قسم الهندسة المعلوماتية

برمجة 3

Java Programming

ا. د. علي عمران سليمان

محاضرات الأسبوع الثاني

الفصل الثاني 2023-2024

Contents 1

References

- Deitel & Deitel, Java How to Program, Pearson; 10th Ed(2015)
- د.علي سليمان، بنى معطيات بلغة JAVA، جامعة تشرين 2013-2014

Objects and Classes 1

- Class: الصنف كود يصف أنماط الكائنات التي يمكن أن تشتق منه، 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 السوكيات behaviors).
  - يمكن التفكير بالصنف كمخطط blueprint لبناء الكائنات الفعلية.
  - عند جريان البرنامج، يمكن أن يستخدم الاصناف من أجل بناء العديد من الكائنات في الذاكرة وبالأنواع المطلوبة.
  - الصنف هو مصدر الكائنات تعرف بأمثال الصنف.

Objects and Classes 2

Example:

This expression creates a Scanner object in memory.

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

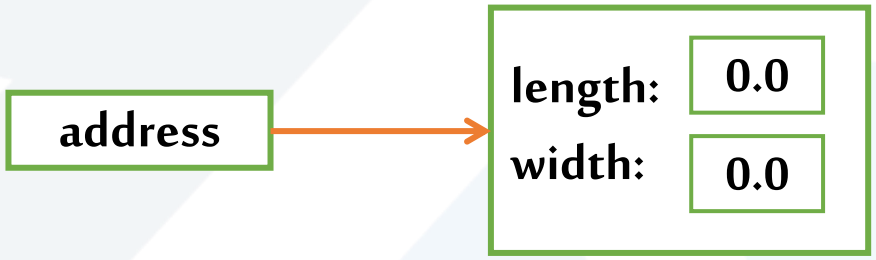
The object's memory address is assigned to the input variable.

Creating a Rectangle object

The box variable holds the address of the Rectangle object.

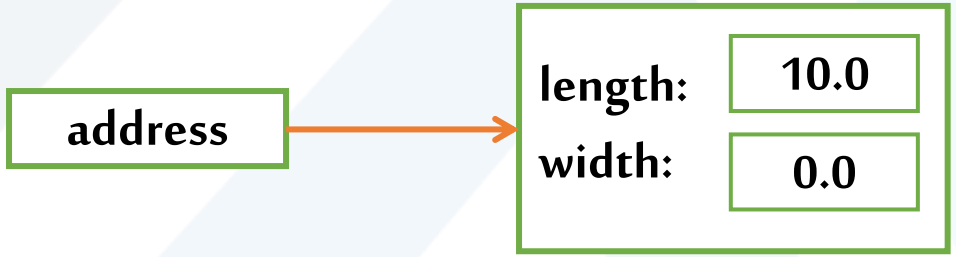
```
Rectangle box = new Rectangle ();
```

A Rectangle object



Calling the setLength Method

```
box.setLength(10.0);
```



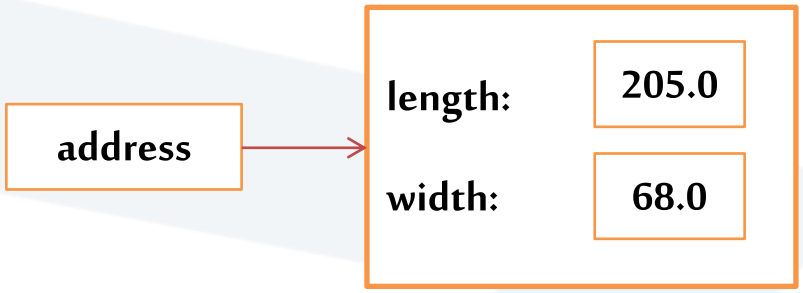
This is the state of the box object after the setLength method executes.

