An Introduction to Polymorphism in Java:مقدمة عن تعدد الأشكال في جافا :

Method Overloadingطريقة التحميل الزائد

In this section, you will learn about one of Java's most exciting features: method overloading.
In Java, two or more methods within the same class can share the same name, as long as their parameter declarations are different. When this is the case, the methods are said to be *overloaded,* and the process is referred to as *method overloading.* Method overloading is one of the ways that Java implements polymorphism.

في هذا القسم ، ستتعرف على إحدى أكثر ميزات Java إثارة: طريقة التحميل الزائد.
في Java ، يمكن أن تشترك طريقتان أو أكثر ضمن نفس الفئة في نفس الاسم ، طالما أن تعريفات المعلمات مختلفة. عندما تكون هذه هي الحالة ، يُقال إن الطرق محملة بشكل زائد ، ويشار إلى العملية على أنها طريقة التحميل الزائد. طريقة التحميل الزائد هي إحدى الطرق التي تنفذ بها Java تعدد الأشكال.

In general, to overload a method, simply declare different versions of it. The compiler takes care of the rest. You must observe one important restriction: the type and/or number of the parameters of each overloaded method must differ. It is not sufficient for two methods to differ only in their return types. (Return types do not provide sufficient information in all cases for Java to decide which method to use.) Of course, overloaded methods *may* differ in their return types, too. When an overloaded method is called, the version of the method whose parameters match the arguments is executed.

بشكل عام ، لإفراط في تحميل طريقة ما ، ما عليك سوى التصريح عن إصدارات مختلفة منها. المترجم يعتني بالباقي. يجب أن تلاحظ قيدًا واحدًا مهمًا: يجب أن يختلف نوع و / أو عدد معلمات كل طريقة محملة بشكل زائد. لا يكفي أن تختلف طريقتان فقط في أنواع الإرجاع الخاصة بهما. (لا توفر أنواع الإرجاع معلومات كافية في جميع الحالات لـ Java لتقرير الطريقة التي يجب استخدامها.) بالطبع ، قد تختلف الطرق ذات التحميل الزائد في أنواع الإرجاع أيضًا. عندما يتم استدعاء طريقة محملة بشكل زائد ، يتم تنفيذ إصدار الطريقة التي تتطابق معلماتها مع الوسائط.

Here is a simple example that illustrates method overloading:

فيما يلي مثال بسيط يوضح طريقة التحميل الزائد:

```
// Demonstrate method overloading. // شرح طريقة التحميل الزائد.
class Overload {
void ovlDemo() {
 System.out.println("No parameters");
}
// Overload ovlDemo for one integer parameter.الزائد  ovlDemo لمعلمة عدد صحيح واحد.
void ovlDemo(int a) {
System.out.println("One parameter: " + a);
}
// Overload ovlDemo for two integer parameters.
                                                    Overload ovlDemo // لمعلمات عدد صحيحين.
int ovlDemo(int a, int b) {
System.out.println("Two parameters: " + a + " " + b);return a + b;
```

```
}
// Overload ovlDemo for two double parameters.
```

// Overload ovlDemo لمعلمتين مزدوجتين.

```
double ovlDemo(double a, double b) {
System.out.println("Two double parameters: " +a + " "+ b);
return a + b;
}
}
class OverloadDemo {
public static void main(String args[]) {Overload ob = new Overload();
int resI; double resD;
// call all versions of ovlDemo()ob.
```

// استدعاء جميع إصدارات ob () ovlDemo.

```
ovlDemo();
System.out.println();
```

```
ob.ovlDemo(2);
 System.out.println();
resI = ob.ovlDemo(4, 6);
System.out.println("Result of ob.ovlDemo(4, 6): " +resI);

System.out.println();

resD = ob.ovlDemo(1.1,  2.32);
 System.out.println("Result of ob.ovlDemo(1.1, 2.32): " +resD);
}
}
```

This program generates the following output:

```
No parameters One parameter: 2
Two parameters: 4 6
Result of ob.ovlDemo(4, 6): 10
Two double parameters: 1.1 2.32
Result of ob.ovlDemo(1.1, 2.32): 3.42
```

As you can see, **ovlDemo( )** is overloaded four times. The first version takes no parameters, the second takes one integer parameter, the third takes two integer parameters, and the fourth takes two **double** parameters. Notice that the first two versions of **ovlDemo( )** return **void**, and the second two return a value.

