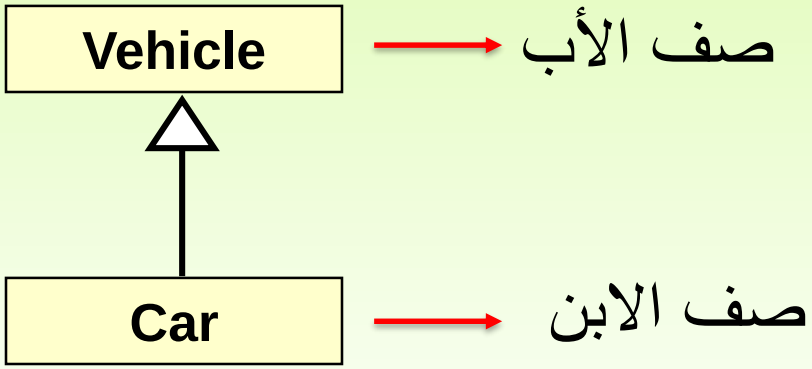
Inheritance in Java
الوراثة في الجافا

Dr. REEMA AL-KAMHA

ما هي الوراثة

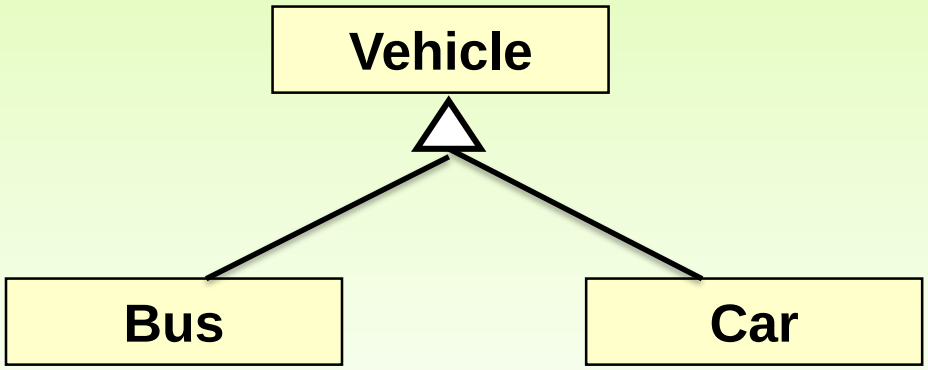
- تسمح لغة الجافا بإعادة استخدام أي صف عن طريق توسيعه، و الحصول منه على صف موسع و بقدرات إضافية.
- تسمى تقنيا إعادة استخدام صف عن طريق توسيعه بالوراثة من الصف أو اختصارا **الوراثة Inheritance**، حيث تسمح بإنشاء صف جديد من صف موجود.
- مفهوم الوراثة بين الصفوف هو أحد المميزات القوية للبرمجة غرضية التوجه
  - لأنها تمكن من إعادة استخدام صفوف مكتوبة مسبقا مما يزيد من إنتاجية المبرمج و يخفف العناء عنه.
  - كذلك من وجهة نظر هندسة البرمجيات، إن إعادة استخدام صفوف يعتبر ميزة لأن ذلك يعفي المبرمج من فحص هذه الصفوف للتأكد من سلامتها و خلوها من الأخطاء البرمجية.
- الفائدة من الوراثة: ***software reuse***

