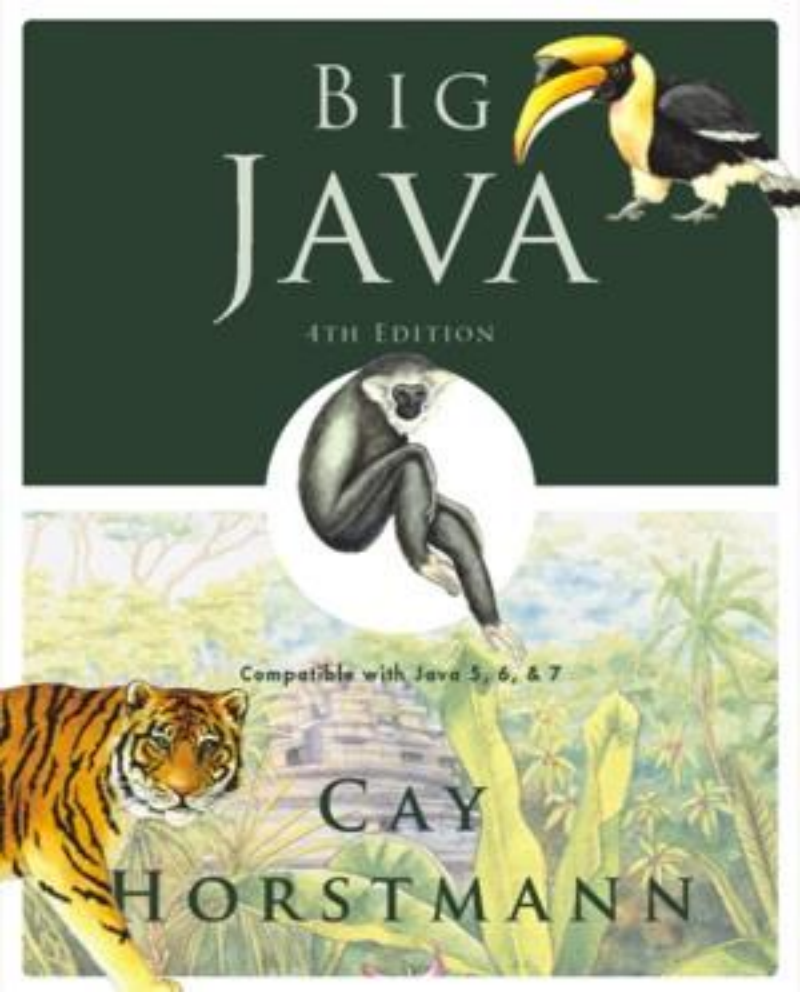


الفصل ٩ – الواجهات وتعدد الأشكال

Chapter 9 – Interfaces and Polymorphism

Chapter Goals

- To be able to declare and use interface types
- To understand the concept of polymorphism
- To appreciate how interfaces can be used to decouple classes
- To learn how to implement helper classes as inner classes
- To implement event listeners in graphical applications

- لتكون قادرة على إعلان واستخدام أنواع واجهة
- لفهم مفهوم تعدد الأشكال
- لنقدر كيف يمكن استخدام واجهات لفصل الطبقات
- لمعرفة كيفية تنفيذ الطبقات المساعد كفئات الداخلية
- لتنفيذ المستمعين الحدث في التطبيقات الرسومية

Using Interfaces for Algorithm Reuse

- Use *interface types* to make code more reusable
- In Chapter 6, we created a `DataSet` to find the average and maximum of a set of *numbers*
- What if we want to find the average and maximum of a set of `BankAccount` values?

- استخدام أنواع واجهة لجعل رمز أكثر قابلة لإعادة الاستخدام
- في الفصل ٦، أنشأنا قاعدة بيانات لإيجاد المتوسط والحد الأقصى من مجموعة من الأرقام
- ما إذا كنا نريد للعثور على المتوسط والحد الأقصى من مجموعة من القيم BankAccount؟

Using Interfaces for Algorithm Reuse

```
public class DataSet // Modified for BankAccount objects {
   private double sum;
   private BankAccount maximum;
   private int count;
   ...
   public void add(BankAccount x)
   {
      sum = sum + x.getBalance();
      if (count == 0 || maximum.getBalance() < x.getBalance())
         maximum = x;
      count++;
   }

   public BankAccount getMaximum()
   {
      return maximum;
   }
}
```

Using Interfaces for Algorithm Reuse

Or suppose we wanted to find the coin with the highest value among a set of coins. We would need to modify the `DataSet` class again:

```
public class DataSet // Modified for Coin objects
{
   private double sum;
   private Coin maximum;
   private int count;
   ...
   public void add(Coin x)
   {
      sum = sum + x.getValue();
      if (count == 0 || maximum.getValue() <
         x.getValue()) maximum = x;
      count++;
   }
```

- أو لنفترض أننا نريد العثور على عملة واحدة وفقا لأعلى قيمة بين مجموعة من القطع النقدية. ونحن بحاجة إلى تعديل فئة DataSet مرة أخرى

Using Interfaces for Algorithm Reuse

```
 public Coin getMaximum()
    {
       return maximum;
    }
}
```

Using Interfaces for Algorithm Reuse

- The algorithm for the data analysis service is the same in all cases; details of measurement differ
- Classes could agree on a method `getMeasure` that obtains the measure to be used in the analysis
- We can implement a single reusable `DataSet` class whose `add` method looks like this:

```
sum = sum + x.getMeasure();
if (count == 0 || maximum.getMeasure() < x.getMeasure())
   maximum = x;
count++;
```

- خوارزمية لخدمة تحليل البيانات هي نفسها في جميع الحالات؛ تفاصيل القياس تختلف
- الطبقات يمكن أن توافق على getMeasure الأسلوب الذي يحصل على قياس لاستخدامها في التحليل
- يمكننا تنفيذ واحدة قابلة لإعادة الاستخدام فئة DataSet الذي يبدو مثل هذا الأسلوب إضافة:

Using Interfaces for Algorithm Reuse

- What is the type of the variable `x`?
  - *`x` should refer to any class that has a `getMeasure` method*
- In Java, an **interface type** is used to specify required operations:

```
public interface Measurable
{
   double getMeasure();
}
```

- Interface declaration lists all methods that the interface type requires

- ما هو نوع المتغير x؟
- ينبغي أن تشير إلى x أي فئة لديها طريقة getMeasure
- في الجافا، يتم استخدام نوع واجهة لتحديد العمليات المطلوبة:

يسرد إعلان واجهة كل الأساليب التي يتطلب نوع الواجهة

Syntax 9.1 Declaring an Interface

Syntax

```
public interface InterfaceName
{
   method signatures
}
```

Example

```
public interface Measurable
{
   double getMeasure();
}
```

The methods of an interface are automatically public.

No implementation is provided.

Interfaces vs. Classes

An interface type is similar to a class, but there are several important differences:

- *All methods in an interface type are **abstract**; they don't have an implementation*
- *All methods in an interface type are **automatically public***
- *An interface type does not have **instance fields***

نوع واجهة مشابهة لفئة، ولكن هناك عدة اختلافات هامة:

- جميع الطرق في نوع واجهة هي مجردة. لم يكن لديهم التنفيذ
- جميع الطرق في نوع واجهة علنية تلقائيا
- لا يكون لها نوع واجهة الحقول المثال

Generic `DataSet` for Measurable Objects

```
public class DataSet
{
   private double sum;
   private Measurable maximum;
   private int count;
   ...
   public void add(Measurable x)
   {
      sum = sum + x.getMeasure();
      if (count == 0 || maximum.getMeasure() < x.getMeasure())
         maximum = x;
      count++;
   }

   public Measurable getMaximum()
   {
      return maximum;
   }
}
```

Implementing an Interface Type

- Use `implements` reserved word to indicate that a class implements an interface type:

استخدام ينفذ كلمة محجوزة للإشارة إلى أن فئة تنفذ نوع واجهة

```
public class BankAccount implements Measurable
{
   public double getMeasure()
   {
      ...
      return balance;
   }
}
```

وهناك فئة يمكن تنفيذ أكثر من نوع واحد واجهة
الطبقة يجب أن تعلن عن الأساليب التي مطلوبة من قبل جميع
الواجهات التي تنفذها

- A class can implement more than one interface type
  - *Class must declare all the methods that are required by all the interfaces it implements*

Implementing an Interface Type

- Another example: مثال آخر

```
public class Coin implements Measurable
{
   public double getMeasure()
   {
      return value;
   }
   ...
}
```

Code Reuse

- A service type such as DataSet specifies an interface for participating in the service
- Use interface types to make code **more reusable**

- وهناك نوع الخدمات مثل مجموعة البيانات يحدد واجهة للمشاركة في خدمة
- استخدام أنواع واجهة لجعل رمز أكثر قابلة لإعادة الاستخدام

Figure 1
Attachments Conform to the Mixer's Interface

Syntax 9.2 Implementing an Interface

Syntax

```
public class ClassName implements InterfaceName, InterfaceName, . . .
{
   instance variables
   methods
}
```

Example

```
public class BankAccount implements Measurable
{
   . . .
   public double getMeasure()
   {
      return balance;
   }
   . . .
}
```

List all interface types that this class implements.

BankAccount instance variables

This method provides the implementation for the method declared in the interface.

