// Constructors

Properties: (var. or constants)
behavior: (methods)

```
Public class Rectangle {
    int length = 20;
    int width;

    int getArea() {
        return length * width;
    }

    Public Rectangle () {
        width = 5; length = 10;
    }

    Public static void main ( ) {

        Rectangle r = new Rectangle();
        System.out.println( r.getArea());
```

- Properties
- behavior
- constructor
- dot
- length ← 50

Methods in a Secondary class can access properties without passing them as an args.

* it can't access vars in other methods

* مثال على overloading ←

```
Rectangle () {
        length = 10;     Width = 5;
}

Rectangle (int newLength , int newWidth) {
        length = newLength;
        Width = newWidth;
}
```

Property's ← معرفة

في ال main

```
Rectangle r1 = new Rectangle();
Rectangle r2 = new Rectangle(7, 4);

Println(r1.getArea);
Println(r2.getArea);
```

→ 50
→ 28

* مثال اعدّل من Driver على ال length في class Rectangle
~~Static لشرط ان تكون~~

//(in Rec):

```
static int length = 10;
```

//(in Driver):

```
Rectangle.length = 100;
```
← عم اعدّل (static)

static ال ... وبتغير بجميع ال

* اذا Property هي Static يفضل خطأ تحتها في UML

Random class :

```
Random r = new Random();
for(int i=0; i<5; i++){
    System.out.println(r.nextInt());
}
```

* في حال كتبنا r.nextInt() ← اعداد صحيحة خطأ
* ~ ~ ~ r.nextDouble() ← اعداد عشرية
* ~ ~ ~ r.nextBoolean() ← true, false

* في حال r.nextInt(100) ← رقم عشوائي صحيح الى 99

لكن هكذا ✗ احصل على نتائج عشوائية غير متكررة

Random r = new Random(اضع رقم معين); ← اكل
وعندها ✗ كلما وضعت الرقم نفسه، سيعطي نفس الارقام التي اعطاها اول مرة.

Seed

للتقريب الى عدد صحيح
Math.round()

للتقريب الى خانتين عشريتين
Math.round(x*100)/100.0

Date class:

Date d1 = new Date();
التاريخ الحالي ← no argument
System.out.println(d1);

Date d2 = new Date(سنة , شهر , يوم);
1900 ← العدد المطروح منها هو
ex → 2000 is 100
1800 is -100
شهر: 0 - 11
ex 4 is May
ex 3 is April
يوم: يبدأ من الرقم (1) عادي

Date dd = new Date(120, 12, 20);
ex System.out.println(dd);
→ 2021, 1, 20
اذا اضفت خانة، تحول على اللي بعدها

Date dd = new Date(112, 10, 31);
ex Print(dd);
→ 2012, 12 شهر, 1

✗ allowed methods

d.setYear();
d.setMonth();
~ Minutes();
~ Seconds();

example

Puplic Static Void main(String [] args){

طريقة ① لارسال object   Date bDate = new Date(90,10,22);

Employee E = new Employee(111, new Date(), bDate);

}

طريقة ② لارسال Object ↑

① ننشئه مسبقاً ونرسل الاسم
② ننشئ اثناء الارسال
* اعتبرت تاريخ التوظيف هو اللحظة الحالية

Class Employee :

```
public class Employee{

  int Id;
  Date hireDate;
  Date birthDate;

  Public Employee(int anId, Date aHireDate, Date aBirthDate){

    Id = anId;
    hireDate = aHireDate;
    birthDate = aBirthDate;

  }
```

Visibility Modifiers for Data fields, methods constructors ال and classes

Public : Can see from any class
Private : 〃 〃 〃 own class
default : can see in the same package, can't see in other Packages (package - private)
no private class *

in UML : public (+) , private (-)

* اذا انا في كلاس (2) ، وكان public (1) كلاس ويحتوي على اشياء private

⇐ استطيع من (2) ان اصل (1) لكن لا اصل private

* لو انا في (2) package مختلفة وكان (1) ~~private~~ default ويحتوي على public

ليس في نفس package: لا اصل الى ال public لعدم وصولي الى (1)

Package P1;	Package P1	Package P2,
Class C1 { 〃 }	public class C2 { can access C1 〃 〃 C3 }	Public class C3 { Can access C2 Can't 〃 C1 }

Can access C2
~~Can't acces~~
Can access C3

مثال / كلهن مع بعض

```
Package p1;

Public class C1 {
  Public int x;
  int y;
  Private int z;

  Public Void m1() { }
  void m2() { }
  Private Void m3() { }
}
```

```
Package p1;

Public class C2 {
  Void aMethod {
  C1 O = new C1();
  from here:
  can access O.x
  "      "   O.y
  Can't  "   O.z

  can invoke O.m1()
  "      "   O.m2()
  Can't  "   O.m3()
}
```

```
Package P2;

Public class C3 {

Void aMethod() {
  C1 O = new C1();
    Can acces O.x
    Cant  "   O.y
    Cant  "   O.z

    Can invoke O.m1()
    Can't  "   O.m2()
    Cant   "   O.m3()
```

(*) حالة خاصة ، اذا كان عندي كلاس (C1) وبداخله Private
مسموح بانشاء object من نوع (C1) داخل (C1)
عادي Private لانه ما ازل اعتبر داخل نفس الكلاس

ex output ?

```
int x = 4, y = 3;
Print ( (x+y) );
```
→ 7

```
Print ( x+y );
```
→ 7 ; it is not a string

```
Print ("in" + x+y);
```
→ in43

```
Print ("in" + (x+y));
```
→ in7

