



قسم الهندسة المعلوماتية

برمجة 3

Java Programming

ا. د. علي عمران سليمان

محاضرات الأسبوع الثاني

Static Method , and Data fields

الفصل الأول 2024-2025

Contents 1



- | | |
|--|---|
| <ol style="list-style-type: none">1. Objects and Classes .2. UML class diagrams .3. Performing output Displaying with print, println, printf.4. Performing Input Scanner and some of its methods.5. default constructor.6. Overloaded Constructors, and methods.7. Static Method , and Data fields.8. Call by value and references.9. copy constructor.10. inherited class. | <ol style="list-style-type: none">11. Inheritance and Constructors.12. Overriding Superclass Methods.3.6 Class JOptionPane Using Dialog Boxes showMessageDialog(), showInputDialog()4.15 GUI &Graphics,4.15 Creating Simple Drawings—Displaying and drawing lines on the screen5.11 Drawing Rectangles and Ovals—Using shapes to represent data. |
|--|---|

References

- Deitel & Deitel, Java How to Program, Pearson; 10th Ed(2015)

- د.علي سليمان، بني معطيات بلغة JAVA، جامعة تشرين 2013-2014

Objects and Classes 1

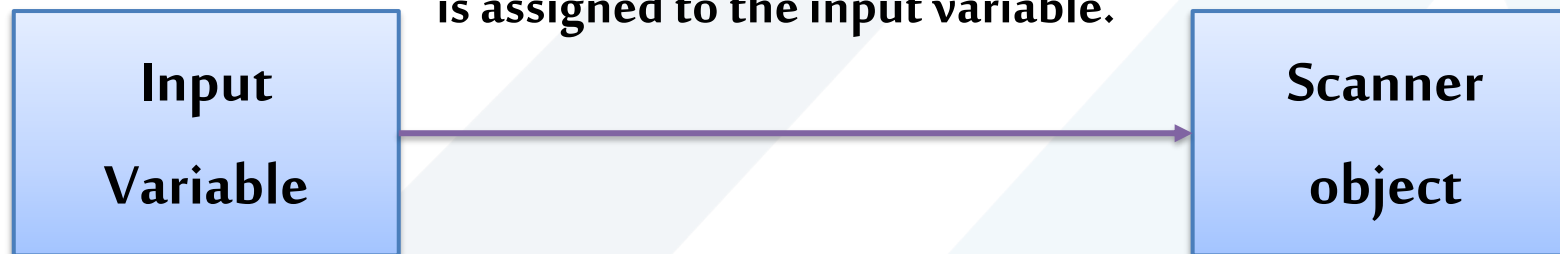
- Class: الصنف كود يصف أنماط الكائنات التي يمكن أن تشتق منه أي مفهوم مجرد لا يمثل في ذاكرة الحاسب، يصف المعطيات التي يمكن للكائن أن تملكها (حقول المعطيات، أو المعطيات الأعضاء، الخصائص Properties جميعها تعبر عن الصفات Attribute)، والأحداث التي يمكن للكائن أن يتضمنها (المناهج methods، الأداء Actions، العمليات operations جميعها تعبر عن السلوكيات behaviors).
- يمكن التفكير بالصنف كمخطط blueprint لبناء الكائنات الفعلية (نمط).
- عند جريان البرنامج، يمكن أن تستخدم الاصناف من أجل بناء العديد من الكائنات في الذاكرة وبأنواع المطلوبة.
- الصنف هو مصدر الكائنات وتعرف بأمثال الصنف.

Example:

This expression creates a
Scanner object in memory.

```
Scanner input = new Scanner(System.in);
```

The object's memory address
is assigned to the input variable.