Other BankAccount methods

UML Diagram of `DataSet` and Related Classes

- Interfaces can reduce the **coupling between classes**
- UML notation:
  - *Interfaces are tagged with a "stereotype" indicator «interface»*
  - *A dotted arrow with a triangular tip denotes the "is-a" relationship between a class and an interface*
  - *A dotted line with an open v-shaped arrow tip denotes the "uses" relationship or dependency*
- Note that `DataSet` is *decoupled* fror `BankAccount` and `Coin`

يمكن أن يقلل من واجهات اقتران بين الطبقات
UML التدوين:

- يتم تمييزها واجهات مع "الصورة النمطية" مؤشر «واجهة»
- والسهم منقط مع طرف الثلاثي يدل على "هو واحد" العلاقة بين فئة واجهة
- خط منقط مع على شكل حرف V مفتوحة السهم طرف يدل على "الاستخدامات" العلاقة أو التبعية

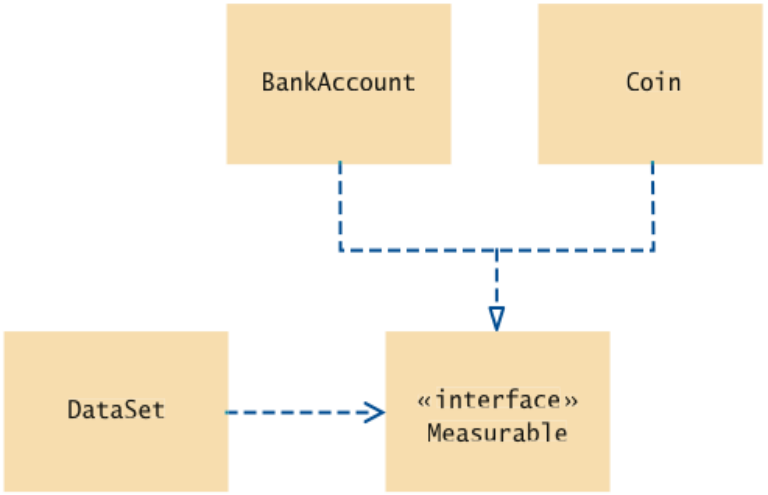


Figure 2 UML Diagram of the `DataSet` Class and the Classes that Implement the `Measurable` Interface

ch09/measure1/DataSetTester.java

```
1  /**
2     This program tests the DataSet class.
3  */
4  public class DataSetTester
5  {
6     public static void main(String[] args)
7     {
8        DataSet bankData = new DataSet();
9
10       bankData.add(new BankAccount(0));
11       bankData.add(new BankAccount(10000));
12       bankData.add(new BankAccount(2000));
13
14       System.out.println("Average balance: " + bankData.getAverage());
15       System.out.println("Expected: 4000");
16       Measurable max = bankData.getMaximum();
17       System.out.println("Highest balance: " + max.getMeasure());
18       System.out.println("Expected: 10000");
19
20       DataSet coinData = new DataSet();
21
```

Continued

ch09/measure1/DataSetTester.java (cont.)

```
22         coinData.add(new Coin(0.25, "quarter"));
23         coinData.add(new Coin(0.1, "dime"));
24         coinData.add(new Coin(0.05, "nickel"));
25
26         System.out.println("Average coin value: " +
coinData.getAverage());
27         System.out.println("Expected: 0.133");
28         max = coinData.getMaximum();
29         System.out.println("Highest coin value: " +
max.getMeasure());
30         System.out.println("Expected: 0.25");
31      }
32   }
```

ch09/measure1/DataSetTester.java (cont.)

Program Run:

```
Average balance: 4000.0
Expected: 4000
Highest balance: 10000.0
Expected: 10000
Average coin value: 0.13333333333333333
Expected: 0.133
Highest coin value: 0.25
Expected: 0.25
```

Self Check 9.1

Suppose you want to use the `DataSet` class to find the `Country` object with the largest population. What condition must the `Country` class fulfill?

Answer: It must implement the `Measurable` interface, and its `getMeasure` method must return the population.

- افترض أنك تريد استخدام فئة DataSet إلى العثور على الكائن البلد مع أكبر عدد من السكان. ما شرط يجب الوفاء الطبقة البلد؟
  الجواب: يجب أن تطبيق واجهة قابلة للقياس، وطريقة getMeasure لها يجب أن يعود السكان.

Self Check 9.2

Why can't the `add` method of the `DataSet` class have a parameter of type `Object`?

Answer: The `Object` class doesn't have a `getMeasure` method, and the `add` method invokes the `getMeasure` method.

- لماذا لا يمكن للأسلوب إضافة فئة DataSet يكون معلمة من نوع الكائن؟
الجواب: ليس لدى فئة الكائن طريقة getMeasure، وطريقة `add` استدعاء أسلوب getMeasure.

Converting Between Class and Interface Types

- You can convert from **a class type to an interface type**, provided
  the class implements the interface

يمكنك تحويل من نوع فئة إلى نوع واجهة، شريطة؟
الطبقة تطبق الواجهة

- `BankAccount account = new BankAccount(10000);`
  `Measurable x = account; // OK`
- `Coin dime = new Coin(0.1, "dime");`
  `Measurable x = dime; // Also OK`

لا يمكن تحويل بين أنواع لا علاقة لها

- **Cannot convert between unrelated types:**

```
Measurable x = new Rectangle(5, 10, 20, 30); // ERROR
```

Because `Rectangle` doesn't implement `Measurable`

Variables of Class and Interface Types

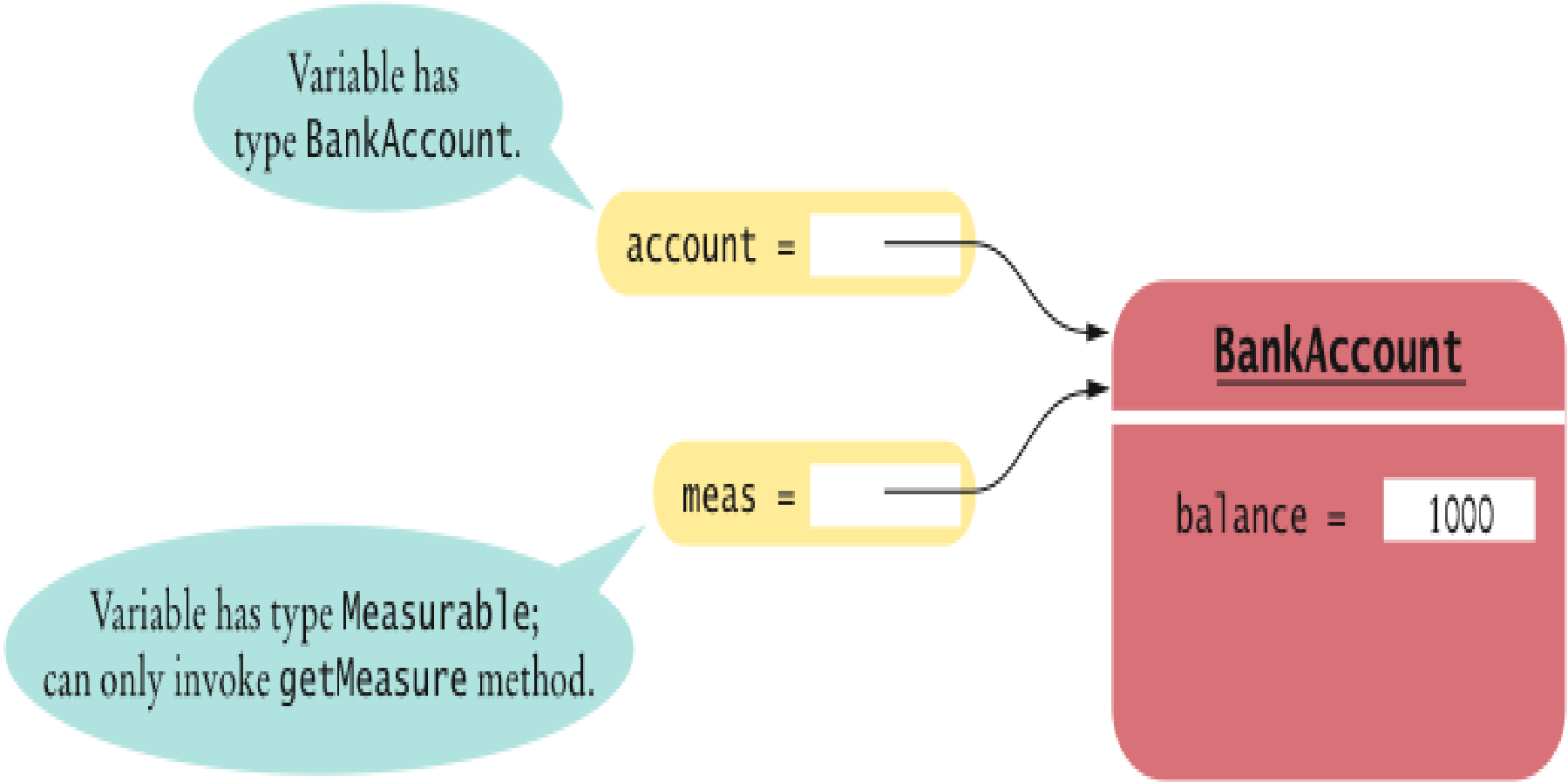


Figure 3 Variables of Class and Interface Types

Casts

- Add `Coin` objects to `DataSet`:

```
DataSet coinData = new DataSet();
coinData.add(new Coin(0.25, "quarter"));
coinData.add(new Coin(0.1, "dime"));
coinData.add(new Coin(0.05, ”nickel"));
Measurable max = coinData.getMaximum(); // Get the largest coin
```

- What can you do with `max`? It's not of type `Coin`:

```
String name = max.getName(); // ERROR
```

- You **need a cast to convert from an interface type to a class type**
- You know it's a `Coin`, but the compiler doesn't. Apply a cast:

```
Coin maxCoin = (Coin) max;
String name = maxCoin.getName();
```

Casts

- If you are wrong and `max` isn't a coin, **the program throws an exception and terminates**
- Difference with casting numbers:
  - *When casting number types you agree to the **information loss***
  - *When casting object types you agree to that **risk of causing an exception***

إذا كنت على خطأ والحد الأقصى ليست عملة، برنامج يطرح استثناء وتنتهي
الفرق مع أرقام الصب:
عندما صب أنواع الرقم الذي وافقت على فقدان المعلومات
عندما صب أنواع الكائنات أنت توافق على أن خطر التسبب استثناء

Self Check 9.3

Can you use a cast `(BankAccount) x` to convert a `Measurable` variable `x` to a `BankAccount` reference?

Answer: Only if `x` actually refers to a `BankAccount` object.

