاشة البرنامج عند عدم إدخال كلمة سر.



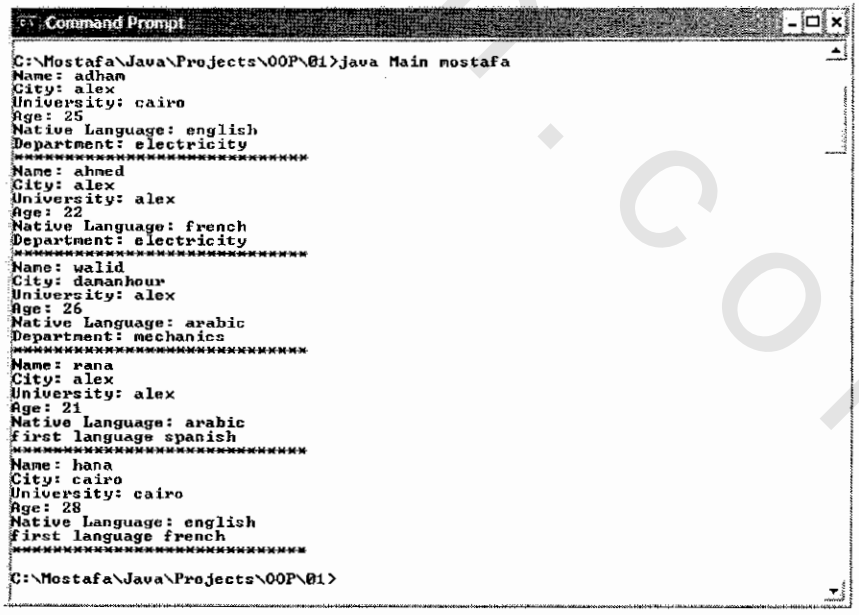
الشكل (19-1) تنفيذ البرنامج عند عدم إدخال كلمة سر

في الشكل (19-2) يتم توضيح شاشة البرنامج عند إدخال كلمة سر خاطئة.



الشكل (19-2) تنفيذ البرنامج عند إدخال كلمة سر خاطئة

في الشكل (19-3) يتم توضيح شاشة البرنامج عند إدخال كلمة سر صحيحة.



الشكل (19-3) تنفيذ البرنامج عند إدخال كلمة سر صحيحة

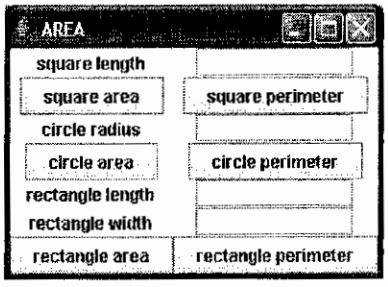
الأهداف المرجو تحقيقها الآن:

1. كيفية التفكير فى المسألة وتحويلها من كلام نصي إلى كود برمجة.
2. كيفية تحديد الفصائل classes وتحديد المتغيرات variables والدوال methods فيها.
3. تحديد نوع المتغيرات Data Types.
4. التعامل مع الوارثة inheritance والتجريد abstraction.
5. استخدام دالة البناء constructor.
6. كيفية استدعاء دالة بناء constructor من داخل دالة بناء constructor أخري.
7. كيفية استدعاء دوال methods من الفصيلة الأم super class.
8. كيفية وضع كلمة سر على البرنامج.
9. التعامل مع المصفوفات arrays.
10. التعامل مع الاستثناءات Exceptions.
11. استخدام الدوارة for Loop.

مثال (2): حساب المحيط Perimeter والمساحة Area:

أولاً: هدف المثال:

في هذا المثال نريد إنشاء برنامج يحتوي علي أداة إطار JFrame كما هو واضح في الشكل (19-4) ، حيث نريد حساب محيط Perimeter ومساحة Area مربع ومستطيل ودائرة.