Instance Fields and Methods

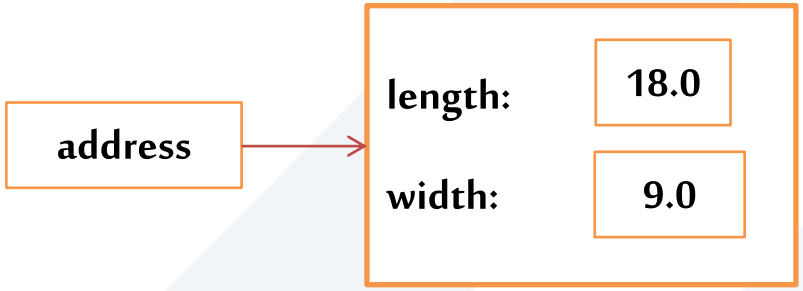
- الحقول والمناهج المصرح عنها سابقاً تدعى أمثال حقول وأمثال مناهج.
- كل كائن من صنف يملك نسخة خاصة به عن أمثال الحقول.
- والمنهج المثل هي المنهج الذي لم يعرف كا static.
- الأمثال للحقول والمناهج تتطلب إنشاء كائن لاستخداماتها اللاحقة والمناهج static ترتبط بالصنف ولاتحتاج لكائن كي تستثمربل يكتب اسم الصنف واسم المنهج وبينهما نقطة.
- كل مستطيل (كائن) يمكن أن يملك ابعاده الخاصة به.

States of Three Different Rectangle Objects

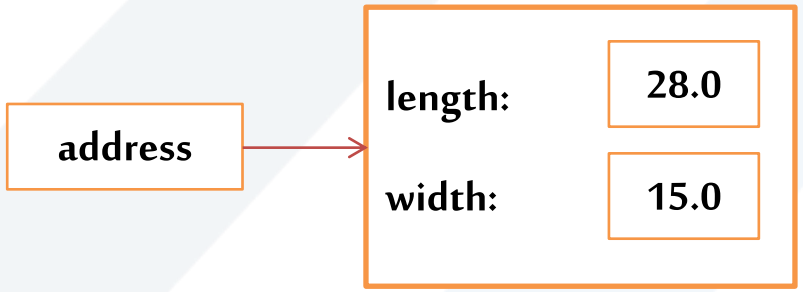
The football variable holds the address of a Rectangle Object.



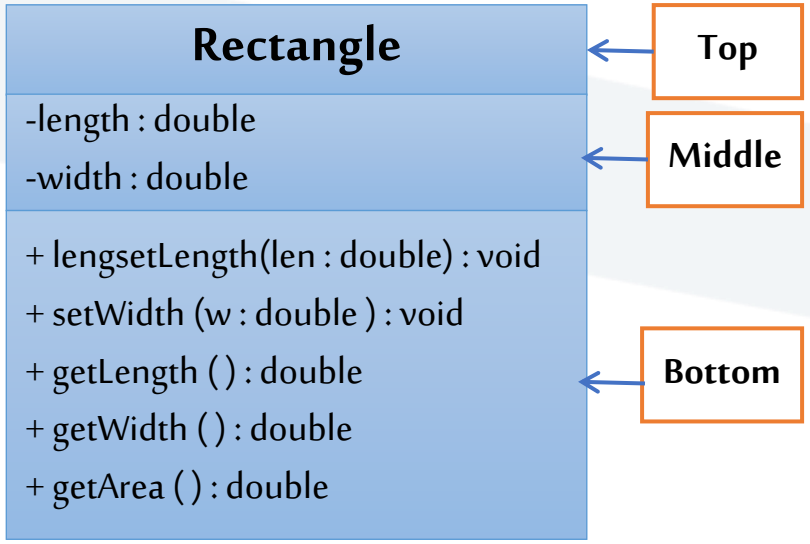
The Volleyball variable holds the address of a Rectangle Object.



The basketball variable holds the address of a Rectangle Object.