يمكنك استخدام الزهر (BankAccount) العاشر لتحويل المتغير للقياس العاشر إلى مرجع BankAccount؟
الإجابة: فقط إذا كان x الواقع يشير إلى كائن BankAccount.

Self Check 9.4

If both `BankAccount` and `Coin` implement the `Measurable` interface, can a `Coin` reference be converted to a `BankAccount` reference?

Answer: No — a `Coin` reference can be converted to a `Measurable` reference, but if you attempt to cast that reference to a `BankAccount`, an exception occurs.

إذا كان كل من BankAccount وكوين تنفيذ واجهة قابلة للقياس، ويمكن تحويل إشارة كوين إلى مرجع BankAccount؟
الجواب: لا - في إشارة كوين يمكن تحويلها إلى إشارة قابلة للقياس، ولكن إذا حاولت أن يلقي إشارة إلى BankAccount، يحدث استثناء.

Polymorphism

تعدد الأشكال

- An interface variable holds a reference to object of a class that implements the interface:

```
Measurable meas;
meas = new BankAccount(10000);
meas = new Coin(0.1, "dime");
```

متغير واجهة يحمل إشارة إلى الاعتراض من الفئة التي تطبق الواجهة:

Note that the object to which `meas` refers doesn't have type `Measurable`; the type of the object is some class that implements the `Measurable` interface

لاحظ أن الكائن الذي يشير الاتفاقات البيئية المتعددة الأطراف لايوجد نوع قابلة للقياس؛ نوع الكائن بعض الفئة التي تطبق واجهة قابلة للقياس

- You can call any of the interface methods:

```
double m = meas.getMeasure();
```

- Which method is called?

Interface Reference

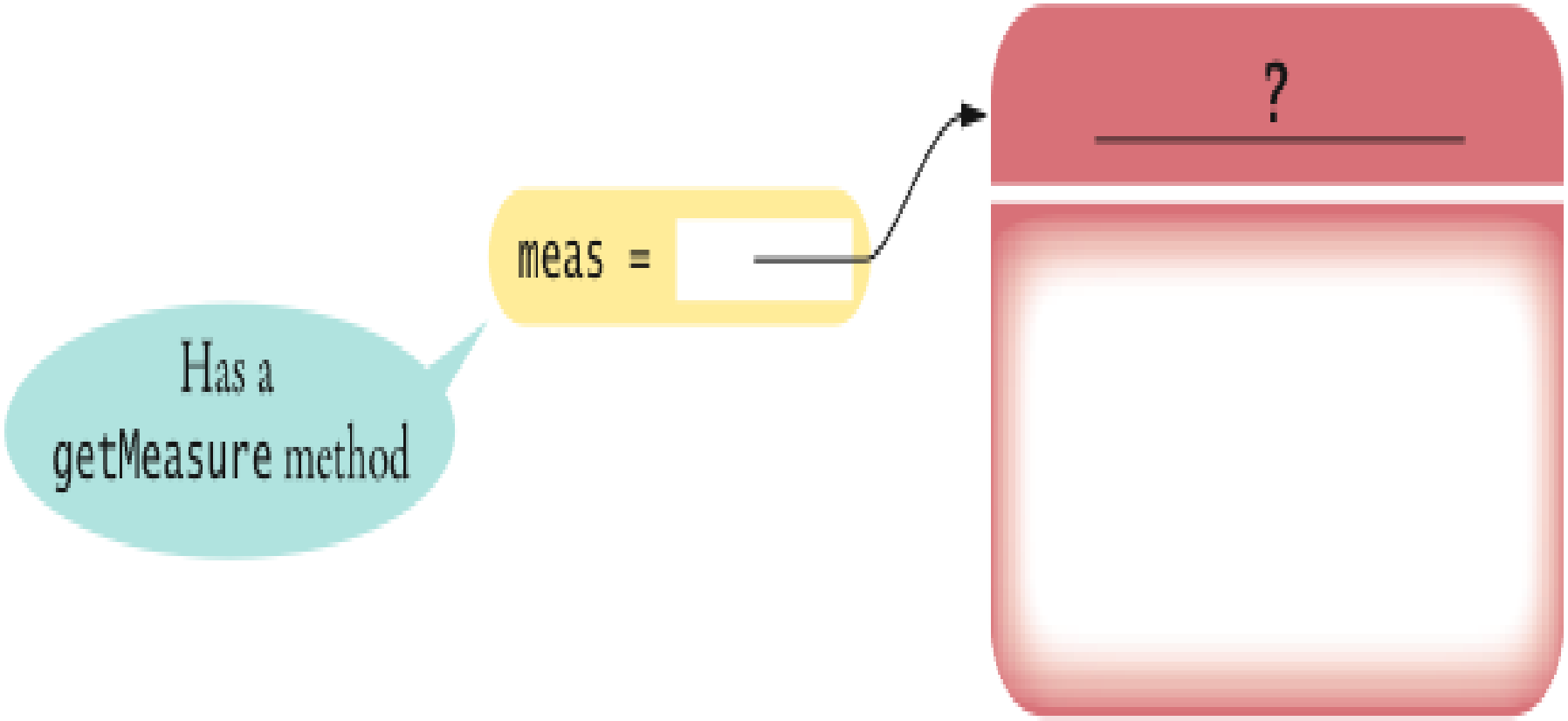


Figure 4 An Interface Reference Can Refer to an Object of Any Class that Implements the Interface

Polymorphism

تعدد الأشكال

- When the virtual machine calls an instance method, it locates the method of the implicit parameter's class — called ***dynamic method lookup***
- If `meas` refers to a `BankAccount` object, then `meas.getMeasure()` calls the `BankAccount.getMeasure` method
- If `meas` refers to a `Coin` object, then method `Coin.getMeasure` is called
- **Polymorphism (many shapes)** denotes the ability to treat objects with differences in behavior in a uniform way

عندما تدعو الجهاز افتراضي أسلوب مثيل، فإنه يقع على أسلوب فئة المعلمة الضمنية ل- استدعاء أسلوب ديناميكي بحث إذا الاتفاقات البيئية المتعددة الأطراف يشير إلى كائن BankAccount، ثم meas.getMeasure () باستدعاء أسلوب BankAccount.getMeasure

إذا الاتفاقات البيئية المتعددة الأطراف يشير إلى كائن كوين، ثم يتم استدعاء أسلوب Coin.getMeasure

تعدد الأشكال (العديد من الأشكال) يدل على القدرة على علاج الكائنات مع الاختلافات في السلوك بطريقة موحدة

Animation 9.1: Polymorphism

Self Check 9.5

Why is it impossible to construct a `Measurable` object?

Answer: `Measurable` is an interface. Interfaces have no fields and no method implementations.

لماذا هو من المستحيل لبناء كائن للقياس؟
الجواب: هو واجهة للقياس. واجهات ليس لها مجالات وتوجد طريقة التنفيذ.

Self Check 9.6

Why can you nevertheless declare a variable whose type is `Measurable`?

Answer: That variable never refers to a `Measurable` object. It refers to an object of some class — a class that implements the `Measurable` interface.

لماذا يمكنك مع ذلك بتعريف متغير نوعه هو للقياس؟
الجواب: هذا المتغير أبدا يشير إلى كائن قابلة للقياس. فإنه يشير إلى كائن من بعض الطبقة - فئة التي تطبق الواجهة قابلة للقياس.

Self Check 9.7

What does this code fragment print? Why is this an example of polymorphism?

ماذا يعني هذا جزء التعليمات البرمجية طباعة؟ لماذا هذا مثال على تعدد الأشكال؟

```
DataSet data = new DataSet();
data.add(new BankAccount(1000));
data.add(new Coin(0.1, "dime"));
System.out.println(data.getAverage());
```

Answer: The code fragment prints 500.05. Each call to `add` results in a call `x.getMeasure()`. In the first call, `x` is a `BankAccount`. In the second call, `x` is a `Coin`. A different `getMeasure` method is called in each case. The first call returns the account balance, the second one the coin value.

جزء التعليمات البرمجية يطبع ٥٠٠.٠٥. كل استدعاء `add` النتائج في x.getMeasure(). في استدعاء الأولى، x هو BankAccount. في استدعاء الثانية، x هو عملة `Coin` . يطلق على طريقة getMeasure مختلفة في كل حالة. استدعاء الأولى بإرجاع رصيد الحساب، ثانية واحدة قيمة العملة.

Using Interfaces for Callbacks

- Limitations of `Measurable` interface:
  - *Can add `Measurable` interface only to classes under your control*
  - *Can measure an object in only one way*
  - *E.g., cannot analyze a set of savings accounts both by bank balance and by interest rate*
- **Callback:** a mechanism for specifying code that is executed at a later time
- In previous `DataSet` implementation, responsibility of measuring lies with the added objects themselves

القيود المفروضة على واجهة قابلة للقياس `Measurable`:
يمكنك إضافة واجهة قابلة للقياس فقط إلى الطبقات تحت سيطرتك
يمكن قياس كائن في اتجاه واحد فقط
على سبيل المثال، لا يمكن تحليل مجموعة من حسابات التوفير سواء عن طريق الرصيد المصرفي وسعر الفائدة
رد: آلية لتحديد التعليمات البرمجية التي يتم تنفيذها في وقت لاحق
في تنفيذ مجموعة البيانات السابقة، مسؤولية تقع على عاتق قياس الكائنات أضاف أنفسهم

Using Interfaces for Callbacks

- Alternative: Hand the object to be measured to a method of an interface:

البديل: من ناحية الكائن الذي يتم قياسه لوسيلة لواجهة

```
public interface Measurer
{
   double measure(Object anObject);
}
```

- `Object` is the “lowest common denominator” of all classes

الكائن هو "القاسم المشترك الأدنى" من جميع الطبقات

Using Interfaces for Callbacks

- The code that makes the call to the callback receives an object of class that implements this interface:

التعليمات البرمجية التي تجعل الدعوة إلى رد الاتصال يتلقى كائن من الفئة التي تطبق هذه الواجهة

```
public DataSet(Measurer aMeasurer)
{
   sum = 0;
   count = 0;
   maximum = null;
   measurer = aMeasurer; // Measurer instance variable
}
```