الشكل (19-4) تنفيذ البرنامج

ثانياً: خطوات الحل:

- نحتاج في هذا المثال إلي أربع فصائل classes حيث نريد إنشاء فصيلة Square و Rectangle و Circle بالإضافة إلي الفصيلة Area المسئولة عن إنشاء نافذة البرنامج.
- الفصائل Square و Rectangle و Circle لا بد أن تحتوي جميعها علي الدوال methods التالية:
- ()calculateArea لحساب المساحة.
- ()calculatePerimeter لحساب المحيط Perimeter.
- ()getPerimeterDescription لإنشاء رسالة تبين المحيط Perimeter.
- ()getAreaDescription لإنشاء رسالة تبين المساحة.
- بالنسبة للفصيلة Square فإنها تحتوي علي متغير يمثل طول المربع.
- بالنسبة للفصيلة Rectangle فإنها تحتوي علي متغيرين يمثلان طول وعرض المستطيل.
- بالنسبة للفصيلة Circle فإنها تحتوي علي متغير يمثل نصف قطر الدائرة.
- هذا بالإضافة إلي احتواء كل فصيلة class علي دالة بناء constructor.

ثالثاً: كود البرمجة:

يحتوي هذا البرنامج علي ثلاثة فصائل classes هم Square و Rectangle و Circle بالإضافة إلي فصيلة class أخري لنافذة البرنامج اسمها Area.

فيما يلي كود الفصيلة Square:

```
1:  public class Square
2:  {
3:    private double length;
4:
5:  Square(double length)
6:  {
7:        this.length=length;
8:  }
9:
10:  public double calculateArea()
11:  {
12:       return length * length;
13:  }
14:
15:  public double calculatePerimeter()
16:  {
17:       return length * 4;
18:  }
19:
20:  public String getPerimeterDescription()
21:  {
22:       return "Perimeter of square = " +
     calculatePerimeter();
23:  }
24:
25:  public String getAreaDescription()
26:  {
27:       return "Area of square = " + calculateArea();
28:  }
29:}
```

obeikan.com

فيما يلي كود الفصيلة Rectangle:

```
1:  public class Rectangle
2:  {
3:    private double length,width;
4:
5:    Rectangle(double length,double width)
6:    {
7:        this.length=length;
8:        this.width=width;
9:    }
10:
11:  public double calculateArea()
12:  {
13:       return length * width;
14:  }
15:
16:  public double calculatePerimeter()
17:  {
18:       return 2 * ( length + width );
19:  }
20:
21:  public String getPerimeterDescription()
22:  {
23:       return "Perimeter of rectangle = " +
     calculatePerimeter();
24:  }
25:
26:  public String getAreaDescription()
27:  {
28:       return "Area of rectangle = " + calculateArea();
29:  }
30:}
```

فيما يلي كود الفصيلة Circle:

```
1:  public class Circle
2:  {
3:   double radius;
4:
5:  Circle(double radius)
6:  {
7:       this.radius=radius;
8:  }
9:
10: public double calculateArea()
11: {
12:          return Math.PI*radius*radius;
13: }
14:
15: public double calculatePerimeter()
16: {
17:          return 2 * Math.PI * radius;
18: }
19:
20: public String getPerimeterDescription()
21: {
22:     return "Perimeter of circle = " +
    calculatePerimeter();
23: }
24:
25: public String getAreaDescription()
26: {
27:     return "Area of circle = " + calculateArea();
28: }
29:}
```