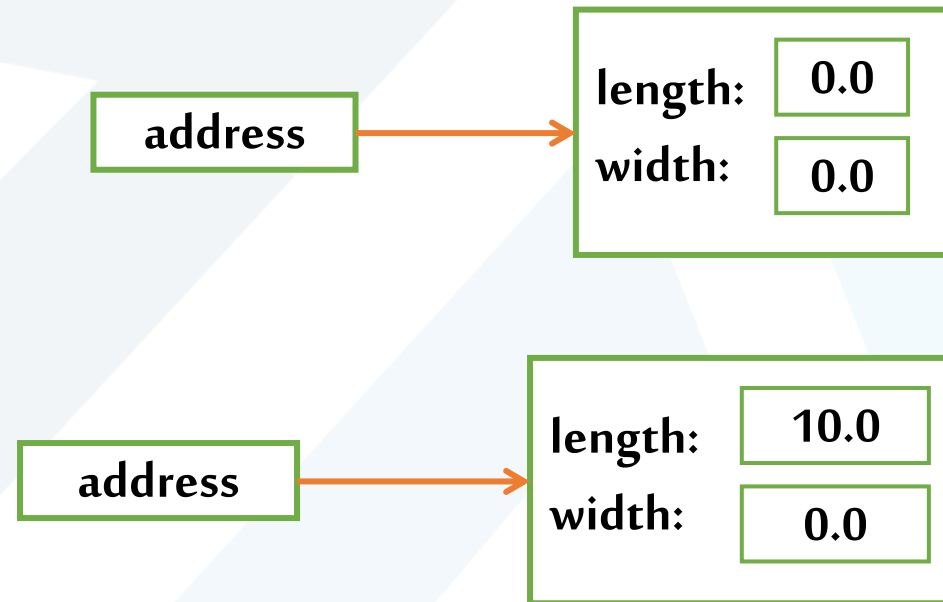
Creating a Rectangle object

The box variable holds the address of the Rectangle object.



```
Rectangle box = new Rectangle ();
```

A Rectangle object



Calling the setLength Method
`box.setLength(10.0);`

This is the state of the box object after the setLength method executes.

Instance Fields and Methods

- الحقول والمناهج المصريح عنها سابقاً تدعى أمثال حقول وأمثال مناهج.
- كل كائن من صنف يملك نسخة خاصة به عن أمثال الحقول وأمثال مناهج.
- والمنهج المثل هو المنهج الذي لم يعرف كـ `static`.
- الأمثال للحقول والمناهج تتطلب إنشاء كائن لاستخداماتها اللاحقة والمناهج `static` ترتبط بالصنف ولا تحتاج لكائن كي تستثمر بل يكتب اسم الصنف واسم المنهج وبينهما نقطة.
- كل مستطيل (كائن) يمكن أن يملك ابعاده الخاصة به.

States of Three Different Rectangle Objects

The football variable holds the address of a Rectangle Object.

address

length: 205.0
width: 68.0

The Volleyball variable holds the address of a Rectangle Object.

address

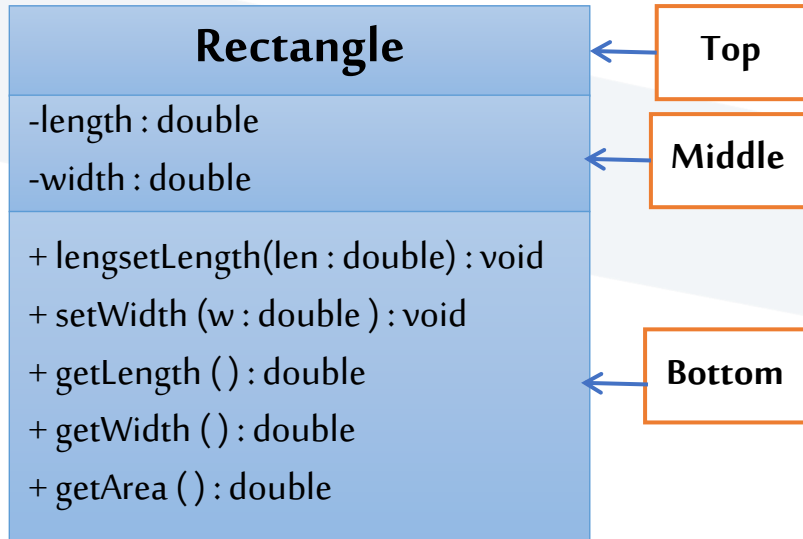
length: 18.0
width: 9.0

The basketball variable holds the address of a Rectangle Object.

address

length: 28.0
width: 15.0

UML class diagrams 1



- يستخدم مخطط UML (Unified Modeling Language) للصف لتلخيص صفات ومناهج الصف.
- تساعد مخططات UML مصممي الأنظمة على تحديد نظام بطريقة رسومية ومستقلة عن لغة البرمجة ، قبل أن يتم تنفيذ النظام بلغة برمجة ما.
- في تخطيط UML يمثل الصف بمستطيل يحوي ثلاث مكونات compartments.
- الجزء العلوي : يتضمن اسم الصف بخط عريض وبهامش توسط .
- الجزء الوسطي : يتضمن أسماء الحقول وتسبق بإشارة - للنوع الخاص ، + للعام ، و # للنوع المحمي يليه : وبعدها نوع معطيات الحقل.
- الجزء السفلي : يتضمن أسماء المناهج حيث يذكر اسم المنهج ومسبوقاً بإشارة نوع الوصول (+ , # , -) عام , محمي , خاص على الترتيب كما هو في حالة أسماء الحقول.
- قوسان دائريان فارغان إذا لم يمرر متغيرات للمنهج، وفي حال تمريرها يذكر الاسم للمتغير يلي كل منها : ثم نمطه وتفصل المتغيرات (الاسماء) عن بعضها بعضاً بفواصل عادية.
- يوضع مابعد القوس الدائري المغلق : ونوع القيمة المعادة وإذا كان لا يعيد شيء يوضع void.
- الباني يوضع في حال كتابته وفق الآتي: يوضع في بداية المناهج بذكر ما بين «*constructor*» ثم اسم صنفه وهو اسمه بطبيعة الحال وبعدها ينطبق عليه ما ينطبق على المناهج الأخرى، إلا أنه لن يعيد شيء ولا حتى void.