تمثيل علاقة الوراثة في لغة UML



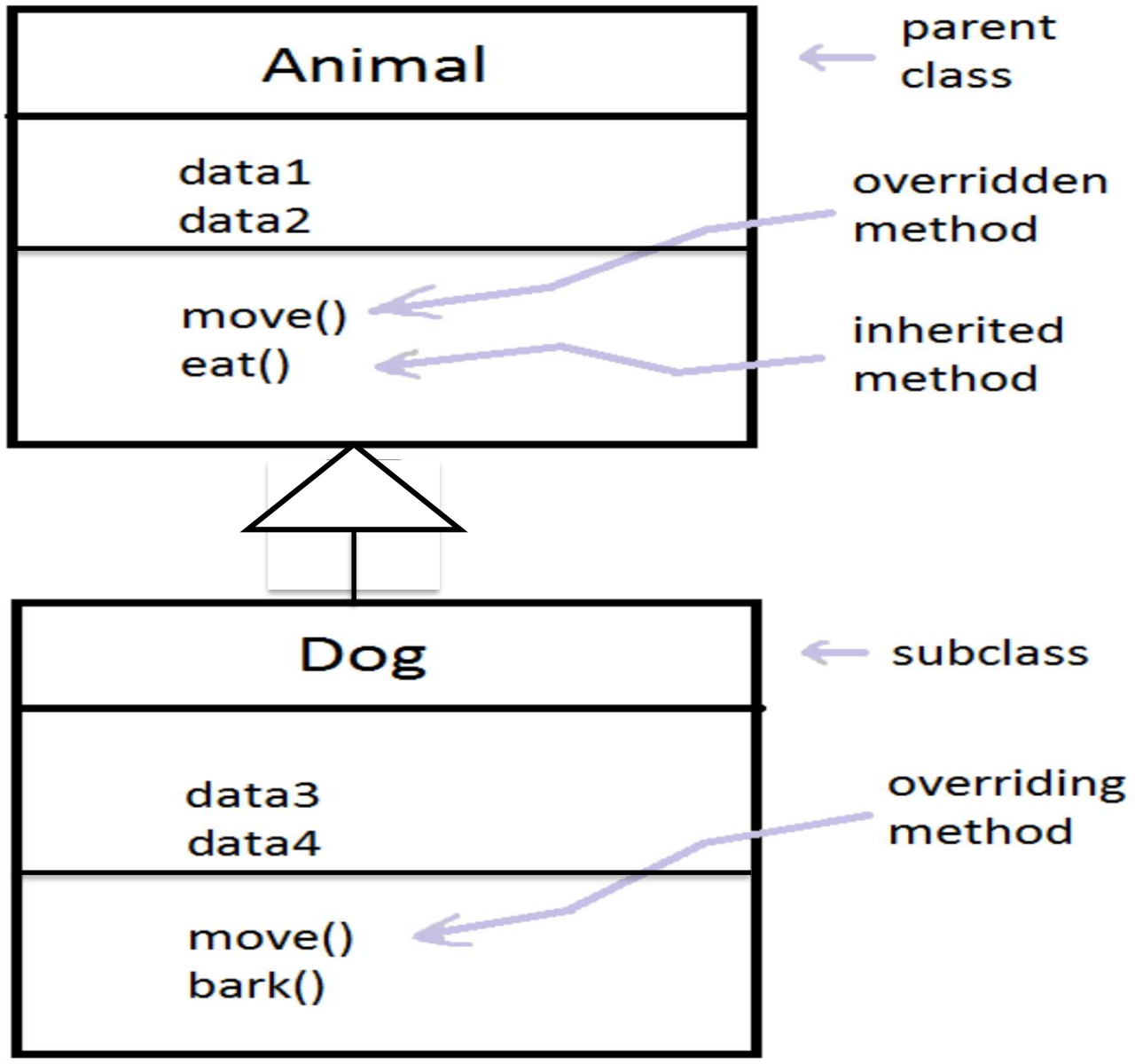
الحاجة لاستخدام الوراثة

من الممكن أن تكون مجموعة من الصفوف تحوي صفات مشتركة، فعلى سبيل المثال السيارة (Car) والباص (Bus) لهما شركة مصنعة وتاريخ تصنيع، فيمكن النظر إليها على أنها وسائل نقل، ووضع الصفات المشتركة (الحقول و الطرق) في صف مشترك مركبة **Vehicle**، ويتم وراثة هذا الصف من هذه الصفوف (السيارة و الباص).



ما هي الوراثة

- ندعو الصف الأب بـ Parent Class أو Superclass أو Base Class
- ندعو الصف الوارث بالصف الابن Child Class أو Derived Class أو Subclass
- يرث **صف الابن Child Class جميع حقول و طرق الصف الأب العامة public و المحمية protected** ، أما **الحقول و الطرق الخاصة private بالأب فلا تورّث**، أي لا يعطيها لصف الابن و هي خاصة به فقط.
- كما يمكن في صف الابن:
  - إضافة حقول جديدة خاصة بصف الابن
  - إضافة طرق Methods جديدة
  - التعديل على طرق Methods موجودة في صف الأب، وتدعى هذه العملية إعادة تعريف الطريقة Method Overriding


Animal
parent
class
data1
data2
overridden
method
move()
eat()
inherited
method
Dog
subclass
data3
data4
overriding
method
move()
bark()

تذكرة

- تعطينا الوراثة إمكانية توسيع صف الأب
- الحقول والطرق الخاصة (private) بالصف الأب لا يتم توريثها
- قد يحوي صف الابن حقول و طرق إضافية خاصة به

تذكرة
محددات الوصول بالنسبة لحقول و طرق الصف

محدد الوصول	الوظيفة
public	إذا ظهر في بداية عبارة التصريح عن حقل أو طريقة فإنه يجعلهما عامين، أي يمكن الوصول إليهما من أي صف آخر
private	إذا ظهر في بداية عبارة التصريح عن حقل أو طريقة فإنه يجعلهما خاصين، ولا يمكن الوصول إليهما من خارج الصف الذي تم التصريح فيه
protected	إذا ظهر في بداية عبارة التصريح عن حقل أو طريقة فإنه يجعلهما محميين، أي يمكن الوصول إليهما من صفوف الأبناء و الصفوف المنتمية إلى الحزمة نفسها
الافتراضي package access	يجعل هذا المحدد الحقول و الطرق عامة بالنسبة للصفوف المنتمية إلى الحزمة نفسها، و لكنها خاصة بالنسبة للصفوف المنتمية إلى مكتبات أخرى. لايوجد كلمة محجوزة لهذا المحدد إذ يكفي أن لا تضع أيا من المحددات السابقة