فيما يلي كود الفصيلة Area:

```
1:  import javax.swing.*;
2:  import java.awt.*;
3:  import java.awt.event.*;
4:
5:  public class Area extends JFrame
6:  {
7:   JLabel
    squarelength,circleradius,rectanglelength,rectanglewidt
    h;
8:   JTextField
    squarelengthtext,circleradiustext,rectanglelengthtext,
9:   rectanglewidthtext;
10:  JButton
    squarearea,squareperimeter,circlearea,circleperimeter,re
    ctanglearea,
11:  rectangleperimeter;
12:  GridBagLayout grid;
13:  GridBagConstraints gbc;
14:
15:  Area()
16:  {
17:       addWindowListener(new WindowAdapter()
18:            {
19:                 public void windowClosing(WindowEvent
    we)
20:                 {
21:                      dispose();
22:                      System.exit(0);
23:                 }
24:            }
25:       );
26:
27:       Container c = getContentPane();
```

```
28:
29:       grid = new GridBagLayout();
30:       gbc = new GridBagConstraints();
31:       c.setLayout(grid);
32:
33:       squarelength = new JLabel("square length");
34:       circleradius = new JLabel("circle radius");
35:       rectanglelength = new JLabel("rectangle length");
36:       rectanglewidth = new JLabel("rectangle width");
37:
38:       squarelengthtext = new JTextField(10);
39:       circleradiustext = new JTextField(10);
40:       rectanglelengthtext = new JTextField(10);
41:       rectanglewidthtext = new JTextField(10);
42:
43:       ButtonListener b = new ButtonListener();
44:
45:       squareperimeter = new JButton("square
   perimeter");
46:       squareperimeter.addActionListener(b);
47:
48:       squarearea = new JButton("square area");
49:       squarearea.addActionListener(b);
50:
51:       rectangleperimeter = new JButton("rectangle
   perimeter");
52:       rectangleperimeter.addActionListener(b);
53:
54:       rectanglearea = new JButton("rectangle area");
55:       rectanglearea.addActionListener(b);
56:
57:       circleperimeter = new JButton("circle perimeter");
58:       circleperimeter.addActionListener(b);
```

```
59:
60:         circlearea = new JButton("circle area");
61:         circlearea.addActionListener(b);
62:
63:         gbc.gridx = 1;
64:         gbc.gridy = 1;
65:         grid.setConstraints(squarelength,gbc);
66:         c.add(squarelength);
67:
68:         gbc.gridx = 2;
69:         gbc.gridy = 1;
70:         grid.setConstraints(squarelengthtext,gbc);
71:         c.add(squarelengthtext);
72:
73:         gbc.gridx = 1;
74:         gbc.gridy = 2;
75:         grid.setConstraints(squarearea,gbc);
76:         c.add(squarearea);
77:
78:         gbc.gridx = 2;
79:         gbc.gridy = 2;
80:         grid.setConstraints(squareperimeter,gbc);
81:         c.add(squareperimeter);
82:
83:         gbc.gridx = 1;
84:         gbc.gridy = 4;
85:         grid.setConstraints(circleradius,gbc);
86:         c.add(circleradius);
87:
88:         gbc.gridx = 2;
89:         gbc.gridy = 4;
90:         grid.setConstraints(circleradiustext,gbc);
91:         c.add(circleradiustext);
92:
```

```
93:     gbc.gridx = 1;
94:     gbc.gridy = 5;
95:     grid.setConstraints(circlearea,gbc);
96:     c.add(circlearea);
97:
98:     gbc.gridx = 2;
99:     gbc.gridy = 5;
100:        grid.setConstraints(circleperimeter,gbc);
101:        c.add(circleperimeter );
102:
103:        gbc.gridx = 1;
104:        gbc.gridy = 6;
105:        grid.setConstraints(rectanglelength,gbc);
106:        c.add(rectanglelength);
107:
108:        gbc.gridx = 2;
109:        gbc.gridy = 6;
110:        grid.setConstraints(rectanglelengthtext,gbc);
111:        c.add(rectanglelengthtext );
112:
113:        gbc.gridx = 1;
114:        gbc.gridy = 7;
115:        grid.setConstraints(rectanglewidth,gbc);
116:        c.add(rectanglewidth);
117:
118:        gbc.gridx = 2;
119:        gbc.gridy = 7;
120:        grid.setConstraints(rectanglewidthtext,gbc);
121:        c.add(rectanglewidthtext );
122:
123:        gbc.gridx = 1;
124:        gbc.gridy = 8;
125:        grid.setConstraints(rectanglearea,gbc);
126:        c.add(rectanglearea);
```

```
127:
128:            gbc.gridx = 2;
129:            gbc.gridy = 8;
130:            grid.setConstraints(rectangleperimeter,gbc);
131:            c.add(rectangleperimeter );
132:
133:            setTitle("AREA");
134:            pack();
135:            setVisible(true);
136:        }
137:
138:        public static void main(String args[])
139:        {
140:            Area area = new Area();
141:        }
142:
143:        class ButtonListener implements ActionListener
144:        {
145:            public void actionPerformed(ActionEvent e)
146:            {
147:                Object obj = e.getSource();
148:
149:                if ( obj == squarearea )
150:                {
151:                    double ll = 0;
152:
153:                    try
154:                    {
155:                        ll                                    =
    Double.parseDouble(squarelengthtext.getText());
156:                    }
157:
158:                    catch(Exception ex)
159:                    {
```