Writing a Class

- A Rectangle object will have the following fields and methods:

```
public class Rectangle
{
    private double length;
    private double width;

    public Rectangle () { System.out.println("default constructor");}
    public Rectangle (double l, double w) {length=l;width = w;
        System.out.println("constructor with argument");}
    public void setLength(double le) { length=le;}
    public void setWidth(double wi) {width = wi;}
    public double getLength() { return length;}
    public double getWidth() { return width;}
    public double perRectangle() {return (length + width)*2; }
    public double areaRectangle() { return length * width; }
}
```

Setter , Mutator

Getter, Accessor

```
Rectangle box1 = new Rectangle(); Rectangle box2 = new Rectangle(15.0, 10.0);
```

Rectangle

-length : double

-width : double

+«constructor» Rectangle ()

+«constructor» Rectangle(length double , width: double)

+ setLength(length : double) : void

+ setWidth (width : double) : void

+ getLength () : double

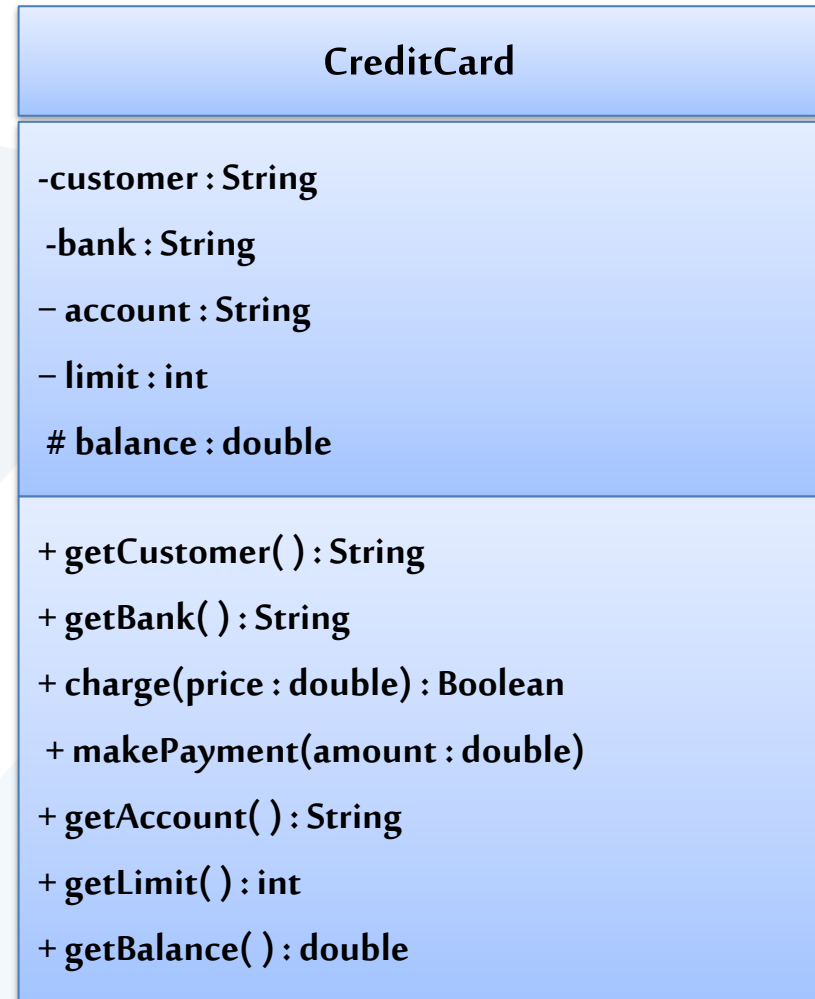
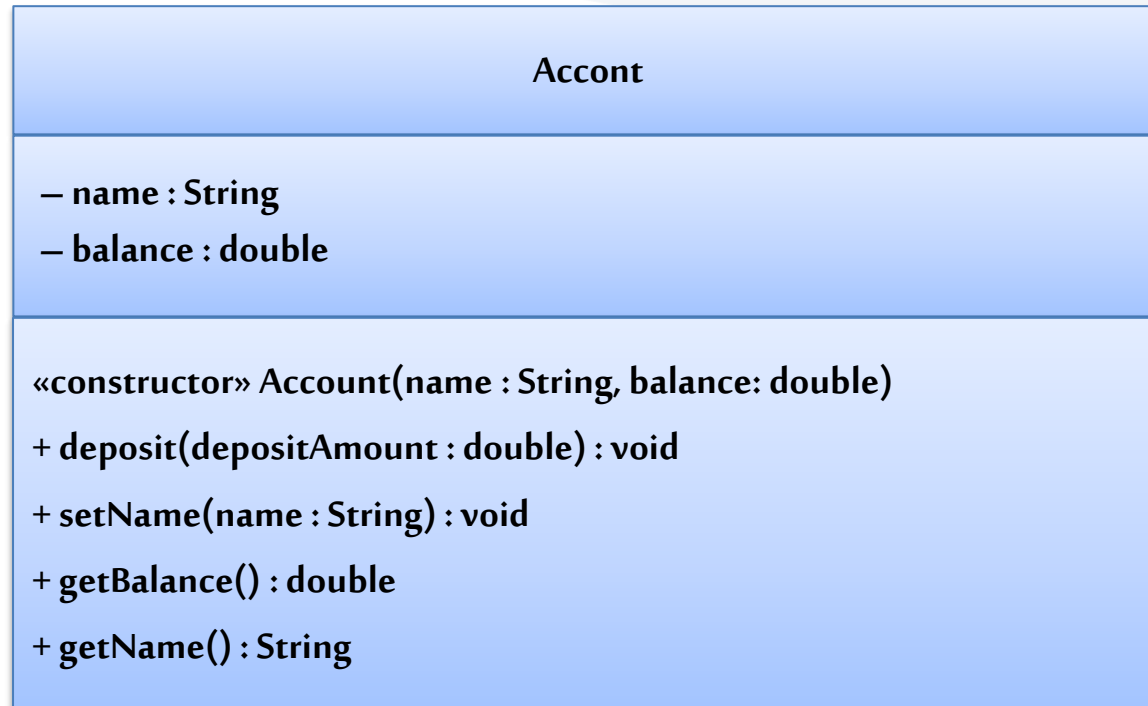
+ getWidth () : double

