محاضرات العملي لمقرر

البرمجة الشيئية

ITGS211

الدرس العملي رقم (6)

إعداد الأستاذة :ملاك ددش

الوراثة في جافا

مفهوم الوراثة في جافا:

في جافا, الكلاس يمكنه أن يرث من كلاس آخر, و بالتالي يحصل على الدوال و المتغيرات الموجودة في هذا الكلاس.

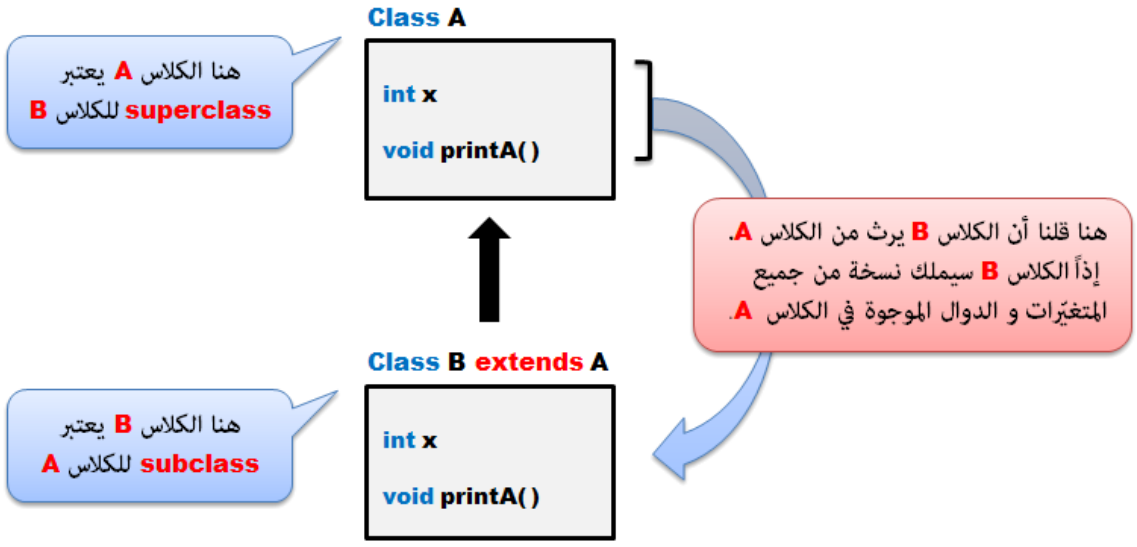
فكرة الوراثة (Inheritance) بسيطة, لكن فائدتها قوية جداً. فمثلاً إذا كنت تريد إنشاء كلاس جديد و لاحظت أنه يوجد كلاس جاهز يحتوي على كودات قد تفيدك يمكنك استغلالها بدل كتابتها من الصفر, أي يمكنك جعل الكلاس الذي قمت بتعريفه يرث هذا الكلاس, و بعدها يمكنك إستخدام جميع المتغيرات و الدوال التي ورثها الكلاس الجديد من الكلاس الجاهز.

مفهوم Superclassو Subclass:

- الكلاس الذي يرث من كلاس آخر يسمى Subclassو يسمى أيضاً derived class,extended class , أوchild class
- الكلاس الذي يورث محتوياته لكلاس آخر يسمى Superclassو يسمى أيضاً base classأو parent class

مثال

الآن لنفترض أننا قمنا بتعريف كلاس إسمه Aيحتوي على متغير إسمه xو دالة إسمها printA(). بعدها قمنا بإنشاء كلاس جديد فارغ إسمه Bو قلنا أنه يرث من الكلاس A. إذاً هذا يعني أن الكلاس Bأصبح يملك نسخة من جميع المتغيرات و الدوال الموجودة في الكلاس A



ال Subclass يرث كل شيء موجود في ال Superclass بشكل تلقائي ما عدا الكونستركتور.

مع العلم أنه يمكن استدعاء كونستركتور ال Superclass من ال Subclass بواسطة الكلمة `super` ال

الكلمة extends في جافا :

الكلمة extends تستخدم لجعل الكلاس يرث من كلاس آخر.

مكان وضع الكلمة extends :

نضع الكلمة extends بعد إسم الكلاس, ثم نضع بعدها إسم الكلاس الذي نريد الوراثة منه.
الكود التالي يعني أن الكلاس B يرث من الكلاس A.

مثال

```
class A {

}

class B extends A {

}
```

إنتبه: في حال كنت تحاول الوراثة من كلاس غير موجود سيظهر لك الخطأ التالي: java.lang.ExceptionInInitializerError

مثال

A.java

```
public class A {

    public int x;

    public void printA() {
        System.out.println("I am from class A");
    }

}
```

```
// هنا قلنا أن الكلاس B يرث المتغيرات و الدوال الموجودة في الكلاس A
public class B extends A {

    // إذاً أي كائن من الكلاس B سيحتوي المتغيرات و الدوال الموجودة في الكلاس A

}
```

```
public class Main {

    public static void main(String[] args) {

        // هنا قمنا بإنشاء كائن من الكلاس B لنتأكد إذا كان يحتوي على الأشياء الموجودة في الكلاس A أم لا
        B b = new B();

        // هنا قمنا باستدعاء الدالة printA() التي ورثها الكلاس B من الكلاس A
        b.printA();

        // و بما أن الكائن b يملك متغير إسمه x أيضاً, يمكننا إعطائه قيمة و عرض قيمته
        b.x = 123;
        System.out.println("x: " +b.x);

    }

}
```