UML class diagrams 1



- يستخدم مخطط UML (Unified Modeling Language) للصنف لتلخيص صفات ومناهج الصنف.
- 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 الصنف بمستطيل يحوي ثلاث مكونات compartments.
- **الجزء العلوي :** يتضمن أسم الصنف بخط عريض وبهامش توسط .
- **الجزء الوسطي :** يتضمن أسماء الحقول وتسبق باشارة – للنوع الخاص، + للعام، و# للنوع المحمي ويليه : وبعدها نوع معطيات الحقل.
- **الجزء السفلي :** يتضمن أسماءالمناهج 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 الأسم للمتغير يلي كل منها : ثم نمطه وتفصل المتغيرات (الاسماء) عن بعضها بعضاً بفواصل عادية.
  - يوضع مابعد القوس الدائري المغلق : ونوع القيمة المعادة وإذا كان لايعيد شيئ يوضع void.
  - الباني يوضع في حال كتابته وفق الأتي: يوضع في بداية المناهج بذكر مابين *«constructor»* ثم اسم صنفة وهو اسمه بطبيعة الحال وبعدها ينطبق عليه ماينطبق على المناهج الأخرى، إلا أنه لن يعيد شيئ 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 constructer");}
        public Rectangle (double l, double w) {length=l;width = w;
                System.out.println("constructer 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
-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

Rectangle box1 = new Rectangle(); Rectangle box2 = new Rectangle(5.0, 10.0);

Class test

```
public class RectangleTest {
        public static void main(String[] args) {
        Rectangle r1= new Rectangle();
        System.out.println("length= "+ r1.getLength() + "width= "+ r1.getWidth());
        r1.setLength(11);
        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

Accont

– name : String
– balance : double

«constructor» Account(name : String, balance: double)
+ deposit(depositAmount : double) : void
+ setName(name : String) : void
+ getBalance() : double
+ getName() : String

CreditCard

-customer : String
-bank : String
– account : String
– limit : int
balance : double

+ getCustomer( ) : String
+ getBank( ) : String
+ charge(price : double) : Boolean
+ makePayment(amount : double)
+ getAccount( ) : String
+ getLimit( ) : int
+ getBalance( ) : double

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();

Displaying with print, println

Performing output 1

- 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.
- المنهج print() يوجه الكمبيوتر لتنفيذ إجراء - عرض الأحرف الموجودة بين علامات الاقتباس المزدوجة (لا يتم عرض علامات الاقتباس نفسها) أو البارامترات الموجوده بين القوسين (يمكن للبيانات ان تتضمن تعبير) لايمكن أن تكون عدد ونص عندها يجب استخدام (+) للقسر، وسيتم ترك مؤشر الاخراج حيث انتهت الطباعة.
- عندما تكتمل مهمته المنهج System.out.<u>println()</u>، فإنه ينقل مؤشر الاخراج حيث انتهت الطباعة إلى بداية السطر التالي.

Performing output 2

Escape sequence

- Escape

sequence	Description
\n	Newline. Position the screen cursor at the beginning of the next line.
\t	Horizontal tab. Move the screen cursor to the next tab stop.
\r	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.
\\	Backslash. Used to print a backslash character.
\"	Double quote. Used to print a double-quote character. For example, System.out.println("\"in quotes\""); displays "in quotes".

Some common escape sequences.

Displaying with printf

Performing output 3

- **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
```

- يحتاج استدعاء المنهج printf() ثلاث وسطاء أو بارامترات، يتم وضعها في القائمة مفصولة عن بعضها البعض بفواصل. تبدأ محددات التنسيق بعلامة النسبة المئوية (%) متبوعة بحرف يمثل نوع البيانات ثم البيانات (يمكن للبيانات ان تتضمن تعبير).

(على سبيل المثال ، محدد التنسيق s % هو عنصر يدل لسلسلة بحروف صغيرة ، S% بحروف كبيرة، d % عنصر يدل لقيمة عدد صحيح، f% عنصر يدل لقيمة حقيقية، محدد التنسيق، .%,20.2f علامة التنسيق الفاصلة (,) يشير إلى أنه يجب إخراج قيمة النقطة العائمة بفاصل تجميع كل ثلاث ارقام بمجموعة، مؤلف 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 الأحرف التي كتبها المستخدم على لوحة المفاتيح كونها وحدة الإدخال القياسية.

number = input.nextInt();