+ perRectangle () : double

+ areaRectangle () : double

```
public class RectangleTest {  
    public static void main(String[] args) {  
        Rectangle r1= new Rectangle();  
        System.out.println("length= "+ r1.getLength() + "width= "+ r1.getWidth());  
        r1.setLength(33);  
        r1.setWidth(22);  
        System.out.println("length= "+ r1.getLength() + "width= "+ r1.getWidth());  
        //If the height and width are made public, we can directly access them  
        //1 System.out.println("length= "+ r1.length + "width= "+ r1.width);  
        Rectangle r2= new Rectangle(30, 20);  
        System.out.println("length= "+ r2.getLength() + "width= "+ r2.getWidth());  
        System.out.println("perimeter= "+ r2.perRectangle() + "\narea= "+ r2.areaRectangle());  
        //1 System.out.println("length= "+ r2.length + "width= "+ r2.width);  
    }  
}
```

Examples 2



Uninitialized Local Reference Variables

- Reference variables can be declared without being initialized.

Rectangle box;

- This statement does not create a Rectangle object, so it is an uninitialized local reference variable.
- A local reference variable must reference an object before it can be used, otherwise a compiler error will occur.

```
box = new Rectangle();
```



- Performing Output with System.out.print

```
System.out.print("Welcome to Java Programming!");
```

- Where System is the class name, it is declared as final. The out is an instance of the System class and is of type PrintStream. Its access specifiers are public and final.
- Class System is part of package java.lang. Notice that class System will not be imported with an import declaration at the beginning of the program.

• توجد ثلاث نسخ من منهج للطباعة print:

1- المنهج print() يوجه الكمبيوتر لتنفيذ إجراء - عرض الأحرف الموجودة بين علامات الاقتباس المزدوجة (لا يتم عرض علامات الاقتباس نفسها) وتعتبر كبارامتر وحيد، وإذا وجد ما بين القوسين أكثر من بارامتر مثل أعداد ونصوص وتعابير وجب استخدام (+) لقسرها أو لضمها لبعضها، وسيتم ترك مؤشر الاخراج حيث انتهت الطباعة.

2- منهج println()، عندما تكتمل مهمة المنهج System.out.println()، فإنه ينقل مؤشر الاخراج حيث انتهت الطباعة إلى بداية السطر التالي.

Escape sequence

- Escape sequence

Description

<code>\n</code>	Newline. Position the screen cursor at the beginning of the next line.
<code>\t</code>	Horizontal tab. Move the screen cursor to the next tab stop.
<code>\r</code>	Carriage return. Position the screen cursor at the beginning of the current line—do not advance to the next line. Any characters output after the carriage return overwrite the characters previously output on that line.
<code>\\</code>	Backslash. Used to print a backslash character.
<code>\"</code>	Double quote. Used to print a double-quote character. For example, <code>System.out.println("\"in quotes\"");</code> displays "in quotes".

Some common escape sequences.