obeikan.com

```
160:
    JOptionPane.showMessageDialog(Area.this,"invalid
  value");
161:                    squarelengthtext.selectAll();
162:                    squarelengthtext.requestFocus();
163:                    return;
164:                }
165:
166:                Square square = new Square(ll);
167:
168:
    JOptionPane.showMessageDialog(Area.this,square.get
  AreaDescription());
169:                squarelengthtext.selectAll();
170:                squarelengthtext.requestFocus();
171:            }
172:
173:            else if (obj == squareperimeter )
174:            {
175:                double ll = 0;
176:
177:                try
178:                {
179:                    ll                              =
  Double.parseDouble(squarelengthtext.getText());
180:                }
181:
182:                catch(Exception ex)
183:                {
184:
    JOptionPane.showMessageDialog(Area.this,"invalid
  value");
185:                    squarelengthtext.selectAll();
186:                    squarelengthtext.requestFocus();
```

```
187:                    return;
188:                }
189:
190:                Square square = new Square(ll);
191:
192:
    JOptionPane.showMessageDialog(Area.this,square.get
  PerimeterDescription());
193:                squarelengthtext.selectAll();
194:                squarelengthtext.requestFocus();
195:            }
196:
197:            else if ( obj == circlearea )
198:            {
199:                double rr = 0;
200:
201:                try
202:                {
203:                    rr                              =
  Double.parseDouble(circleradiustext.getText());
204:                }
205:
206:                catch(Exception ex)
207:                {
208:
    JOptionPane.showMessageDialog(Area.this,"invalid
  value");
209:                    circleradiustext.selectAll();
210:                    circleradiustext.requestFocus();
211:                    return;
212:                }
213:
214:                Circle circle = new Circle(rr);
215:
```

```
216:
     JOptionPane.showMessageDialog(Area.this,circle.get
    AreaDescription());
217:                 circleradiustext.selectAll();
218:                 circleradiustext.requestFocus();
219:             }
220:
221:             else if ( obj == circleperimeter )
222:             {
223:                 double rr = 0;
224:
225:                 try
226:                 {
227:                     rr                                    =
    Double.parseDouble(circleradiustext.getText());
228:                 }
229:
230:                 catch(Exception ex)
231:                 {
232:
     JOptionPane.showMessageDialog(Area.this,"invalid
    value");
233:                     circleradiustext.selectAll();
234:                     circleradiustext.requestFocus();
235:                     return;
236:                 }
237:
238:                 Circle circle = new Circle(rr);
239:
240:
     JOptionPane.showMessageDialog(Area.this,circle.get
    PerimeterDescription());
241:                 circleradiustext.selectAll();
242:                 circleradiustext.requestFocus();
```

```
243:            }
244:
245:            else if ( obj == rectanglearea )
246:            {
247:                double rr = 0;
248:
249:                try
250:                {
251:                    rr                                    =
    Double.parseDouble(rectanglelengthtext.getText());
252:                }
253:
254:                catch(Exception ex)
255:                {
256:
     JOptionPane.showMessageDialog(Area.this,"invalid
    value");
257:                    rectanglelengthtext.selectAll();
258:                    rectanglelengthtext.requestFocus();
259:                    return;
260:                }
261:
262:                double ww = 0;
263:
264:                try
265:                {
266:                    ww                                    =
    Double.parseDouble(rectanglewidthtext.getText());
267:                }
268:
269:                catch(Exception ex)
270:                {
271:
     JOptionPane.showMessageDialog(Area.this,"invalid
```

```
    value");
272:                    rectanglewidthtext.selectAll();
273:                    rectanglewidthtext.requestFocus();
274:                    return;
275:                }
276:
277:                Rectangle rectangle = new
    Rectangle(rr,ww);
278:
279:
     JOptionPane.showMessageDialog(Area.this,rectangle.
    getAreaDescription());
280:                rectanglelengthtext.selectAll();
281:                rectanglelengthtext.requestFocus();
282:            }
283:
284:            else if ( obj == rectangleperimeter )
285:            {
286:                double rr = 0;
287:
288:                try
289:                {
290:                    rr                              =
    Double.parseDouble(rectanglelengthtext.getText());
291:                }
292:
293:                catch(Exception ex)
294:                {
295:
     JOptionPane.showMessageDialog(Area.this,"invalid
    value");
296:                    rectanglelengthtext.selectAll();
297:                    rectanglelengthtext.requestFocus();
298:                    return;
```

```
299:                    }
300:
301:                    double ww = 0;
302:
303:                    try
304:                    {
305:                        ww                                    =
    Double.parseDouble(rectanglewidthtext.getText());
306:                    }
307:
308:                    catch(Exception ex)
309:                    {
310:
      JOptionPane.showMessageDialog(Area.this,"invalid
    value");
311:                        rectanglewidthtext.selectAll();
312:                        rectanglewidthtext.requestFocus();
313:                        return;
314:                    }
315:
316:                    Rectangle    rectangle    =    new
    Rectangle(rr,ww);
317:
318:
      JOptionPane.showMessageDialog(Area.this,rectangle.
    getPerimeterDescription());
319:                    rectanglelengthtext.selectAll();
320:                    rectanglelengthtext.requestFocus();
321:                }
322:            }
323:        }
324:}
```