Performing input 3

```
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 و عند الإدخال يضاف لنهاية العدد f و nextLine لادخال شريط محرفي

و;char s= input.next().charAt(0) لإدخال واسناد المحرف الأول للمتغير s.

;char str2=in.next().charAt(3)سيتم إدخال أربعة محارف وإسناد المحرف الرابع للمتعير str2.

The Default Constructor

- 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 آخراً مكتوباً.
- 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 5.0, 10.0

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

OR

```
public Rectangle() { length = 1.0;width = 1.0; }
```

عندها يمكن أن نشتق كائن

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

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 اكثر من واحده مما سبق.
- يطلق عليه طريقة التحميل الزائد، وهذا ينطبق أيضا 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

BankAccount
-balance:double -accountName: String
+BankAccount() +BankAccount(double , string) +BankAccount(double): +deposit(amount:double):void +deposit(str:String):void +withdraw(amount:double):void +withdraw(str:String):void +setBalance(b:double):void +setBalance(str:String):void +getBalance():double

Static Class Members

- متى نحتاج لاعضاء الصنف (مناهج وحقول) أن تكون static ثابته؟
- عندما نرغب بأن تتبع هذه الاعضاء للصنف وليس للكائن فقط، أي عندما يكون المنهج خدمي(مثل توابع الرياضيات غير المعرفة في Math) وعندما يكون الحقل تابع لكل الكائنات (مثلاً عد الكائنات المشتقه من صنف) أي حقل شامل Global.
- عندما نحتاج مناهج فقط لتقوم بعمل خدمي ولانرغب بتخزين النتائج، أي نحن لانرغب بتخزين معطيات عن هذه الكائنات فلا داعي لانشائها، ونوفر أماكن في الذاكرة كانت ستخصص للكائنات.
- لاستدعاء طريقة ثابتة أو استخدام حقل ثابت يتم استخدام اسم الصنف، بدلاً من اسم الكائن.

مثال استدعاء منهج الجزر التربيعي من الصنف Math وإعطائها القمة 25.

- **Example:**

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

Class name

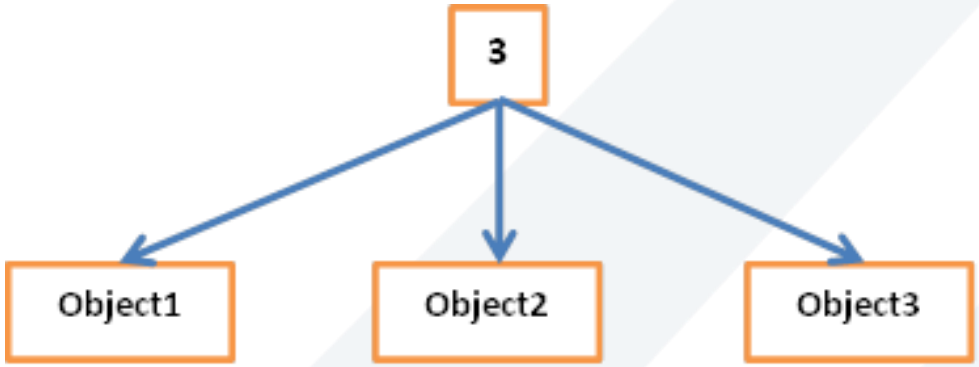
Static Method

Static Fields

- يتم الإعلان عن حقول الصنف بأنها ثابته باستخدام الكلمة المفتاحية static بين محدد الوصول ونوع الحقل وكمثال لعد الكائنات المشتقة من صنف نضع *instanceCount = 0* ضمن الصنف ونزيد هذا الحقل بمقدار 1 في كل باني للصنف *private static int instanceCount = 0;*
- تتم تهيئة الحقل العددي إلى 0 مرة واحدة فقط ، بغض النظر عن عدد مرات إنشاء الصنف.
  - يتم تهيئة الحقول الثابتة الأولية إلى 0 إذا لم يتم إجراء تهيئة لها *initialization*.

instanceCount field

(static)



Static Methods

- يتم الإعلان عن طريقه ثابتة من خلال وضع الكلمة المفتاحية static بين معدِّل الوصول ونوع القيمة المعاده للطريقة.

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

- عندما يحتوي صنف على طريقة ثابتة ، ليس لاداعب لإنشاء مثيل للصنف من أجل استدعاء الطريقة، بفرض أن الطريقة *add* ضمن الصنف *Calc*.