- The System.out.printf method (f means “formatted”) displays formatted data.

```
System.out.printf("%s\n%s\n", "Welcome to", "Java Programming!");
```

```
//System.out.printf("%4d % , 20.2f\n %n", year, amount); //left justified %-20s
```

3- منهج printf() يحتاج المنهج إلى ثلاث وسطاء أو بارامترات، يتم وضعها في القائمة. تبدأ محددات التنسيق بعلامة النسبة المئوية (%) متبوعة بحرف يمثل نوع البيانات ثم فاصمة، تفصل البيانات عن بعضها البعض بفواصل (يمكن للبيانات ان تتضمن تعبير).
(مثلاً: محدد التنسيق % s هو عنصر يدل لسلسلة بحروف صغيرة ، %S سلسلة بحروف كبيرة، % d عنصر يدل لقيمة عدد صحيح، %f عنصر يدل لقيمة حقيقية،

محدد التنسيق % ,20.2f : حيث % علامة التنسيق، الفاصلة (,) يشير إلى أنه يجب إخراج قيمة النقطة العائمة بفواصل لجميع كل ثلاث ارقام بمجموعة ومؤلفه من 20 مرتبة ، مرتبتين عشريتين، غياب الإشارة قبل 20 أي موجبة وبالتالي الهامش يميني).

- لذا فإن هذا المثال الأول يستبدل "Welcome to " مع %s الأول والانتقال للسطر التالي بفعل %n وبعدها يستبدل "java Programming!" مع %s الثانيه ثم الانتقال للسطر التالي من %n الثانية، ويمكن استخدام \n بدل %n مع printf.
- لايمكن استخدام %n للانتقال للسطر التالي بدل \n في وسطاء System.out.print(); أو System.out.println();
- ملاحظة: سيتم تغطية هذه الحالات في تمرين اخر المحاضرة.

Performing input 1

Input with `input.nextInt()`;

```
import java.util.Scanner; // program uses class Scanner
// create a Scanner to obtain input from the command window
int number ;
```

• تصريح الاستيراد الذي يساعد المترجم على تحديد صنف يتم استخدامه في هذا البرنامج. يشير إلى أن البرنامج يستخدم صنف Scanner المحددة مسبقًا من الحزمة المسماة `java.util`.

• يحجز للمتغير موقع في ذاكرة الكمبيوتر حيث يمكن تخزين القيم لاستخدامها لاحقًا في أحد البرامج. يجب التصريح عن المتغيرات باسم ونوع قبل استخدامها. يتيح اسم المتغير للبرنامج الوصول إلى القيمة الموجودة في الذاكرة.

Performing input 2

```
new Scanner(System.in);
```

```
Scanner input = new Scanner(System.in);
```

in is basically the instance of *InputStream* from *java.lang* package.

• إعلان عن كائن بالاسم (input) ونوع (Scanner) للكائن المستخدم في هذه التعليمة. يسمح Scanner للبرنامج بقراءة البيانات لاستخدامها في البرنامج.

يمكن أن تأتي البيانات من عدة مصادر، مثلاً من المستخدم عبر لوحة المفاتيح أو ملف على القرص. قبل استخدام Scanner، يجب أن تقوم بإنشائه وتحديد مصدر البيانات.

يجب تهيئة الكائن input من النمط Scanner من خلال منادات الباني العبارة في الطرف اليميني من إشارة النسب.

• `new Scanner(System.in);` يستخدم هذا التعبير الكلمة الأساسية `new` لإنشاء كائن Scanner في ذاكرة الحاسب من خلال منادات المنهج الباني للصنف Scanner مع ارسال بارامتر من النوع System.in له، ليقرأ المدخلات التي كتبها المستخدم على لوحة المفاتيح كونها وحدة الإدخال القياسية.

```
number = input.nextInt();
```

```
System.out.print("Enter integer number: "); // prompt read int number from user  
number = input.nextInt(); // read Double number, line nextDouble(), nextLine()
```

- يستخدم المنهج `nextInt` من الكائن `input` لإدخال عدد صحيح من المستخدم عبر لوحة المفاتيح. في هذه المرحلة، ينتظر البرنامج أن يقوم المستخدم بكتابة الرقم والضغط على مفتاح `Enter` لإرسال الرقم إلى المكان المحجوز للمتغير `number` المعرف كعدد صحيح قبل استخدامه.
- نضع نتيجة استدعاء الأسلوب `nextInt()` في المتغير `number` باستخدام عامل التخصيص `=`.
- عامل التخصيص `=` يسمى العامل الثنائي لأنه يحتوي على معاملين ونتيجة لاستدعاء الأسلوب `input.nextInt()`. المتغير والقيمة التي ستنسب له.
- يستخدم `input.nextDouble()` لإدخال عدد حقيقي و `input.nextFloat()` لإدخال عدد `float`، وعند الإدخال يضاف لنهاية العدد `f` و `nextLine` لإدخال شريط محرفي و `char s = input.next().charAt(0);` لإدخال وإسناد المحرف الأول للمتغير `s`.
- `char str2 = in.next().charAt(3);` سيتم إدخال أربعة محارف وإسناد المحرف الرابع للمتغير `str2`.