مثال (3): حساب المحيط Perimeter والمساحة Area باستخدام الـ interface:

أولاً: هدف المثال:

نريد تعديل المثال السابق بحيث يتم استخدام الـ interface في تنفيذ البرنامج. لاحظ أن الوظيفة الأساسية للبرنامج لن تتغير.

ثانياً: خطوات الحل:

- نحتاج في هذا المثال إلي interface اسمه MyShape بالإضافة إلي أربع فصائل classes حيث نريد إنشاء فصيلة Square و Rectangle و Circle بالإضافة إلي الفصيلة Area المسئولة عن إنشاء نافذة البرنامج.
- الـ interface المسمي MyShape يحتوي علي الدوال methods المشتركة التالية:
- الدالة calculateArea() والدالة calculatePerimeter() والدالة getPerimeterDescription() والدالة getAreaDescription().
- الفصائل Square و Rectangle و Circle ستقوم بتنفيذ implement الـ interface المسمي Shape وهذا هو التعديل الوحيد فيها.
- سيتم أيضاً تعديل الفصيلة Area بحيث تستخدم الـ interface المسمي Shape في تنفيذ البرنامج.

ثالثاً: كود البرمجة:

يحتوي هذا البرنامج علي interface اسمه MyShape وعلي ثلاثة فصائل classes هم Square و Rectangle و Circle بالإضافة إلي فصيلة class أخري لنافذة البرنامج اسمها Area.

فيما يلي كود الـ interface المسمي MyShape:

```
1:  public interface MyShape
2:  {
3:    double calculateArea();
4:    double calculatePerimeter();
```

```
5:   String getPerimeterDescription();
6:   String getAreaDescription();
7: }
```

فيما يلي كود الفصيلة Square:

```
1: public class Square implements MyShape
2: {
3:   private double length;
4:
5:   Square(double length)
6:   {
7:       this.length=length;
8:   }
9:
10: public double calculateArea()
11: {
12:      return length * length;
13: }
14:
15: public double calculatePerimeter()
16: {
17:      return length * 4;
18: }
19:
20: public String getPerimeterDescription()
21: {
22:      return "Perimeter of square = " +
    calculatePerimeter();
23: }
24:
25: public String getAreaDescription()
26: {
27:      return "Area of square = " + calculateArea();
28: }
29:}
```

obeikan.com

فيما يلي كود الفصيلة Rectangle:

```
1:  public class Rectangle implements MyShape
2:  {
3:    private double length,width;
4:
5:   Rectangle(double length,double width)
6:   {
7:        this.length=length;
8:        this.width=width;
9:   }
10:
11:  public double calculateArea()
12:  {
13:       return length * width;
14:  }
15:
16:  public double calculatePerimeter()
17:  {
18:       return 2 * ( length + width );
19:  }
20:
21:  public String getPerimeterDescription()
22:  {
23:       return "Perimeter of rectangle = " +
     calculatePerimeter();
24:  }
25:
26:  public String getAreaDescription()
27:  {
28:       return "Area of rectangle = " + calculateArea();
29:  }
30:}
```