سنحصل على النتيجة التالية عند التشغيل.

```
I am from class A
x: 123
```

الكلمة superفي جافا :

الكلمة super تستخدم للأهداف التالية:

- **للتمييز بين الأشياء (المتغيرات و الدوال) الموجودة في الـ Superclassو Subclassفي حال كانت الأسماء مستخدمة في كلا الكلاسَين.**
- **لإستدعاء الـكونستركتور الموجود في الـ Superclass.**

إذاً الكلمة super تستخدم لإستدعاء الأشياء الموجودة في الـ Superclass.

طريقة استخدام الكلمة super لإستدعاء متغير من الـ Superclass

نضع الكلمة super, بعدها نقطة, ثم نضع إسم المتغير الذي نريد إستدعائه من الـ Superclass

مثال :

```
public class A {

    public int x = 5;

}
```

```
public class B extends A {                                  // هنا قلنا أن الكلاس B يرث من الكلاس A

    public int x = 20;                                      // ن المتغير الموجود في الكلاس A مع وضع قيمة مختلفة له

    public void printBoth() {                               // عند استدعاء هذه الدالة سيتم عرض الأشياء التالية
        System.out.println("x in B contain: " +x);          // هنا سيتم عرض قيمة الـ x الموجودة في الكلاس B
        System.out.println("x in B contain: " +this.x);     // هنا سيتم عرض قيمة الـ x الموجودة في الكلاس B
        System.out.println("x in A contain: " +super.x);    // هنا سيتم عرض قيمة الـ x الموجودة في الكلاس A
    }

}
```

```
public class Main {

    public static void main(String[] args) {

        B b = new B();      // هنا قمنا بإنشاء كائن من الكلاس B من أجل إستدعاء الدالة printBoth() منه

        b.printBoth();      // هنا قمنا باستدعائها

    }

}
```

سنحصل على النتيجة التالية عند التشغيل

```
x in B contain: 20
x in B contain: 20
x in A contain: 5
```

طريقة استخدام الكلمة superلإستدعاء دالة من Superclass

نضع الكلمة ,superبعدها نقطة, ثم نضع إسم الدالة التي نريد إستدعائها من الـ Superclass

مثال :

```
public class A {

    public void print() {
        System.out.println("This is print() method from the class A");
    }

}
```

```
public class B extends A {              // هنا قلنا أن الكلاس B يرث من الكلاس A

    // هنا قمنا بتعريف نفس الدالة الموجودة في الكلاس A لذلك وضعنا @Override قبلها, مع وضع جملة مختلفة في دالة الطباعة
    @Override
    public void print() {
        System.out.println("This is print() method from the class B");
    }

    // هنا قمنا بتعريف دالة مهمتها فقط إستدعاء الدوال الموجودة بداخلها
    public void printBoth() {
        print();                // هنا سيتم إستدعاء الدالة print() الموجودة في الكلاس B
        this.print();           // هنا سيتم إستدعاء الدالة print() الموجودة في الكلاس B
        super.print();          // هنا سيتم إستدعاء الدالة print() الموجودة في الكلاس A
    }

}
```

```
public class Main {

    public static void main(String[] args) {

        B b = new B();      // هنا قمنا بإنشاء كائن من الكلاس B من أجل إستدعاء الدالة printBoth() منه

        b.printBoth();      // هنا قمنا باستدعائها

    }

}
```

سنحصل على النتيجة التالية عند التشغيل

```
This is print() method from the class B
This is print() method from the class B
This is print() method from the class A
```

في حال قام الـ **Subclass** بتعريف دالة كانت أصلاً موجودة في الـ **Superclass** يجب كتابة الكلمة `@Override` قبلها مباشرة, و هكذا سيفهم المترجم أن الـ **Subclass** قام بتعريف الدالة التي ورثها من الـ **Superclass** من جديد.

طريقة استخدامها عند استدعاء كونستركتور:

يمكن استدعاء كونستركتور الـ Superclass من داخل كونستركتور الـ Subclass من خلال الكلمة super.

```
super()                    // عند استدعاء كونستركتور فارغ نكتب هكذا فقط

// أو هكذا

super( parameter )     // عند استدعاء كونستركتور يحتوي على باراميترات, عليك تمرير قيم له.
```