- The measurer instance variable carries out the measurements:

المتغير المثال كيال ينفذ القياسات

```
public void add(Object x)
{
   sum = sum + measurer.measure(x);
   if (count == 0 || measurer.measure(maximum) < measurer.measure(x))
      maximum = x;
   count++;
}
```

Using Interfaces for Callbacks

- A specific callback is obtained by implementing the Measurer interface:

يتم الحصول على رد محدد من خلال تنفيذ واجهة كيال

```
public class RectangleMeasurer implements Measurer
{
   public double measure(Object anObject)
   {
      Rectangle aRectangle = (Rectangle) anObject;
      double area = aRectangle.getWidth() *
         aRectangle.getHeight();
      return area;
   }
}
```

- Must cast from `Object` to `Rectangle`:

يجب أن يلقي من كائن إلى مستطيل

```
Rectangle aRectangle = (Rectangle) anObject;
```

Using Interfaces for Callbacks

- Pass measurer to data set constructor: تمرير كيال إلى مجموعة البيانات منشئ

```
Measurer m = new RectangleMeasurer();
DataSet data = new DataSet(m);
data.add(new Rectangle(5, 10, 20, 30));
data.add(new Rectangle(10, 20, 30, 40));
...
```

UML Diagram of `Measurer` Interface and Related Classes

Note that the `Rectangle` class is decoupled from the `Measurer` interface

لاحظ أن الطبقة مستطيل وتنفصل عن واجهة كيال

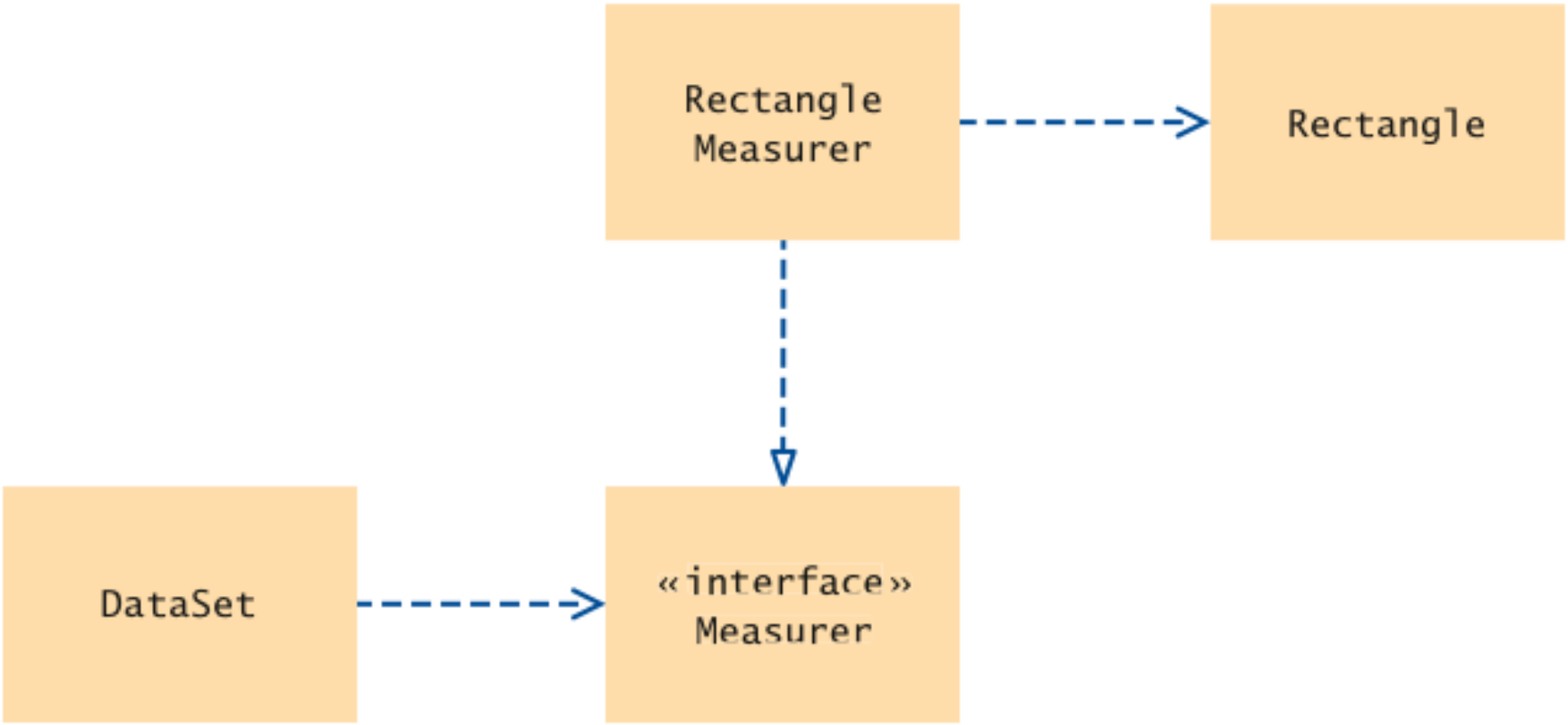


Figure 5 UML Diagram of the DataSet Class and the Measurer Interface

ch09/measure2/Measurer.java

```
1  /**
2     Describes any class whose objects can measure other objects.
3  */
4  public interface Measurer
5  {
6     /**
7        Computes the measure of an object.
8        @param anObject the object to be measured
9        @return the measure
10    */
11    double measure(Object anObject);
12 }
```

ch09/measure2/RectangleMeasurer.java

```
1  import java.awt.Rectangle;
2
3  /**
4     Objects of this class measure rectangles by area.
5  */
6  public class RectangleMeasurer implements Measurer
7  {
8     public double measure(Object anObject)
9     {
10       Rectangle aRectangle = (Rectangle) anObject;
11       double area = aRectangle.getWidth() * aRectangle.getHeight();
12       return area;
13    }
14 }
```

ch09/measure2/DataSet.java

```
1  /**
2     Computes the average of a set of data values.
3  */
4  public class DataSet
5  {
6     private double sum;
7     private Object maximum;
8     private int count;
9     private Measurer measurer;
10
11    /**
12       Constructs an empty data set with a given measurer.
13       @param aMeasurer the measurer that is used to measure data values
14    */
15    public DataSet(Measurer aMeasurer)
16    {
17       sum = 0;
18       count = 0;
19       maximum = null;
20       measurer = aMeasurer;
21    }
22
```

Continued

ch09/measure2/DataSet.java (cont.)

```
23      /**
24         Adds a data value to the data set.
25         @param x a data value
26      */
27      public void add(Object x)
28      {
29         sum = sum + measurer.measure(x);
30         if (count == 0 || measurer.measure(maximum) < measurer.measure(x))
31            maximum = x;
32         count++;
33      }
34
35      /**
36         Gets the average of the added data.
37         @return the average or 0 if no data has been added
38      */
39      public double getAverage()
40      {
41         if (count == 0) return 0;
42         else return sum / count;
43      }
44
```

Continued

ch09/measure2/DataSet.java (cont.)

```
45      /**
46         Gets the largest of the added data.
47         @return the maximum or 0 if no data has been added
48      */
49      public Object getMaximum()
50      {
51         return maximum;
52      }
53   }
```

ch09/measure2/DataSetTester2.java

```
1  import java.awt.Rectangle;
2
3  /**
4     This program demonstrates the use of a Measurer.
5  */
6  public class DataSetTester2
7  {
8     public static void main(String[] args)
9     {
10       Measurer m = new RectangleMeasurer();
11
12       DataSet data = new DataSet(m);
13
14       data.add(new Rectangle(5, 10, 20, 30));
15       data.add(new Rectangle(10, 20, 30, 40));
16       data.add(new Rectangle(20, 30, 5, 15));
17
18       System.out.println("Average area: " + data.getAverage());
19       System.out.println("Expected: 625");
20
```

Continued

ch09/measure2/DataSetTester2.java (cont.)

```
21         Rectangle max = (Rectangle) data.getMaximum();
22         System.out.println("Maximum area rectangle: " + max);
23         System.out.println("Expected: "
24            + "java.awt.Rectangle[x=10,y=20,width=30,height=40]");
25      }
26   }
```

Program Run:

```
Average area: 625
Expected: 625
Maximum area rectangle:java.awt.Rectangle[x=10,y=20,width=30,height=40]
Expected: java.awt.Rectangle[x=10,y=20,width=30,height=40]
```

Self Check 9.8

Suppose you want to use the `DataSet` class of Section 9.1 to find the longest `String` from a set of inputs. Why can't this work?

Answer: The `String` class doesn't implement the `Measurable` interface.

افترض أنك تريد استخدام فئة DataSet القسم ٩.١ للعثور على أطول سلسلة من مجموعة من المدخلات. لماذا لا يمكن لهذا العمل؟
الجواب: فئة سلسلة لا تنفذ واجهة قابلة للقياس.

Self Check 9.9

How can you use the `DataSet` class of this section to find the longest `String` from a set of inputs?

Answer: Implement a class `StringMeasurer` that implements the `Measurer` interface.

كيف يمكنك استخدام فئة DataSet من هذا الباب لايجاد أطول سلسلة من مجموعة من المدخلات؟
الجواب: تنفيذ StringMeasurer الفئة التي تطبق الواجهة كيال.

Self Check 9.10

Why does the `measure` method of the `Measurer` interface have one more parameter than the `getMeasure` method of the `Measurable` interface?

Answer: A measurer measures an object, whereas `getMeasure` measures “itself”, that is, the implicit parameter.

لماذا طريقة قياس واجهة كيال يكون واحد المعلمة أكثر من طريقة getMeasure واجهة قابلة للقياس؟
الجواب: وكيال يقيس كائن، في حين أن التدابير getMeasure "نفسها"، وهذا هو، المعلمة ضمنية.