فيما يلي كود الفصيلة Circle:

```
1:  public class Circle implements MyShape
2:  {
3:   double radius;
4:
5:   Circle(double radius)
6:   {
7:        this.radius=radius;
8:   }
9:
10:  public double calculateArea()
11:  {
12:           return Math.PI*radius*radius;
13:  }
14:
15:  public double calculatePerimeter()
16:  {
17:           return 2 * Math.PI * radius;
18:  }
19:
20:  public String getPerimeterDescription()
21:  {
22:       return "Perimeter of circle = " +
     calculatePerimeter();
23:  }
24:
25:  public String getAreaDescription()
26:  {
27:       return "Area of circle = " + calculateArea();
28:  }
29:}
```

obeikan.com

فيما يلي كود الفصيلة Area:

```
1:  import javax.swing.*;
2:  import java.awt.*;
3:  import java.awt.event.*;
4:
5:  public class Area extends JFrame
6:  {
7:   JLabel
    squarelength,circleradius,rectanglelength,rectanglewidt
    h;
8:   JTextField
    squarelengthtext,circleradiustext,rectanglelengthtext,
9:   rectanglewidthtext;
10:  JButton
    squarearea,squareperimeter,circlearea,circleperimeter,re
    ctanglearea,
11:  rectangleperimeter;
12:  GridBagLayout grid;
13:  GridBagConstraints gbc;
14:
15:  Area()
16:  {
17:      addWindowListener(new WindowAdapter()
18:          {
19:              public void windowClosing(WindowEvent
    we)
20:              {
21:                  dispose();
22:                  System.exit(0);
23:              }
24:          }
25:      );
26:
27:      Container c = getContentPane();
28:
```

```
29:       grid = new GridBagLayout();
30:       gbc = new GridBagConstraints();
31:       c.setLayout(grid);
32:
33:       squarelength = new JLabel("square length");
34:       circleradius = new JLabel("circle radius");
35:       rectanglelength = new JLabel("rectangle length");
36:       rectanglewidth = new JLabel("rectangle width");
37:
38:       squarelengthtext = new JTextField(10);
39:       circleradiustext = new JTextField(10);
40:       rectanglelengthtext = new JTextField(10);
41:       rectanglewidthtext = new JTextField(10);
42:
43:       ButtonListener b = new ButtonListener();
44:
45:       squareperimeter = new JButton("square
    perimeter");
46:       squareperimeter.addActionListener(b);
47:
48:       squarearea = new JButton("square area");
49:       squarearea.addActionListener(b);
50:
51:       rectangleperimeter = new JButton("rectangle
    perimeter");
52:       rectangleperimeter.addActionListener(b);
53:
54:       rectanglearea = new JButton("rectangle area");
55:       rectanglearea.addActionListener(b);
56:
57:       circleperimeter = new JButton("circle perimeter");
58:       circleperimeter.addActionListener(b);
59:
60:       circlearea = new JButton("circle area");
61:       circlearea.addActionListener(b);
```

```
62:
63:         gbc.gridx = 1;
64:         gbc.gridy = 1;
65:         grid.setConstraints(squarelength,gbc);
66:         c.add(squarelength);
67:
68:         gbc.gridx = 2;
69:         gbc.gridy = 1;
70:         grid.setConstraints(squarelengthtext,gbc);
71:         c.add(squarelengthtext);
72:
73:         gbc.gridx = 1;
74:         gbc.gridy = 2;
75:         grid.setConstraints(squarearea,gbc);
76:         c.add(squarearea);
77:
78:         gbc.gridx = 2;
79:         gbc.gridy = 2;
80:         grid.setConstraints(squareperimeter,gbc);
81:         c.add(squareperimeter);
82:
83:         gbc.gridx = 1;
84:         gbc.gridy = 4;
85:         grid.setConstraints(circleradius,gbc);
86:         c.add(circleradius);
87:
88:         gbc.gridx = 2;
89:         gbc.gridy = 4;
90:         grid.setConstraints(circleradiustext,gbc);
91:         c.add(circleradiustext);
92:
93:         gbc.gridx = 1;
94:         gbc.gridy = 5;
95:         grid.setConstraints(circlearea,gbc);
96:         c.add(circlearea);
```

```
97:
98:       gbc.gridx = 2;
99:       gbc.gridy = 5;
100:          grid.setConstraints(circleperimeter,gbc);
101:          c.add(circleperimeter );
102:
103:          gbc.gridx = 1;
104:          gbc.gridy = 6;