كما ترى ، تم تحميل ovlDemo () فوق طاقته أربع مرات. الإصدار الأول لا يأخذ أي معلمات ، والثاني يأخذ معلمة صحيحة واحدة ، والثالث يأخذ معلمتين صحيحين ، والرابع يأخذ معلمتين مزدوجتين. لاحظ أن الإصدارين الأولين من ovlDemo () يُرجعان قيمة فارغة ، بينما يُرجع الإصداران الآخران قيمة.

Polymorphism تعدد الأشكال

- Polymorphism means *many* (poly) *shapes* (morph)
- In Java, polymorphism refers to the fact that you can have multiple methods with thesame name in the same class
- There are two kinds of polymorphism:
  - Overloading
    - Two or more methods with different signatures
  - Overriding
    - Replacing an inherited method with another having the same signature

- تعدد الأشكال يعني العديد من الأشكال (المتعددة) (شكل)
- في Java ، يشير تعدد الأشكال إلى حقيقة أنه يمكن أن يكون لديك طرق متعددة بنفس الاسم في نفس الفئة
- هناك نوعان من تعدد الأشكال:
- التحميل الزائد

طريقتان أو أكثر بتوقيعات مختلفة

- تجاوز

استبدال طريقة موروثة بأخرى لها نفس التوقيع

Overloading

```
class Test {
public static void main(String args[]) {myPrint(5);
myPrint(5.0);
}

static void myPrint(int i) { System.out.println("int i = " + i);
}
```

```
static void myPrint(double d) { // same name, different parameters
System.out.println("double d = " + d);
}
}
```

int i = 5
double d = 5.0

Why overload a method?؟لماذا تفرط في طريقة

- So you can use the same names for methods that do essentially the same thing
  - Example: println(int), println(double), println(boolean), println(String), etc.
- So you can supply defaults for the parameters:

```
int increment(int amount) {count = count + amount;return count;
}
int increment() {
return increment(1);
}
```

  - Notice that one method can call another of the same name
- So you can supply additional information:

```
void printResults() {
System.out.println("total = " + total + ", average = " +average);
}
void printResult(String message) { System.out.println(message + ":
");printResults();
}
```

لذا يمكنك استخدام نفس الأسماء للأساليب التي تقوم بنفس الشيء بشكل أساسي
مثال: println، println (int) println (مزدوج) ، println (منطقي) ، println (سلسلة) ، إلخ.
حتى تتمكن من توفير الإعدادات الافتراضية للمعلمات:
زيادة int (كمية int) {count = count + مبلغ؛ عدد العودة
{
زيادة int () {
زيادة العائد (1) ؛
{
لاحظ أن إحدى الطرق يمكنها استدعاء أخرى بنفس الاسم
حتى تتمكن من توفير معلومات إضافية:
نتائج طباعة باطلة () {
System.out.println("total =" + total + "، المتوسط =" + average) ؛
{
void printResult (String message) {System.out.println (message + ":") ؛
printResults () ،
{

Legal assignments التخصيصات القانونية

```
class Test {
public static void main(String args[]) {double d;
int i;
d = 5;                          // legal
i = 3.5;                        // illegali = (int) 3.5;    // legal
}
}
```

- Widening is legal
- Narrowing is illegal

- التوسيع قانوني
- التضييق غير قانوني

Legal method calls:استدعاءات الطريقة القانونية :

```
class Test {
public static void main(String args[]) {myPrint(5);
}

static void myPrint(double d) {System.out.println(d);
}
}
```

- Legal because parameter transmission is equivalent to assignment
- myPrint(5) is like double d = 5; System.out.println(d);

■ قانوني لأن إرسال المعلمات يعادل التخصيص

Illegal method calls: غير قانونية

```
class Test {
public static void main(String args[]) {myPrint(5.0);
}

static void myPrint(int i) {System.out.println(i);
}
}
```

- Illegal because parameter transmission is equivalent to assignment
- myPrint(5.0) is like int i = 5.0; System.out.println(i);

■ غير قانوني لأن إرسال المعلمات يعادل التخصيص ■ ■