Inner Classes

- Trivial class can be declared inside a method:

يمكن تعريف الطبقة تافهة داخل طريقة

```
public class DataSetTester3
{
   public static void main(String[] args)
   {
      class RectangleMeasurer implements Measurer
      {
         ...
      }
      Measurer m = new RectangleMeasurer();
      DataSet data = new DataSet(m);
      ...
   }
}
```

Inner Classes

- If inner class is declared inside an enclosing class, but outside its methods, it is available to all methods of enclosing class:

إذا أعلن الطبقة الداخلية داخل الطبقة أرفق، ولكن خارج أساليبها، وهي متوفرة لجميع طرق أرفق الطبقة

```
public class DataSetTester3
{
   class RectangleMeasurer implements Measurer
   {
      . . .
   }

   public static void main(String[] args)
   {
      Measurer m = new RectangleMeasurer();
      DataSet data = new DataSet(m);
      . . .
   }
}
```

Inner Classes

- Compiler turns an inner class into a regular class file:

```
DataSetTester$1$RectangleMeasurer.class
```

مترجم يتحول فئة الداخلية في ملف الدرجة العادية:

ch09/measure3/DataSetTester3.java

```
1  import java.awt.Rectangle;
2
3  /**
4     This program demonstrates the use of an inner class.
5  */
6  public class DataSetTester3
7  {
8     public static void main(String[] args)
9     {
10       class RectangleMeasurer implements Measurer
11       {
12          public double measure(Object anObject)
13          {
14             Rectangle aRectangle = (Rectangle) anObject;
15             double area
16                   = aRectangle.getWidth() * aRectangle.getHeight();
17             return area;
18          }
19       }
20
21       Measurer m = new RectangleMeasurer();
22
23       DataSet data = new DataSet(m);
24
```

Continued

ch09/measure3/DataSetTester3.java (cont.)

```
25          data.add(new Rectangle(5, 10, 20, 30));
26          data.add(new Rectangle(10, 20, 30, 40));
27          data.add(new Rectangle(20, 30, 5, 15));
28
29          System.out.println("Average area: " + data.getAverage());
30          System.out.println("Expected: 625");
31
32          Rectangle max = (Rectangle) data.getMaximum();
33          System.out.println("Maximum area rectangle: " + max);
34          System.out.println("Expected: "
35             + "java.awt.Rectangle[x=10,y=20,width=30,height=40]");
36       }
37    }
```

Self Check 9.11

Why would you use an inner class instead of a regular class?

Answer: Inner classes are convenient for insignificant classes. Also, their methods can access variables and fields from the surrounding scope.

لماذا كنت تستخدم الطبقة الداخلية بدلا من فئة العادية؟
الجواب: الطبقات الداخلية هي مريحة لفئات ضئيلة. أيضا، يمكن أن أساليبهم الوصول المتغيرات والحقول من نطاق المحيطة بها.

Self Check 9.12

How many class files are produced when you compile the `DataSetTester3` program?

Answer: Four: one for the outer class, one for the inner class, and two for the `DataSet` and `Measurer` classes.

كم عدد ملفات فئة يتم إنتاجها عند ترجمة البرنامج DataSetTester٣؟
الجواب: أربعة: واحدة للطبقة الخارجية، واحدة للطبقة الداخلية، واثنين من لDataSet و كيال الطبقات.

Operating Systems

A Graphical Software Environment for the Linux Operating System

Mock Objects

- Want to test a class before the entire program has been completed ترغب في اختبار فئة قبل تم الانتهاء من البرنامج بأكمله
- A **mock object** provides the same services as another object, but in a simplified manner يوفر كائن وهمي نفس الخدمات كائن آخر، ولكن بطريقة مبسطة
- **Example:** a grade book application, `GradingProgram`, manages quiz scores using class `GradeBook` with methods:

```
public void addScore(int studentId, double score)
public double getAverageScore(int studentId)
public void save(String filename)
```

- Want to test `GradingProgram` without having a fully functional `GradeBook` class

Mock Objects

- Declare an interface type with the same methods that the `GradeBook` class provides
  - *Convention: use the letter `I` as a prefix for the interface name:*

```
public interface IGradeBook
{
    void addScore(int studentId, double score);
    double getAverageScore(int studentId);
    void save(String filename);
    . . .
}
```

أعلن نوع واجهة مع نفس الأساليب التي توفر الطبقة دفتر التقديرات الاتفاقية: استخدام حرف Iكما بادئة لاسم واجهة:

- The `GradingProgram` class should *only* use this interface, never the `GradeBook` class which implements this interface

الطبقة GradingProgramيجب أن تستخدم فقط هذه الواجهة، أبدا فئة دفتر التقديرات التي تنفذ هذه الواجهة

Mock Objects

- Meanwhile, provide a simplified mock implementation, restricted to the case of one student and without saving functionality:

وفي الوقت نفسه، توفر تنفيذ وهمية مبسط، يقتصر على حالة طالب واحد ودون توفير وظائف:

```
public class MockGradeBook implements IGradeBook
{
   private ArrayList<Double> scores;
   public void addScore(int studentId, double score)
   {
      // Ignore studentId
      scores.add(score);
   }
   double getAverageScore(int studentId)
   {
      double total = 0;
      for (double x : scores) { total = total + x; }
      return total / scores.size();
   }
   void save(String filename)
   {
      // Do nothing
   }
   . . .
```

Mock Objects

- Now construct an instance of `MockGradeBook` and use it immediately to test the `GradingProgram` class
- When you are ready to test the actual class, simply use a `GradeBook` instance instead
- Don’t erase the mock class — it will still come in handy for regression testing

- الآن بناء مثيل موك الدرجة كتاب واستخدامها على الفور لاختبار الطبقة برنامج الدرجات
- عندما كنت على استعداد لاختبار الطبقة الفعلية، لمجرد استخدام مثيل الدرجة كتاب بدلا من ذلك
- لا محو الطبقة وهمية - سوف لا تزال تأتي في متناول اليدين لاختبار الانحدار

Self Check 9.13

Why is it necessary that the real class and the mock class implement the same interface type?

Answer: You want to implement the `GradingProgram` class in terms of that interface so that it doesn't have to change when you switch between the mock class and the actual class.

- لماذا هل من الضروري أن الطبقة الحقيقية والطبقة وهمية تنفيذ نفس نوع واجهة؟
- الإجابة: أنت تريد أن تنفيذ برنامج الدرجة الدرجات من حيث أن واجهة بحيث لا تضطر إلى تغيير عند التبديل بين الطبقة وهمية والطبقة الفعلية.

Self Check 9.14

Why is the technique of mock objects particularly effective when the `GradeBook` and `GradingProgram` class are developed by two programmers?

Answer: Because the developer of `GradingProgram` doesn't have to wait for the `GradeBook` class to be complete.

لماذا هو تقنية الأجسام وهمية فعالة بشكل خاص عندما يتم وضع دفتر الدرجات والطبقة برنامج الدرجات من قبل اثنين من المبرمجين؟
الجواب: لأنه ليس لدى المطور من برنامج الدرجات إلى الانتظار لفئة دفتر التقديرات ليكون كاملا.

Events, Event Sources, and Event Listeners

- User interface *events* include key presses, mouse moves, button clicks, and so on
- Most programs don't want to be flooded by boring events
- A program can indicate that it only cares about certain specific events

وتشمل الأحداث واجهة المستخدم الضغط على مفتاح، يتحرك الماوس، نقرات الزر، وهلم جرا
معظم البرامج لا يريدون أن غمرت الأحداث مملة
ويمكن للبرنامج أن تبين أنه يهتم أحداث معينة محددة فقط

Events, Event Sources, and Event Listeners

- **Event listener:**
  - *Notified when event happens*
  - *Belongs to a class that is provided by the application programmer*
  - *Its methods describe the actions to be taken when an event occurs*
  - *A program indicates which events it needs to receive by installing event listener objects*

- الحدث المسموع :
- إعلامك عندما يحدث حدث
- ينتمي إلى فئة التي يتم توفيرها من قبل مبرمج التطبيق
- أساليبها تصف الإجراءات الواجب اتخاذها عند حدوث حدث
- ويشير برنامج الأحداث التي يحتاجها لتلقي طريق تثبيت الأجسام المستمع الحدث

- **Event source:**
  - *User interface component that generates a particular event*
  - *Add an event listener object to the appropriate event source*
  - *When an event occurs, the event source notifies all event listeners*

- مصدر الحدث:
- عنصر واجهة المستخدم الذي يقوم بإنشاء حدث معين
- إضافة كائن المستمع الحدث من مصدر الحدث المناسب
- عند حدوث الحدث، مصدر الحدث بإعلام جميع المستمعين الحدث

Events, Event Sources, and Event Listeners

- Example: A program that prints a message whenever a button is clicked:

والبرنامج الذي يطبع رسالة كلما يتم النقر على زر:

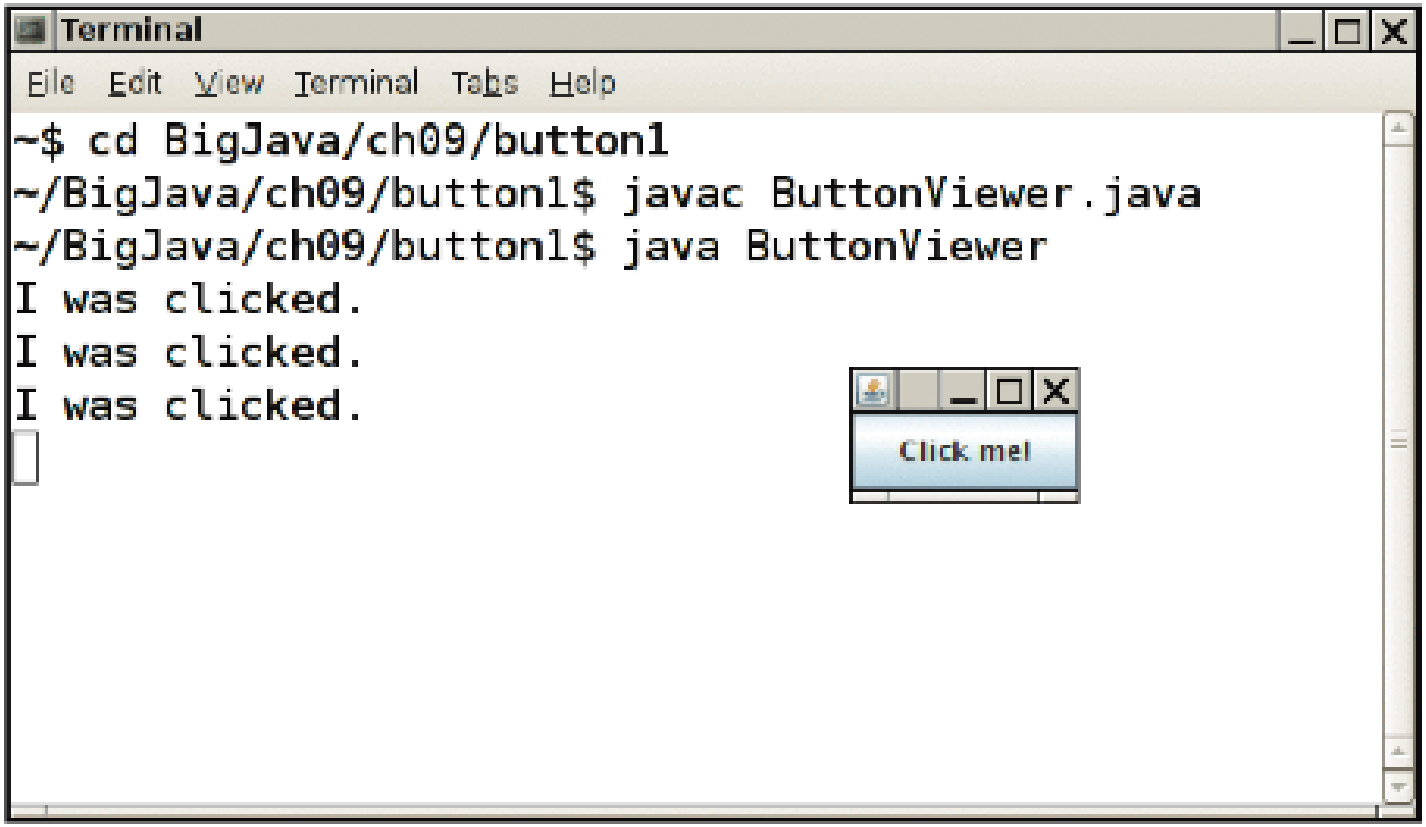


Figure 6 Implementing an Action Listener

Events, Event Sources, and Event Listeners

- Use `JButton` components for buttons; attach an `ActionListener` to each button
- `ActionListener` interface:

```
public interface ActionListener
{
   void actionPerformed(ActionEvent event);
}
```

- Need to supply a class whose `actionPerformed` method contains instructions to be executed when button is clicked
- `event` parameter contains details about the event, such as the time at which it occurred

استخدام مكونات JButton للأزرار. إرفاق ActionListener إلى كل زر

تحتاج إلى توفير فئة الذين طريقة actionPerformed؟ يحتوي على التعليمات الواجب تنفيذها عند النقر على زر
معلمة الحدث تحتوي على تفاصيل حول هذا الحدث، مثل؟ الوقت الذي وقعت

Events, Event Sources, and Event Listeners

- Construct an object of the listener and add it to the button:

```
ActionListener listener = new ClickListener();
button.addActionListener(listener);
```

بناء كائن من المستمع وإضافته إلى الزر:

ActionListener المستمع = ClickListener جديدة ()؛ ؟ button.addActionListener
(المستمع)؛

ch09/button1/ClickListener.java

```
1  import java.awt.event.ActionEvent;
2  import java.awt.event.ActionListener;
3
4  /**
5     An action listener that prints a message.
6  */
7  public class ClickListener implements ActionListener
8  {
9     public void actionPerformed(ActionEvent event)
10    {
11       System.out.println("I was clicked.");
12    }
13 }
```

ch09/button1/ButtonViewer.java

```
1  import java.awt.event.ActionListener;
2  import javax.swing.JButton;
3  import javax.swing.JFrame;
4
5  /**
6     This program demonstrates how to install an action listener.
7  */
8  public class ButtonViewer
9  {
10    private static final int FRAME_WIDTH = 100;
11    private static final int FRAME_HEIGHT = 60;
12
13    public static void main(String[] args)
14    {
15       JFrame frame = new JFrame();
16       JButton button = new JButton("Click me!");
17       frame.add(button);
18
19       ActionListener listener = new ClickListener();
20       button.addActionListener(listener);
21
```

Continued

ch09/button1/ButtonViewer.java (cont.)

```
22          frame.setSize(FRAME_WIDTH, FRAME_HEIGHT);
23          frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
24          frame.setVisible(true);
25       }
26    }
```

Self Check 9.15

Which objects are the event source and the event listener in the `ButtonViewer` program?

Answer: The `button` object is the event source. The `listener` object is the event listener.

أيا من هذه الأجسام مصدر الحدث والمستمع الحدث في برنامج ButtonViewer؟
الإجابة: الكائن زر هو مصدر الحدث. الكائن المستمع هو المستمع الحدث.

Self Check 9.16

Why is it legal to assign a `ClickListener` object to a variable of type `ActionListener`?

Answer: The `ClickListener` class implements the `ActionListener` interface.

لماذا هو القانونية لتعيين كائن ClickListener إلى متغير من نوع ActionListener؟
الإجابة: فئة ClickListener بتنفيذ واجهة ActionListener.

Using Inner Classes for Listeners

- Implement simple listener classes as inner classes like this:

تنفيذ الطبقات المستمع بسيطة مثل الطبقات الداخلية من هذا القبيل

```
JButton button = new JButton("...");
// This inner class is declared in the same method as the
// button variable
class MyListener implements ActionListener
{
   ...
};
ActionListener listener = new MyListener();
button.addActionListener(listener);
```

- This places the trivial listener class exactly where it is needed, without cluttering up the remainder of the project

وهذا ما يضع الطبقة المستمع تافهة بالضبط حيث تمس الحاجة إليها، دون العبث الفترة المتبقية من المشروع

Using Inner Classes for Listeners

- Methods of an inner class can access the variables from the enclosing scope
  - *Local variables that are accessed by an inner class method must be declared as final*
- **Example:** Add interest to a bank account whenever a button is clicked:

يمكن أن أساليب الطبقة الداخلية الوصول إلى المتغيرات من نطاق أرفق
يجب تعريف المتغيرات المحلية التي يتم الوصول إليها عن طريق أسلوب فئة الداخلي على أنه نهائي
مثال: إضافة الفائدة إلى الحساب المصرفي كلما يتم النقر على زر:

Using Inner Classes for Listeners

```
JButton button = new JButton("Add Interest");
final BankAccount account =
   new BankAccount(INITIAL_BALANCE);
// This inner class is declared in the same method as
// the account and button variables.
class AddInterestListener implements ActionListener
{
   public void actionPerformed(ActionEvent event)
   {
      // The listener method accesses the account
      // variable from the surrounding block
      double interest = account.getBalance()
         * INTEREST_RATE / 100;
      account.deposit(interest);
   }
};
ActionListener listener = new AddInterestListener();
button.addActionListener(listener);
```

ch09/button2/InvestmentViewer1.java

```
1  import java.awt.event.ActionEvent;
2  import java.awt.event.ActionListener;
3  import javax.swing.JButton;
4  import javax.swing.JFrame;
5
6  /**
7     This program demonstrates how an action listener can access
8     a variable from a surrounding block.
9  */
10 public class InvestmentViewer1
11 {
12    private static final int FRAME_WIDTH = 120;
13    private static final int FRAME_HEIGHT = 60;
14
15    private static final double INTEREST_RATE = 10;
16    private static final double INITIAL_BALANCE = 1000;
17
18    public static void main(String[] args)
19    {
20       JFrame frame = new JFrame();
21
```

Continued

ch09/button2/InvestmentViewer1.java (cont.)

```
22          // The button to trigger the calculation
23          JButton button = new JButton("Add Interest");
24          frame.add(button);
25
26          // The application adds interest to this bank account
27          final BankAccount account = new BankAccount(INITIAL_BALANCE);
28
29          class AddInterestListener implements ActionListener
30          {
31             public void actionPerformed(ActionEvent event)
32             {
33                // The listener method accesses the account variable
34                // from the surrounding block
35                double interest = account.getBalance() * INTEREST_RATE / 100;
36                account.deposit(interest);
37                System.out.println("balance: " + account.getBalance());
38             }
39          }
40
```

Continued

ch09/button2/InvestmentViewer1.java (cont.)

```
41         ActionListener listener = new AddInterestListener();
42         button.addActionListener(listener);
43
44         frame.setSize(FRAME_WIDTH, FRAME_HEIGHT);
45         frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
46         frame.setVisible(true);
47      }
48   }
```

Program Run:

```
balance: 1100.0
balance: 1210.0
balance: 1331.0
balance: 1464.1
```

Self Check 9.17

Why would an inner class method want to access a variable from a surrounding scope?

Answer: Direct access is simpler than the alternative — passing the variable as a parameter to a constructor or method.

لماذا طريقة الطبقة الداخلية يريدون الوصول إلى متغير من نطاق المحيطة؟
الجواب: الوصول المباشر هو أبسط من البديل - تمرير متغير كمعلمة إلى منشئ أو الأسلوب.

Self Check 9.18

Why would an inner class method want to access a variable from a surrounding If an inner class accesses a local variable from a surrounding scope, what special rule applies?

Answer: The local variable must be declared as `final`.

لماذا طريقة الطبقة الداخلية يريدون الوصول إلى متغير من المحيطة إذا فئة الداخلية يصل متغير محلي من نطاق المحيطة بها، ما حكم خاص ينطبق؟
الإجابة: يجب أن يتم تعريف المتغير المحلي نهائية.