المؤثر final

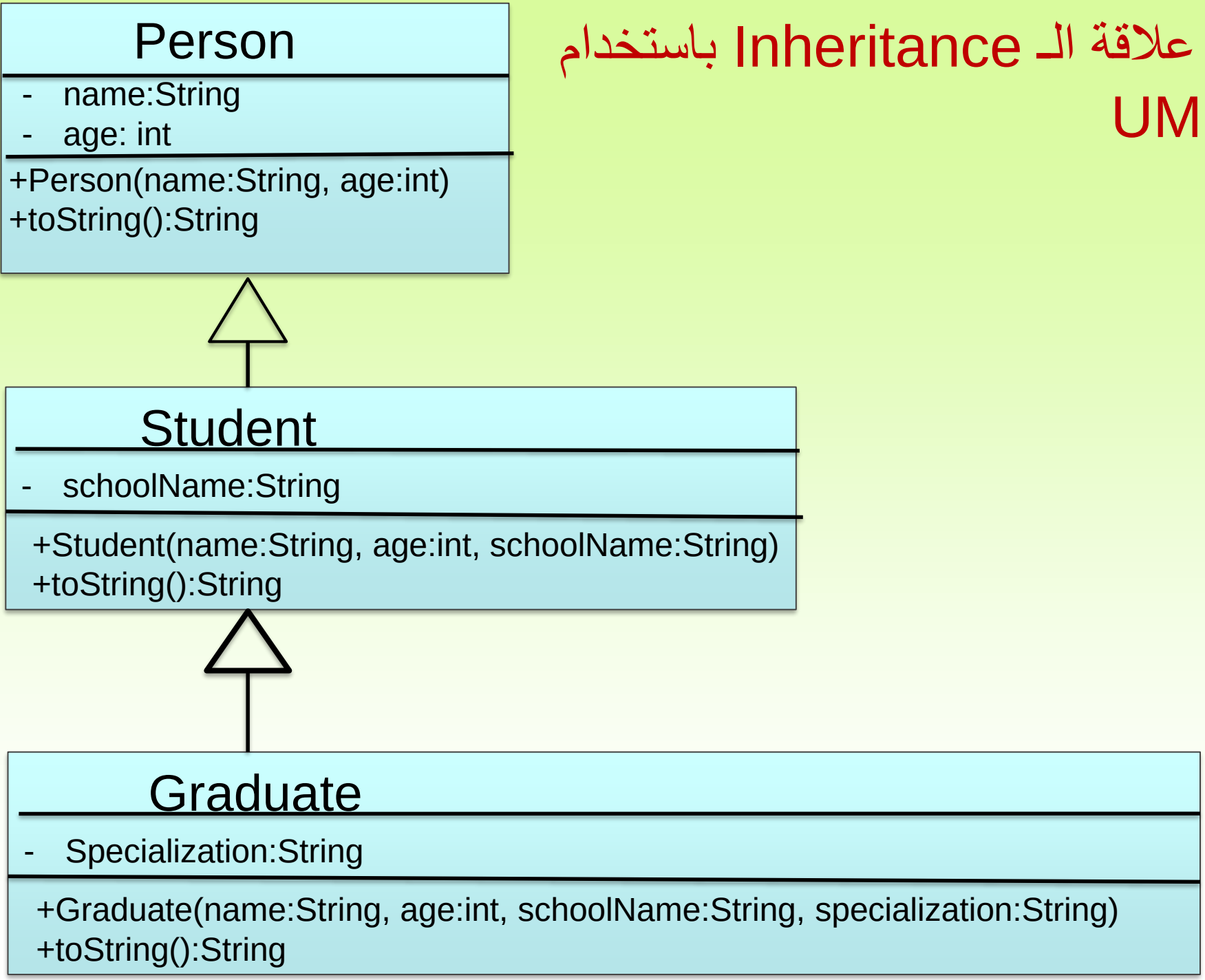
- كل صف في الجافا يحق له أن يكون صف أب إلا إذا قلنا غير ذلك أي يكون final
- **إذا ظهر المؤثر final في عبارة التصريح عن متغير**، فإن هذا المتغير يصبح ثابتا (أي لا يمكن تغيير قيمته أثناء التنفيذ)
- **إذا ظهر المؤثر final في عبارة التصريح عن طريقة method**، فإن هذه الطريقة تصبح ثابتة، فإن الصفوف الأبناء لا تستطيع إعادة تعريف التعليمات البرمجية للطريقة الثابتة
- **إذا ظهر المؤثر final في عبارة التصريح عن صف**، فإنه يجعله ثابتا. كنتيجة لذلك، فإن الصف لا يمكن أن يورث، أي لا يمكن إعادة استخدام الصف الثابت بواسطة الوراثة.