في حال كان الـ Superclass يملك فقط كونستركتور لا يحتوي أي باراميترات (أي مثل كونستركتور إفتراضي), سيقوم المترجم باستدعائه بشكل تلقائي في الـ Subclass حتى لو لم تقم باستدعائه بنفسك.

```
public class A {

    public int x;
    public int y;

    // هنا قمنا بتعريف الكونستركتور الإفتراضي للكلاس A
    // و بما أنه لا يوجد غيره في الكلاس A. سيتم تنفيذه بشكل تلقائي في أي كلاس يرث منه
    public A() {
        x = 50;
        y = 100;
    }

}
```

```
public class B extends A {          // هنا قلنا أن الكلاس B يرث من الكلاس A

    // عند استدعاء الكونستركتور الإفتراضي للكلاس B, أي عند إنشاء كائن منه
    // سيتم إستدعاء الكونستركتور الإفتراضي الموجود في الكلاس A حتى و إن لم نقم باستدعائه

}
```

```java
public class Main {

    public static void main(String[] args) {

        // هنا قمنا بإنشاء كائن من الكلاس B من أجل عرض قيم المتغيرات التي ورثها من الكلاس A
        B b = new B();

        System.out.println("x: " + b.x);
        System.out.println("y: " + b.y);

    }

}
```

سنحصل على النتيجة التالية عند التشغيل

```
x: 50
y: 100
```

مثال :

```java
public class A {

    public int x;
    public int y;

    // هنا قمنا بتعريف الكونستركتور الإفتراضي للكلاس A
    // و بما أنه لا يوجد غيره في الكلاس A. سيتم تنفيذه بشكل تلقائي في أي كلاس يرث منه
    public A() {
        x = 50;
        y = 100;
    }

}
```

```java
public class B extends A {      // هنا قلنا أن الكلاس B يرث من الكلاس A

    public int z;               // هنا قمنا بتعريف المتغير z

    public B() {                // عند استخدام هذا الكونستركتور لإنشاء كائن من الكلاس B
        super();                // سيتم إستدعاء كونستركتور الكلاس A
        z = 123;                // و سيتم إعطاء قيمة للمتغير z
        x = 9;                  // ل قيمة المتغير x في أي كائن من الكلاس B بالقيمة 9, مع الملاحظة أنها ستبقى 50 في أي كائن من الكلاس A
    }

}
```

```java
public class Main {

    public static void main(String[] args) {

        // هنا قمنا بإنشاء كائن من الكلاس B من أجل عرض قيم المتغيرات التي ورثها من الكلاس A
        B b = new B();

        System.out.println("In class B we have:");
        System.out.println("x: " + b.x);
        System.out.println("y: " + b.y);
        System.out.println("z: " + b.z);

        System.out.println();

        // هنا قمنا بإنشاء كائن من الكلاس A من أجل التأكد من أن قيم متغيراته لا تتأثر إذا تم تغيير نفس المتغيرات في أي كلاس يرث منه
        A a = new A();

        System.out.println("In class A we have:");
        System.out.println("x: " + a.x);    // B الموجود في الكلاس x مختلفة عن قيمة المتغير A الموجود في الكلاس x ة المتغير
        System.out.println("y: " + a.y);

    }

}
```

سنحصل على النتيجة التالية عند التشغيل

```
In class B we have:
x: 9
y: 100
z: 123

In class A we have:
x: 50
y: 100
```

و في حال كان الـ Superclass يملك أكثر من كونستركتور, ستكون مجبر على تعريف كونستركتور في الـ Subclass يستدعي أي كونستركتور من الكونستركتورات الموجودة في الـ Superclass.

مثال :

```java
public class A {

    public int x;
    public int y;

    // الكلاس A يحتوي على كونستركتور يسمح بتمرير قيم أولية للمتغيرات x و y عند استدعائه
    // هنا يجب استدعاء هذا الكونستركتور في أي Subclass لأنه لا يوجد كونستركتور لا يحتوي على باراميترات
    public A(int x, int y) {
        this.x = x;
        this.y = y;
    }

}
```

```java
public class B extends A {              // هنا قلنا أن الكلاس B يرث من الكلاس A

    // هنا يجب تعريف كونستركتور واحد على الأقل يستدعي الكونستركتور الموجود في الكلاس A

    // هنا قمنا بتعريف كونستركتور يمرر القيمتين 123 و 456 في كونستركتور الـ Superclass A
    public B() {
        super(123, 456);
    }

    // هنا قمنا بتعريف كونستركتور نمرر له قيمتين عند استدعائه, فيقوم بدوره بتمريرهما في كونستركتور الـ Superclass A
    public B(int p1, int p2) {
        super(p1, p2);
    }

}
```

```java
public class Main {

    public static void main(String[] args) {

        // هنا قمنا بإنشاء كائن من الكلاس B باستخدام كونستركتور الكلاس B الذي لا يأخذ باراميترات
        B b1 = new B();
        System.out.println("Here the constructor generate these values for the object b1:");
        System.out.println("x: " + b1.x);
        System.out.println("y: " + b1.y);

        System.out.println();

        // هنا قمنا بإنشاء كائن من الكلاس B باستخدام كونستركتور الكلاس B الذي يأخذ 2 باراميتر
        B b2 = new B(47, 13);
        System.out.println("Here the constructor generate these values for the object b2:");
        System.out.println("x: " + b2.x);
        System.out.println("y: " + b2.y);

    }

}
```

سنحصل على النتيجة التالية عند التشغيل

```
Here the constructor generate these values for the object b1:
x: 123
y: 456

Here the constructor generate these values for the object b2:
x: 47
y: 13
```