Data Field Encapsulation

if the (Property, method, ...) is private, it can't be seen from other class

* ~~if can't be edited ever from its class~~

* Getter(accessor) and Setter(mutator) methods + Private
  - Getter → can see Private Values
  - Setter → can edit Private Values (in some conditions which you choose)

naming:
- setProperty
- getProperty

footer: → is Property

Note if the Property is Private

كانت Private
↙

Public constructor(String name, int Id){

* ← عادي عندها نقدر نحطها كثير Constructors

```
Public class Teacher{
    Private String name;          // property
    Private int id;

    Public Teacher(String aName, int anId){      // constructor
        name = aName;
        setId(anId);      → حتى نختصر هنا مباشرة
    }

    Void setId(int newId){        // set
        if (newId >= 1000 && newId < 10 000)   // شرط معين
            id = newId;
        else
            id = 1111;
    }

    int getId(){                  // get
        return id;
    }
```

Passing objects to method

Pass by Sharing

in main

```
}
    Teacher T1 = new Teacher("Ali", 3333);
    Teacher T2 = new Teacher("Rami", 2222);

    Teacher max = maxTeacher(T1, T2);
    System.out.println(max.getId());
}
```

↑ Private لان

طبع في main ان الحين ي عادة
لانه لا يوجد access على properties
← لازم نطبع صفة مرة بالـ get

method

(type) (name) (args)

```
Public static Teacher maxTeacher(t1, t2) {
    if (t1.getId() > t2.getId())
        return t1;   (object)
    else
        return t2;
}
```

output → 3333

---

additional case:

in main

```
System.out.println(T1.getId());
~ ~ ~ (max.getId());
```

in method

```
t1.SetId(1111);
if (),
```

نضيف

output → 1111 pointers لان

→ 2222 بصير على الاصل

Array of object:

~~it is~~ 1D Array of object is similar to 2-D array of int.

Teacher [] teachers = new Teachers[3];

لم نعرّفها بعد

(*) teachers[0] = new Teacher ("Ali", 1111);

وهكذا

ex System.out.Println(teachers[0].getId());
→ 1111

لو لم أكتب المحاذي لـ (*) ، وطلبت الـ id
⇒ null.pointer.exception
ولا ينفذ ما بعدها (ينفذ ما قبلها)

Immutable classes/object

* يعني غير قادر على أي أتعدل على أي Property

*Note: String class is immutable

الخطوات
1) All properties are private
2) No setters functions.
لا يقبل التعديل قدره لا constructor
هنا نقصد به بعد انشاء الـ object و بعد تمرير المعلومات من خلال الـ constructor [ليس القوة السابقة]

الأمر لا بد منهما

3) Getters functions don't return mutable type
it returns only Immutable (like simple types) int, char
or like (String or any ~~object~~ Immutable object)

عدم وجود الخاصية او الخطوة رقم (3) ، تسمح لي بالتعديل هكذا :

لنفرض ان عندي من Class Teacher

Date لـ → Teacher t = new Teacher();

الطريقة Date x = t.getBirthDate();

② x.setYear(100);

Print(t.getBirthDate());

output → 2000, month, ~~day~~

هذا تعديل

* هنا فكرة Pointer ، لان x اعطاني العنوان الى القيمة والتعديل عليها

حل المشكلة We copy the Value then I return the copy, so : I can't access the real Value from the Value returned

Note Pointer as, لان ← two objects بين

Copy طبعن ←

← اي تعديل على اي منهما ، لا يؤثر على قيمة الآخر

* Scope rules for Fields + method var.

* ① اي ترتيب صحيح بين Property و الـ constructor و method

إلا بحالة واحدة : ان يكون احدها يعتمد على الآخر بالترتيب

ex
```
int x = 10;
int y = x+2;
```
← Property

↑ الترتيب مهم

* أمّا داخل كل constructor method ← الترتيب مهم

---

* this Keyword

1- hidden field access
2- constructor calling

لاستخدام ①

```
private String name;
private int id;

Teacher (String name, int id) {
    this.name = name;
    this.id = id;
}
```

this.id ← تبعت الـ object

id ← القيمة الجديدة

name, id ← نفس الاسم

this ← كل واحد له two objects ممكن يكون

استخدام
(2)

عندي أكثر من constructor

```
Public Teacher(String name, int id, Date birthDate){

 this.name = name;
 this.id = id;
 this.birthDate = birthDate;
}

Public Teacher() {
  this("Ahmed", 1111, new Date(100, 5, 20));
}
```

↑ بالأصل عند تعريف default values

← هو يرسل للـ constructor الأول ويعبي القيم

(لأنه الـ default هي قيم واحدة، اما هنا ممكن يكون عندي كذا حالة

⇐ بالتالي ممكن احط أكثر من حالة وليس حالة واحدة عند default

```
Public Teacher(int id){

  this("Khaled", id, new Date());
}

Public Teacher(int id){
  this("Ahmad", id);
}
```

(ممنوع احتفظ بالأخير مباشرة) يرسل لهذا

```
Public Teacher(String name, int id){
  this(name, id, new Date());
}
```

يرسل للأول

```
Public Teacher(int id){
  this("Ahmad", id);
  birthDate = new Date();
```

calling ل constructor لازم يكون اول سطر this - اللازم

Note: Calling constructor في this ممكن اكتب جملة اخرى بعد شرط لازم this اول سطر داخل الـ constructor.

```
Public Teacher(
    int id ) {

this("Ahmad", id);
this.birthDate = new Date();
```

```
Public Teacher(int id) {

this("Ahmad", id, new Date());
```

هكذا فقط، بشرط وجود constructor اخر يستقبل (name, id) ويضعها

```
this.name = name
this.id = id
```

يكونان داخل الـ cons. الذي استدعيناه.

birthDate = new Date(); → بينما هذه ستكون في الـ constructor الذي استدعيناه