- عند إنشاء كائن سيتم نداء المنهج الباني constructor بشكل تلقائي.
- إذا لم يتم كتابة الباني من قبل المبرمج، ستقوم Java بتوفير provides واحد عند إجراء المطابقة للصنف، والباني المنشئ من Java يعرف بالباني الافتراضي.
- سيتم الاسناد لكل الحقول العددية بما فيها المحرفية القيمة الصفر.
- سيتم الاسناد لكل الحقول المنطقية إلى false.
- سيتم الاسناد لكل الحقول المرجعية إلى null.
- الباني الافتراضي default constructor يكون بدون وسطاء no-arg ويستخدم للتجهيز بالقيم الافتراضية.
- الباني الافتراضي default constructor لن يتم تنفيذه من قبل Java إذا كان هناك بانياً آخر مكتوباً ولم تتم كتابته.
- يمكننا كتابة الباني الخاص بنا بدون وسطاء no-arg وفق أحد الحالتين في الأولى يضع القيم الافتراضية 0.0, 0.0 في الثانية يضع القيم 15.0, 10.0

```
public Rectangle(){}
```

OR

```
public Rectangle() { length = 15.0; width = 10.0;
```

```
}  
Rectangle box1 = new Rectangle();
```

عندها يمكن أن نشق كائن

Overloading Methods and Constructors

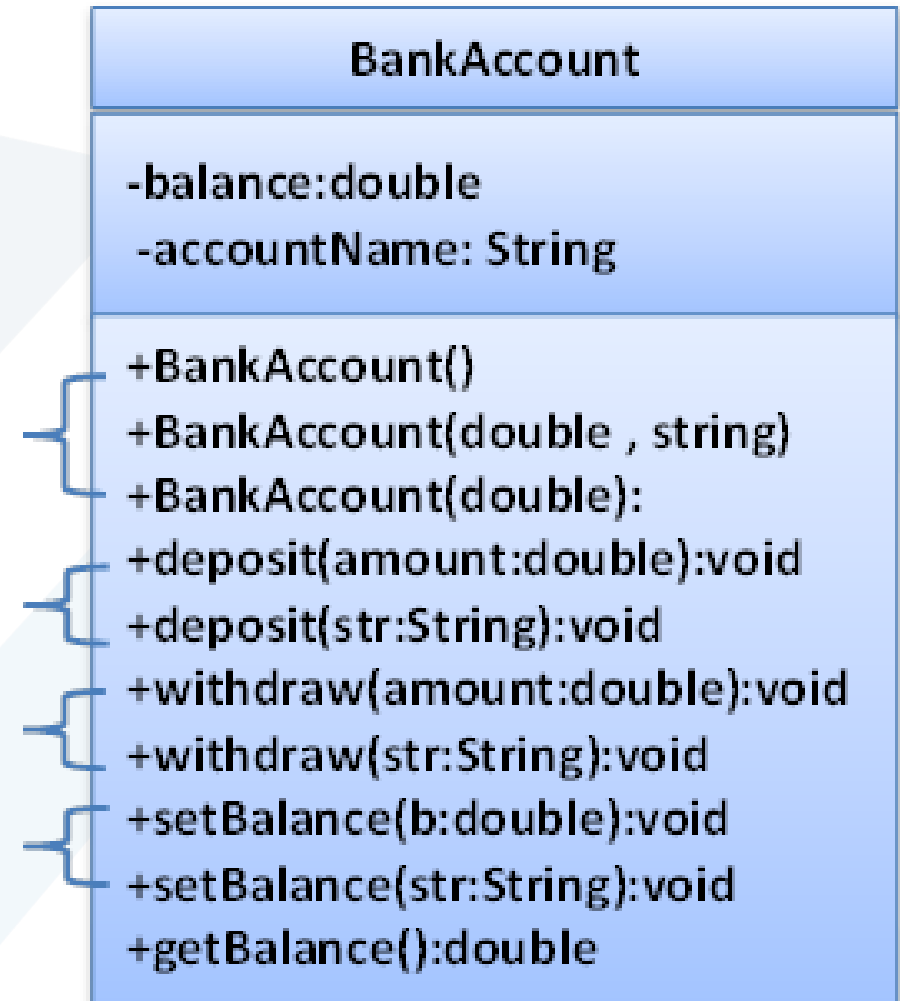
- قد يكون لطريقتين أو أكثر في الصنف نفس الاسم طالما أن قوائم المتغيرات الممرة الخاصة بهم مختلفة نوعاً أو ترتيباً أو كماً أو أكثر من واحد مما سبق.
- يطلق عليه طريقة التحميل الزائد Overloading، وهذا ينطبق أيضاً على constructors.
- أسلوب التحميل الزائد مهم لأنك في بعض الأحيان تحتاج إلى عدة طرق مختلفة الإجراء وبنفس الاسم.

```
public int add(int num1, int num2)
{ int sum = num1 + num2; return sum; }
```

```
public String add (String str1, String str2)
{ String combined = str1 + str2; return combined; }
```