Building Applications with Buttons

- Example: Investment viewer program; whenever button is clicked, interest is added, and new balance is displayed:

برنامج مشاهد للاستثمار، مثال: كلما يتم النقر على زر، يتم إضافة الفائدة، ويتم عرض توازن جديد:

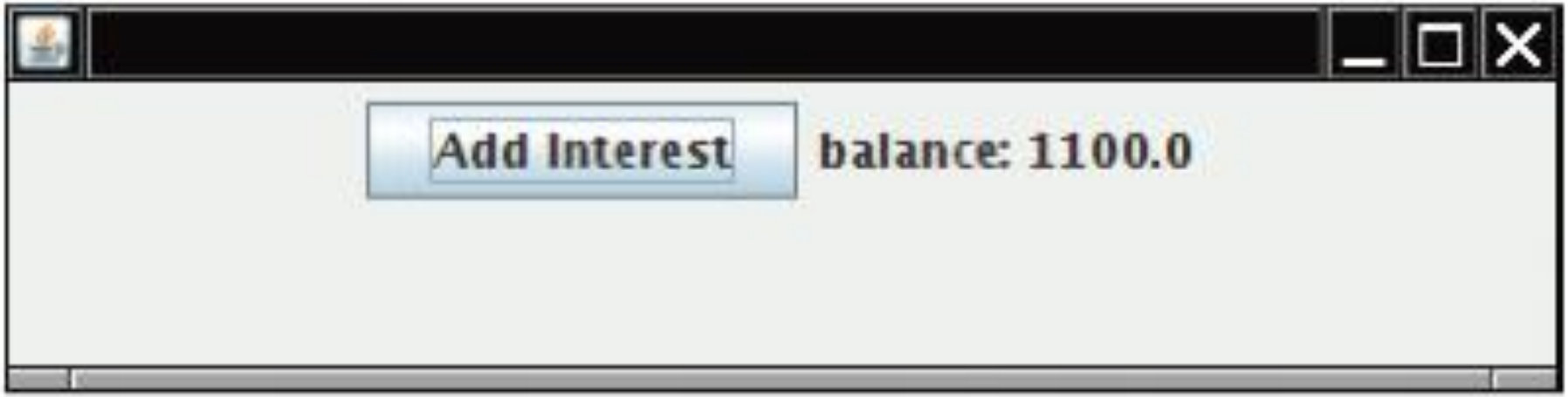


Figure 7 An Application with a Button

Building Applications with Buttons

- Construct an object of the `JButton` class: بناء كائن من الدرجة JButton

```
JButton button = new JButton("Add Interest");
```

- We need a user interface component that displays a message:

نحن بحاجة إلى عنصر واجهة المستخدم التي يعرض رسالة

```
JLabel label = new JLabel("balance: ”
    + account.getBalance());
```

- Use a `JPanel` container to group multiple user interface components together:

- استخدام وعاء JPanel إلى مجموعة متعددة مكونات واجهة المستخدم معا

```
JPanel panel = new JPanel();
panel.add(button);
panel.add(label);
frame.add(panel);
```

Building Applications with Buttons

- Listener class adds interest and displays the new balance:
  - الطبقة المستمع يضيف الفائدة وتعرض توازن جديد

```
class AddInterestListener implements ActionListener
{
   public void actionPerformed(ActionEvent event)
   {
      double interest = account.getBalance() *
         INTEREST_RATE / 100;
      account.deposit(interest);
      label.setText("balance=" + account.getBalance());
   }
}
```

- Add `AddInterestListener` as inner class so it can have access to surrounding `final` variables (`account` and `label`)

  إضافة AddInterest المستمع من الطبقة الداخلية لذلك يمكن أن يكون الوصول إلى المتغيرات المحيطة النهائية (حساب والتسمية)

ch09/button3/InvestmentViewer2.java

```
1  import java.awt.event.ActionEvent;
2  import java.awt.event.ActionListener;
3  import javax.swing.JButton;
4  import javax.swing.JFrame;
5  import javax.swing.JLabel;
6  import javax.swing.JPanel;
7  import javax.swing.JTextField;
8
9  /**
10    This program displays the growth of an investment.
11 */
12 public class InvestmentViewer2
13 {
14    private static final int FRAME_WIDTH = 400;
15    private static final int FRAME_HEIGHT = 100;
16
17    private static final double INTEREST_RATE = 10;
18    private static final double INITIAL_BALANCE = 1000;
19
20    public static void main(String[] args)
21    {
22       JFrame frame = new JFrame();
23
```

Continued

ch09/button3/InvestmentViewer2.java (cont.)

```
24          // The button to trigger the calculation
25          JButton button = new JButton("Add Interest");
26
27          // The application adds interest to this bank account
28          final BankAccount account = new BankAccount(INITIAL_BALANCE);
29
30          // The label for displaying the results
31          final JLabel label = new JLabel("balance: " + account.getBalance());
32
33          // The panel that holds the user interface components
34          JPanel panel = new JPanel();
35          panel.add(button);
36          panel.add(label);
37          frame.add(panel);
38
```

Continued

ch09/button3/InvestmentViewer2.java (cont.)

```
39          class AddInterestListener implements ActionListener
40          {
41             public void actionPerformed(ActionEvent event)
42             {
43                double interest = account.getBalance() * INTEREST_RATE / 100;
44                account.deposit(interest);
45                label.setText("balance: " + account.getBalance());
46             }
47          }
48
49          ActionListener listener = new AddInterestListener();
50          button.addActionListener(listener);
51
52          frame.setSize(FRAME_WIDTH, FRAME_HEIGHT);
53          frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
54          frame.setVisible(true);
55       }
56    }
```

Self Check 9.19

How do you place the `"balance: ..."` message to the left of the `"Add Interest"` button?

Answer: First add `label` to the panel, then add `button`.

كيف يمكنك وضع "التوازن: ..." رسالة إلى يسار الزر "إضافة الفائدة"؟
الإجابة: أولا نضيف علامة إلى لوحة، ثم إضافة زر واحدة.

Self Check 9.20

Why was it not necessary to declare the `button` variable as `final`?

Answer: The `actionPerformed` method does not access that variable.

لماذا كان من غير الضروري أن يعلن على زر متغير نهائية؟
الجواب: طريقة عمل يقوم لا الوصول إلى هذا المتغير.

Processing Timer Events

- `javax.swing.Timer` generates equally spaced timer events, sending events to installed action listeners
- Useful whenever you want to have an object updated in regular intervals

javax.swing.Timer يولد الأحداث الموقت متباعدة بشكل متساو، وإرسال الأحداث إلى المستمعين العمل المثبتة
مفيد كلما كنت تريد أن يكون كائن تحديثها في فترات منتظمة

Processing Timer Events

- Declare a class that implements the `ActionListener` interface:

أعلن فئة التي تطبق الواجهة عمل المستمع

```
class MyListener implements ActionListener
{
   void actionPerformed(ActionEvent event)
   {
        Listener action (executed at each timer event)
   }
}
```

- Add listener to timer and start timer:

إضافة المستمع إلى توقيت وبدء الموقت

```
MyListener listener = new MyListener();
Timer t = new Timer(interval, listener);
t.start();
```

ch09/timer/RectangleComponent.java

Displays a rectangle that can be moved

The `repaint` method causes a component to repaint itself. Call this method whenever you modify the shapes that the `paintComponent` method draws

يعرض مستطيل التي يمكن نقلها
يتسبب أسلوب إعادة رسم repaint مكون إلى إعادة رسم نفسه. استدعاء هذا الأسلوب كلما قمت بتعديل الأشكال التي تلفت طريقة paintComponent

```
 1  import java.awt.Graphics;
 2  import java.awt.Graphics2D;
 3  import java.awt.Rectangle;
 4  import javax.swing.JComponent;
 5
 6  /**
 7     This component displays a rectangle that can be moved.
 8  */
 9  public class RectangleComponent extends JComponent
10  {
11     private static final int BOX_X = 100;
12     private static final int BOX_Y = 100;
13     private static final int BOX_WIDTH = 20;
14     private static final int BOX_HEIGHT = 30;
15
```

Continued

ch09/timer/RectangleComponent.java (cont.)

```
16      private Rectangle box;
17
18      public RectangleComponent()
19      {
20         // The rectangle that the paintComponent method draws
21         box = new Rectangle(BOX_X, BOX_Y, BOX_WIDTH, BOX_HEIGHT);
22      }
23
24      public void paintComponent(Graphics g)
25      {
26         Graphics2D g2 = (Graphics2D) g;
27
28         g2.draw(box);
29      }
30
```

Continued

ch09/timer/RectangleComponent.java (cont.)

```
31      /**
32         Moves the rectangle by a given amount.
33         @param x the amount to move in the x-direction
34         @param y the amount to move in the y-direction
35      */
36      public void moveBy(int dx, int dy)
37      {
38         box.translate(dx, dy);
39         repaint();
40      }
41   }
```

ch09/timer/RectangleMover.java

```
1  import java.awt.event.ActionEvent;
2  import java.awt.event.ActionListener;
3  import javax.swing.JFrame;
4  import javax.swing.Timer;
5
6  /**
7     This program moves the rectangle.
8  */
9  public class RectangleMover
10 {
11    private static final int FRAME_WIDTH = 300;
12    private static final int FRAME_HEIGHT = 400;
13
14    public static void main(String[] args)
15    {
16       JFrame frame = new JFrame();
17
18       frame.setSize(FRAME_WIDTH, FRAME_HEIGHT);
19       frame.setTitle("An animated rectangle");
20       frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
21
```

Continued

ch09/timer/RectangleMover.java (cont.)

```
22          final RectangleComponent component = new RectangleComponent();
23          frame.add(component);
24
25          frame.setVisible(true);
26
27          class TimerListener implements ActionListener
28          {
29             public void actionPerformed(ActionEvent event)
30             {
31                component.moveBy(1, 1);
32             }
33          }
34
35          ActionListener listener = new TimerListener();
36
37          final int DELAY = 100; // Milliseconds between timer ticks
38          Timer t = new Timer(DELAY, listener);
39          t.start();
40       }
41    }
```

Self Check 9.21

Why does a timer require a listener object?