105:          grid.setConstraints(rectanglelength,gbc);
106:          c.add(rectanglelength);
107:
108:          gbc.gridx = 2;
109:          gbc.gridy = 6;
110:          grid.setConstraints(rectanglelengthtext,gbc);
111:          c.add(rectanglelengthtext );
112:
113:          gbc.gridx = 1;
114:          gbc.gridy = 7;
115:          grid.setConstraints(rectanglewidth,gbc);
116:          c.add(rectanglewidth);
117:
118:          gbc.gridx = 2;
119:          gbc.gridy = 7;
120:          grid.setConstraints(rectanglewidthtext,gbc);
121:          c.add(rectanglewidthtext );
122:
123:          gbc.gridx = 1;
124:          gbc.gridy = 8;
125:          grid.setConstraints(rectanglearea,gbc);
126:          c.add(rectanglearea);
127:
128:          gbc.gridx = 2;
129:          gbc.gridy = 8;
130:          grid.setConstraints(rectangleperimeter,gbc);
131:          c.add(rectangleperimeter );
```

```
132:
133:        setTitle("AREA");
134:        pack();
135:        setVisible(true);
136:    }
137:
138:    public static void main(String args[])
139:    {
140:        Area area = new Area();
141:    }
142:
143:    class ButtonListener implements ActionListener
144:    {
145:        public void actionPerformed(ActionEvent e)
146:        {
147:            Object obj = e.getSource();
148:
149:            if ( obj == squarearea )
150:            {
151:                double ll = 0;
152:
153:                try
154:                {
155:                    ll                              =
  Double.parseDouble(squarelengthtext.getText());
156:                }
157:
158:                catch(Exception ex)
159:                {
160:
   JOptionPane.showMessageDialog(Area.this,"invalid
  value");
161:                    squarelengthtext.selectAll();
162:                    squarelengthtext.requestFocus();
163:                    return;
164:                }
```

```
165:
166:                    MyShape shape = new Square(ll);
167:
168:
    JOptionPane.showMessageDialog(Area.this,shape.getA
    reaDescription());
169:                    squarelengthtext.selectAll();
170:                    squarelengthtext.requestFocus();
171:                }
172:
173:                else if (obj == squareperimeter )
174:                {
175:                    double ll = 0;
176:
177:                    try
178:                    {
179:                        ll                                    =
    Double.parseDouble(squarelengthtext.getText());
180:                    }
181:
182:                    catch(Exception ex)
183:                    {
184:
    JOptionPane.showMessageDialog(Area.this,"invalid
    value");
185:                        squarelengthtext.selectAll();
186:                        squarelengthtext.requestFocus();
187:                        return;
188:                    }
189:
190:                    MyShape shape = new Square(ll);
191:
192:
    JOptionPane.showMessageDialog(Area.this,shape.getP
    erimeterDescription());
```

```
193:                squarelengthtext.selectAll();
194:                squarelengthtext.requestFocus();
195:            }
196:
197:            else if ( obj == circlearea )
198:            {
199:                double rr = 0;
200:
201:                try
202:                {
203:                    rr                                    =
   Double.parseDouble(circleradiustext.getText());
204:                }
205:
206:                catch(Exception ex)
207:                {
208:
    JOptionPane.showMessageDialog(Area.this,"invalid
   value");
209:                    circleradiustext.selectAll();
210:                    circleradiustext.requestFocus();
211:                    return;
212:                }
213:
214:                MyShape shape = new Circle(rr);
215:
216:
    JOptionPane.showMessageDialog(Area.this,shape.get
   AreaDescription());
217:                circleradiustext.selectAll();
218:                circleradiustext.requestFocus();
219:            }
220:
221:            else if ( obj == circleperimeter )
222:            {
223:                double rr = 0;
```

```
224:
225:                try
226:                {
227:                    rr                                    =
    Double.parseDouble(circleradiustext.getText());
228:                }
229:
230:                catch(Exception ex)
231:                {
232:
    JOptionPane.showMessageDialog(Area.this,"invalid
    value");
233:                    circleradiustext.selectAll();
234:                    circleradiustext.requestFocus();
235:                    return;
236:                }
237:
238:                MyShape shape = new Circle(rr);
239:
240:
    JOptionPane.showMessageDialog(Area.this,shape.getP
    erimeterDescription());
241:                circleradiustext.selectAll();
242:                circleradiustext.requestFocus();
243:            }
244:
245:            else if ( obj == rectanglearea )
246:            {
247:                double rr = 0;
248:
249:                try
250:                {
251:                    rr                                    =
    Double.parseDouble(rectanglelengthtext.getText());
252:                }
```

```
253:
254:                    catch(Exception ex)
255:                    {
256:
     JOptionPane.showMessageDialog(Area.this,"invalid
   value");
257:                        rectanglelengthtext.selectAll();
258:                        rectanglelengthtext.requestFocus();
259:                        return;
260:                    }
261:
262:                    double ww = 0;
263:
264:                    try
265:                    {
266:                        ww                              =
   Double.parseDouble(rectanglewidthtext.getText());
267:                    }
268:
269:                    catch(Exception ex)
270:                    {
271:
     JOptionPane.showMessageDialog(Area.this,"invalid
   value");
272:                        rectanglewidthtext.selectAll();
273:                        rectanglewidthtext.requestFocus();
274:                        return;
275:                    }
276:
277:                    MyShape        shape        =        new
   Rectangle(rr,ww);
278:
279:
     JOptionPane.showMessageDialog(Area.this,shape.get
   AreaDescription());
280:                    rectanglelengthtext.selectAll();
```

```
281:                rectanglelengthtext.requestFocus();
282:            }
283:
284:            else if ( obj == rectangleperimeter )
285:            {
286:                double rr = 0;
287:
288:                try
289:                {
290:                    rr                                    =
    Double.parseDouble(rectanglelengthtext.getText());
291:                }
292:
293:                catch(Exception ex)
294:                {
295:
     JOptionPane.showMessageDialog(Area.this,"invalid
    value");
296:                    rectanglelengthtext.selectAll();
297:                    rectanglelengthtext.requestFocus();
298:                    return;
299:                }
300:
301:                double ww = 0;
302:
303:                try
304:                {
305:                    ww                                    =
    Double.parseDouble(rectanglewidthtext.getText());
306:                }
307:
308:                catch(Exception ex)
309:                {
310:JOptionPane.showMessageDialog(Area.this,"invalid
    value");
311:rectanglewidthtext.selectAll();
```

```
312:rectanglewidthtext.requestFocus();
313:                    return;
314:                }
315:
316:                MyShape shape = new
    Rectangle(rr,ww);
317:
318:JOptionPane.showMessageDialog(Area.this,shape.get
    PerimeterDescription());
319:rectanglelengthtext.selectAll();
320:rectanglelengthtext.requestFocus();
321:            }
322:        }
323:    }
324:}
```

ملخص الفصل:

- تعلمنا في هذا الفصل العديد من تطبيقات البرمجة موجهة الهدف Object-Oriented Programming (OOP).
- في الفصل القادم نتعلم - بإذن الله - العديد من التطبيقات الأخري ، فتابع معنا الفصل القادم.