The BankAccount Example

- Overloaded Constructors
- Overloaded deposit methods
- Overloaded withdraw methods
- Overloaded setBalance methods



- متى نحتاج لاعضاء الصنف (مناهج وحقول) أن تكون static ثابتة؟
- عندما نرغب بأن تتبع هذه الاعضاء للصنف وليس للكائن فقط، أي عندما يكون المنهج خدمي utility classes (مثل توابع الرياضيات غير المعرفة في صنف Math) من مكتبة Java القياسية.
- وعندما يكون الحقل تابع لكل الكائنات (مثلاً عد الكائنات المشتقة من صنف) أي حقل مشترك.
- عندما نحتاج مناهج لتشغيلها ولانرغب بتخزين الكائنات، أي نحن لانرغب بتخزين معطيات عن هذه الكائنات فلا داعي لانشائها، ونوفر أماكن في الذاكرة كانت ستخصص للكائنات.
- لاستدعاء طريقة static أو استخدام حقل معطيات static يتم استخدام اسم الصنف، بدلاً من اسم الكائن.
- مثال استدعاء منهج الجزر التربيعي من الصنف Math وإعطائها القيمة 25.

• Example:

```
double val = Math.sqrt(25.0);
```

Class name

Static Method

Static Fields

- يتم الإعلان عن حقول الصنف بأنها static باستخدام الكلمة المفتاحية static بين محدد الوصول ونوع الحقل.
- يمكن تعريف scope من النمط static كما يلي.

```
class Test {  
    static int i;  
    static  
        { i=100;          float f=13.22f;          String s= "Adam";          }  
}
```

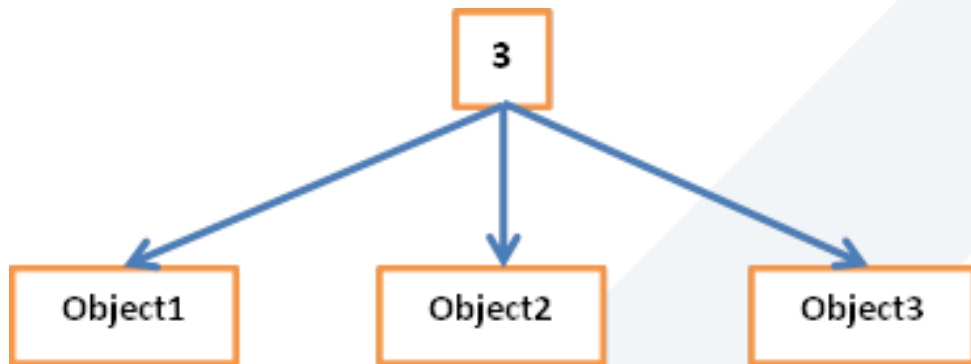
- إن ترتيب خطوات الترجمة تكون:
 - 1- تجهيز initialization المتغيرات من نمط static .
 - 2- تجهيز initialization المتغيرات من نمط Dynamic .
 - 3- مناداة الباني constructor .

• لا يمكن تعريف متحول static ضمن الطرق بل محصور بالاصناف.

- يتم تهيئة الحقول الثابتة الأولية إلى 0 إذا لم يتم إجراء تهيئة لها initialization.

- مثال لعد الكائنات المشتقة من صنف نضع `private static int instanceCount = 0;` ضمن الصنف ونزيد هذا الحقل بمقدار 1 في كل باني للصنف أو أن نبدأ بـ 500 وننقص واحد في كل باني.

`instanceCount` field (static).



• يتم الإعلان عن static Methods من خلال وضع الكلمة المفتاحية static بين معدّل الوصول ونوع القيمة المعاده.

```
public static int getInstanceCount() { return instanceCount; }
```

• عندما يحتوي صنف على طريقة static، لاداعي لإنشاء مثيل للصنف من أجل استدعاء الطريقة، بفرض أن الطريقة add ضمن الصنف Calc.

```
System.out.println(Calc.add(121, 112));
```

• تعتبر الطرق static ملائمة لأنها قد يتم استدعاؤها على مستوى الصنف.

• الحقول والطرق static تابعة للصنف وبالتالي في الحالة العامة لا تتواصل الطرق static مع الحقول أو الطرق غير static أي مع instance لأنها تابعة للكائنات، بل فقط مع الحقول static.

• يمكن الوصول إليها من الكائنات ولكن ذلك غير محبب.