Answer: The timer needs to call some method whenever the time interval expires. It calls the `actionPerformed` method of the listener object.

لماذا توقيت تتطلب كائن المستمع؟
الإجابة: يحتاج الموقت لاستدعاء بعض الطريقة كلما انتهاء الفترة الزمنية. ويدعو طريقة actionPerformed الكائن المستمع.

Self Check 9.22

What would happen if you omitted the call to `repaint` in the `moveBy` method?

Answer: The moved rectangles won't be painted, and the rectangle will appear to be stationary until the frame is repainted for an external reason.

ماذا سيحدث إذا حذفت الدعوة إلى إعادة رسم في طريقة moveBy؟
الإجابة: لن يتم رسم المستطيلات المنقولة، وسيظهر المستطيل لتكون ثابتة حتى يتم طلاؤها إطار لسبب خارجي.

Mouse Events

- Use a mouse listener to capture mouse events
- Implement the `MouseListener` interface:

استخدام المستمع الماوس لالتقاط أحداث الماوس
تطبيق واجهة MouseListener:

```
public interface MouseListener
{
   void mousePressed(MouseEvent event);
   // Called when a mouse button has been pressed on a
   // component
   void mouseReleased(MouseEvent event);
   // Called when a mouse button has been released on a
   // component
   void mouseClicked(MouseEvent event);
   // Called when the mouse has been clicked on a component
   void mouseEntered(MouseEvent event);
   // Called when the mouse enters a component
   void mouseExited(MouseEvent event);
   // Called when the mouse exits a component
}
```

Mouse Events

- `mousePressed`, `mouseReleased`:

Called when a mouse button is pressed or released

يسمى عندما يتم الضغط على زر الماوس أو الإفراج عنهم

- `mouseClicked`:

If button is pressed and released in quick succession, and mouse hasn't moved

إذا تم الضغط على زر وأفرج عنه في تتابع سريع، ولم تحرك الماوس

- `mouseEntered`, `mouseExited`:

Mouse has entered or exited the component's area

الماوس دخلت أو خرجت منطقة المكون

Mouse Events

- Add a mouse listener to a component by calling the `addMouseListener` method:

```
public class MyMouseListener implements MouseListener
{
   // Implements five methods
}
MouseListener listener = new MyMouseListener();
component.addMouseListener(listener);
```

إضافة المستمع الماوس إلى العنصر بواسطة استدعاء الأسلوب addMouseListener:

- Sample program: enhance `RectangleComponent` — when user clicks on rectangle component, move the rectangle

نموذج البرنامج: تعزيز RectangleComponent - عندما ينقر المستخدم على عنصر المستطيل، حرك المستطيل

ch09/mouse/RectangleComponent.java

```
1  import java.awt.Graphics;
2  import java.awt.Graphics2D;
3  import java.awt.Rectangle;
4  import javax.swing.JComponent;
5
6  /**
7     This component displays a rectangle that can be moved.
8  */
9  public class RectangleComponent extends JComponent
10 {
11    private static final int BOX_X = 100;
12    private static final int BOX_Y = 100;
13    private static final int BOX_WIDTH = 20;
14    private static final int BOX_HEIGHT = 30;
15
16    private Rectangle box;
17
18    public RectangleComponent()
19    {
20       // The rectangle that the paintComponent method draws
21       box = new Rectangle(BOX_X, BOX_Y, BOX_WIDTH, BOX_HEIGHT);
22    }
23
```

Continued

ch09/mouse/RectangleComponent.java (cont.)

```
24      public void paintComponent(Graphics g)
25      {
26         Graphics2D g2 = (Graphics2D) g;
27
28         g2.draw(box);
29      }
30
31      /**
32         Moves the rectangle to the given location.
33         @param x the x-position of the new location
34         @param y the y-position of the new location
35      */
36      public void moveTo(int x, int y)
37      {
38         box.setLocation(x, y);
39         repaint();
40      }
41   }
```

Mouse Events

- Call `repaint` when you modify the shapes that `paintComponent` draws:

```
box.setLocation(x, y);
repaint();
```

استدعاء إعادة رسم عند تعديل الأشكال التي تلفت paint Component:

box.setLocation (س، ص)؛

إعادة رسم ()؛

Mouse Events

- Mouse listener: if the mouse is pressed, `listener` moves the rectangle to the mouse location:

الماوس المستمع: إذا تم الضغط على الماوس، المستمع يتحرك المستطيل إلى الموقع الفأر:

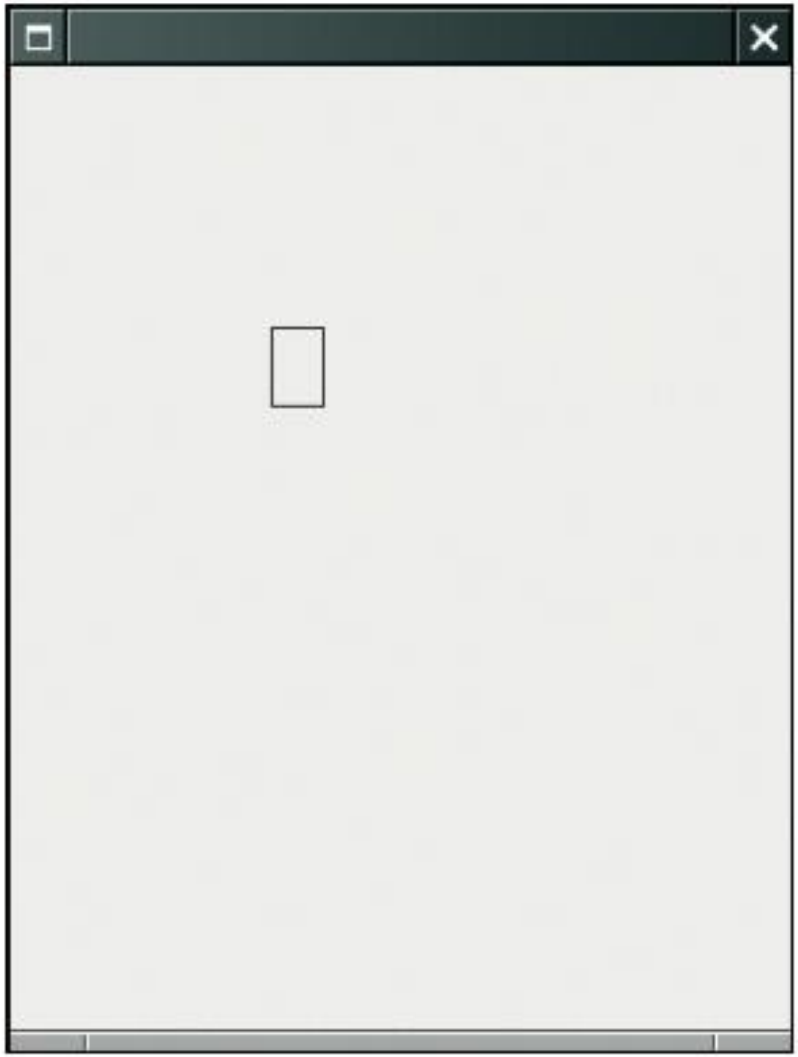
```
class MousePressListener implements MouseListener
{
   public void mousePressed(MouseEvent event)
   {
      int x = event.getX();
      int y = event.getY();
      component.moveTo(x, y);

   }
   // Do-nothing methods
   public void mouseReleased(MouseEvent event) {}
   public void mouseClicked(MouseEvent event) {}
   public void mouseEntered(MouseEvent event) {}
   public void mouseExited(MouseEvent event) {}
}
```

- All five methods of the interface must be implemented; unused methods can be empty

يجب أن تنفذ جميع الطرق الخمسة واجهة؛ يمكن الأساليب غير المستخدمة تكون فارغة

`RectangleComponentViewer` Program Run

Figure 8
Clicking the Mouse Moves the Rectangle



ch09/mouse/RectangleComponentViewer.java

```
1  import java.awt.event.MouseListener;
2  import java.awt.event.MouseEvent;
3  import javax.swing.JFrame;
4
5  /**
6     This program displays a RectangleComponent.
7  */
8  public class RectangleComponentViewer
9  {
10    private static final int FRAME_WIDTH = 300;
11    private static final int FRAME_HEIGHT = 400;
12
13    public static void main(String[] args)
14    {
15       final RectangleComponent component = new RectangleComponent();
16
```

Continued

ch09/mouse/RectangleComponentViewer.java (cont.)

```
17          // Add mouse press listener
18
19          class MousePressListener implements MouseListener
20          {
21             public void mousePressed(MouseEvent event)
22             {
23                int x = event.getX();
24                int y = event.getY();
25                component.moveTo(x, y);
26             }
27
28             // Do-nothing methods
29             public void mouseReleased(MouseEvent event) {}
30             public void mouseClicked(MouseEvent event) {}
31             public void mouseEntered(MouseEvent event) {}
32             public void mouseExited(MouseEvent event) {}
33          }
34
35          MouseListener listener = new MousePressListener();
36          component.addMouseListener(listener);
37
```

Continued

ch09/mouse/RectangleComponentViewer.java (cont.)

```
38          JFrame frame = new JFrame();
39          frame.add(component);
40
41          frame.setSize(FRAME_WIDTH, FRAME_HEIGHT);
42          frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
43          frame.setVisible(true);
44       }
45    }
```

Self Check 9.23

Why was the `moveBy` method in the `RectangleComponent` replaced with a `moveTo` method?

Answer: Because you know the current mouse position, not the amount by which the mouse has moved.

لماذا تم طريقة moveBy في RectangleComponent استبدال طريقة MoveTo ل؟
الجواب: لأنك تعرف موضع الماوس الحالي، وليس المبلغ الذي انتقلت الماوس.

Self Check 9.24

Why must the `MousePressListener` class supply five methods?

Answer: It implements the `MouseListener` interface, which has five methods.

لماذا يجب على إمدادات من الدرجة MousePressListener خمس طرق؟
الجواب: إنه تطبق الواجهة MouseListener، التي لديها خمس طرق.