```
System.out.println(Calc.add(121, 212));
```

- تعتبر الطرق الثابتة ملائمة لأنها قد يتم استدعاؤها على مستوى الصنف.
- تُستخدم عادةً لإنشاء فئات أدوات مساعدة utility classes، مثل فئة الرياضيات Math في مكتبة **Java** القياسية.
- لا تتواصل الطرق الثابتة مع حقول غير الثابتة، بل فقط مع الحقول الثابتة، إنما تتواصل الطرق غير الثابتة مع الحقول الثابتة.

Static Methods

```
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;}
}
```

Static Methods

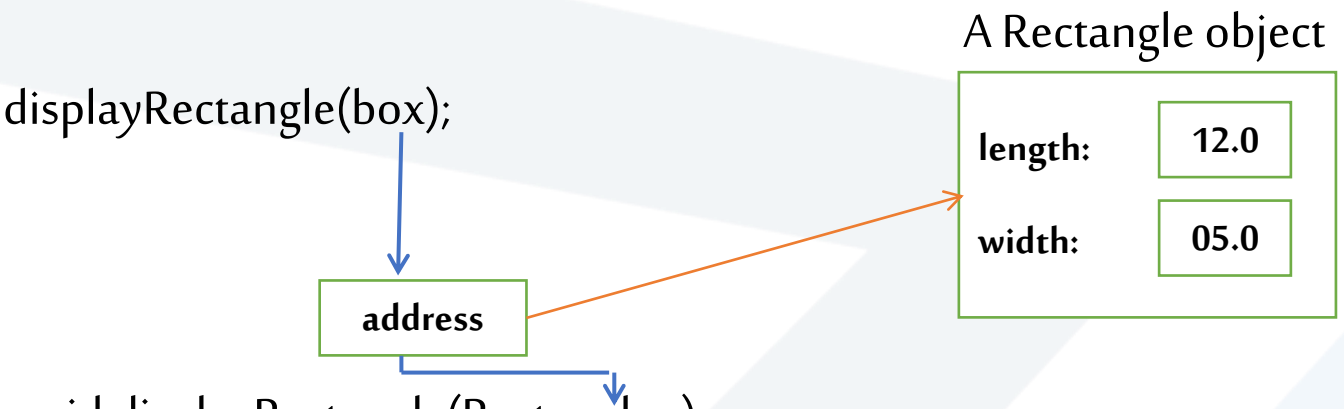
```
public class CalcTest {
    public static void main(String[] args) {
        Calc c1 =new Calc(10,20);Calc c0 =new Calc(); Calc c2 =new Calc(23,17);Calc c3 =new Calc();
        Calc c4 =new Calc(); System.out.println("th mult   is0 = "+c0.mult());
        System.out.println("th mult   is = "+c1.mult());
        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).
- المتغير المرجعي reference type هو عنوان الكائن في الذاكرة.
- لا يتم تمرير نسخة من الكائن ، بل مرجع الكائن فقط.
- عندما تتلقى طريقة متغير مرجعي كوسيطة ، فإن أية تعديل للكائن ضمن الطريقة تعديل محتويات الكائن المشار إليه بواسطة المتغير، لإن العمل يتم على نفس المكان المحجوز في الذاكرة.

Passing Objects as Arguments




```
public static void displayRectangle(Rectangle r)
{
 // Display the length and width.

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

 }
```

Returning Objects From Methods

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

A BankAccount Object

balance: 3200.0

address

```
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.");
```

only the addresses of the objects are compared.

Methods That Copy Objects

- هناك طريقتان لنسخ كائن.

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

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

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

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

```
public Rectangle(Rectangle r2)
    { length = r2.length; width = r2. width ; }   // end Create  copy constructor
    Rectangle  r1= new Rectangle(13, 9);          // Create r3, a copy of r1
    Rectangle  r3 = Rectangle (r1);
```

انتهت محاضرة الأسبوع 2