• تستطيع الطرق غير static التواصل مع الحقول static.

```
public class Calc
{
    private static int instanceCount = 0;
    private int a;      private int b;
    public Calc()      { instanceCount++; }
    public Calc(int x, int y) { instanceCount++;      a=x;      b=y;      }
    public static int add(int aa, int bb) {      return aa+bb;      }
    public int sub()      {      return a-b; }
    public int mult()      { return a*b; }
    public static int getInstanceCount()      { return instanceCount; }
}
```

```
public class CalcTest {
    public static void main(String[] args) {
        Calc c1 = new Calc(10,20); Calc c0 = new Calc(); Calc c2 = new Calc(23,17); Calc c3 = new Calc();
        Calc c4 = new Calc(); System.out.println("th mult is0 = "+c0.mult());
        System.out.println("th mult is = "+c1.mult());
        System.out.println("th sub is = "+c2.sub());
        System.out.println(Calc.add(111,222));
        System.out.println("the number of object = "+Calc.getInstanceCount());
        int year=2020; double amount=2334443.446;
        System.out.printf("%4d %, 20.2f\n %n", year, amount);
        System.out.printf("%4d %, 20.2f\n %n", year*100, amount*2);
        System.out.println(year+year); } //end main
} //end class CalaTest
```

```
th mult is0 = 0
th mult is = 200
th sub is = 6
333
the number of object = 5
2020      2,334,443.45

202000    4,668,886.89

4040
```

Passing Objects as Arguments

- يمكن تمرير الكائنات إلى الأساليب كوسيطات.
- يقوم Java بتمرير جميع البارامترات للنمط الاولي primitive data type بالقيمة.
- عند تمرير كائن كوسيط ، يتم تمرير قيمة المتغير المرجعي له (العنوان) لا يتم تمرير نسخة من الكائن .
- المتغير المرجعي reference type هو عنوان الكائن في الذاكرة.
- عندما تتلقى طريقة متغير مرجعي كوسيط ، فإن أية تعديل للكائن ضمن الطريقة تعديل محتويات الكائن المشار إليه بواسطة المتغير، لأن العمل يتم على نفس المكان المحجوز في الذاكرة.

Passing Objects as Arguments

```
displayRectangle(box);
```

address

A Rectangle object

length: 12.0

width: 05.0

```
public static void displayRectangle(Rectangle r)
```

```
{
```

```
// Display the length and width.
```

```
System.out.println("Length: " + r.getLength() + " Width: " + r.getWidth());
```

```
}
```

```
account = getAccount();
```

address

A BankAccount Object

balance: 3200.0

```
public static BankAccount getAccount()  
{  
  ...  
  return new BankAccount(balance);  
}
```

Using The == operators with objects

- If we try the following:

```
Rectangle r1 = new Rectangle(10,50);
```

```
Rectangle r2 = new Rectangle(10,50);
```

```
if (r1 == r2) // This is a mistake
```

```
    System.out.println("The objects are the same.");
```

```
Else
```

```
    System.out.println("The objects are not the same.");
```

The objects are not the same.

only the addresses of the objects are compared.

سيتم طباعة :

• هناك طريقتان لنسخ كائن.

- لا يمكنك استخدام عامل النسب لنسخ ما يشير له المراجع (محتوى الكائن) بشكل مباشر بل يتم من خلال.

1- نسخة مرجعية فقط: هذا ببساطة هو نسخ عنوان كائن إلى متغير مرجعي لكائن آخر.

2- نسخة عميقة Deep copy يتضمن ذلك إنشاء مثيل جديد للفئة ونسخ القيم من كائن إلى كائن آخر.

```
Rectangle r1 = new Rectangle(100,50);
```

الحالة الاولى :

```
Rectangle r2 = new Rectangle(100,50);
```

```
r2=r1;
```

```
if (r1 == r2) System.out.println("The objects are the same.");
```

```
Else System.out.println("The objects are not the same.");
```

The objects are the same.

سيتم طباعة :

لوتم التعديل على إحداهما سينطبق على الاخر نظراً لإن الاثنان يؤشران لنفس المكان ضمن الذاكرة:

```
r2.setLength(66);
```

```
System.out.println(r1.getLength()); → 66
```


- A copy constructor accepts an existing object of the same class and clones it

الحالة الثانية:

ويمكن أن نحمل الباني بشكل زائد بارسال قيم الحقول وإسنادها أو بارسال كائن له مثل r2 ونسخ حقول المرسل لحقول المبنى:

```
public Rectangle () { System.out.println("default constructor");} // end constructor with out argument "default constructor"
public Rectangle (double l, double w) {length=l; width = w;
    System.out.println("constructor with argument");} // end constructor with argument
```

```
Rectangle r1 = new Rectangle(100,50);
```

```
public Rectangle(Rectangle r2)
{ length = r2.length; width = r2. width ; } // end Create copy constructor
```

```
Rectangle r2 = new Rectangle(r1);
```

OR

```
Rectangle r1= new Rectangle(13, 9); // Create r3, a copy of r1
```

```
Rectangle r3 = Rectangle (r1);
```

انتهت محاضرات الأسبوع الثاني