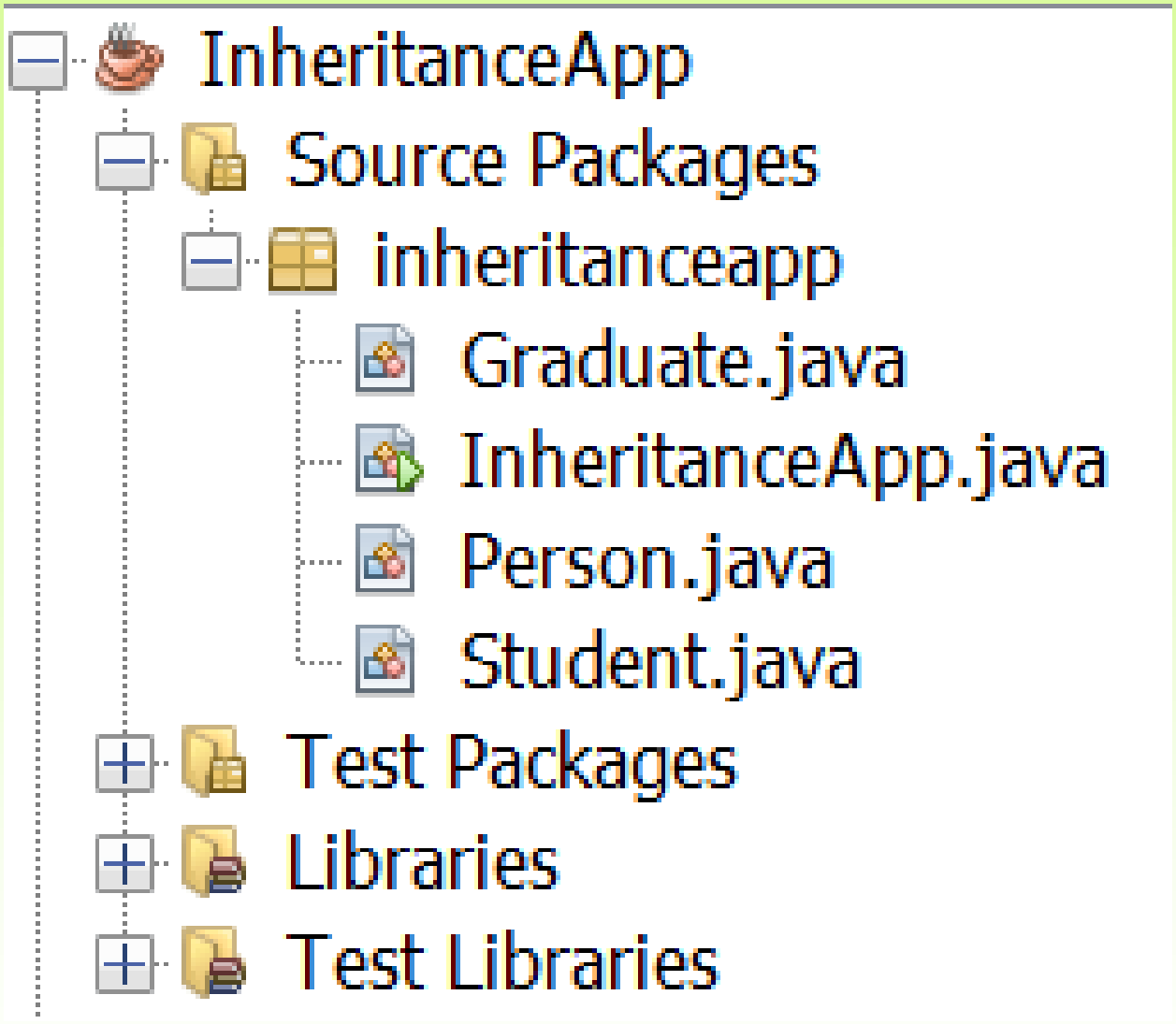
مثال على علاقة الوراثة في الجافا

اكتب برنامجا بلغة الجافا للتوصيف التالي:

- **تعريف صف اسمه Person** له اسم و عمر، وباني مناسب للتوصيف، ويحوي طريقة toString تعطي **اسم الشخص و عمره** و ترجع String
- **تعريف صف Student علما أن الطالب هو شخص و له اسم المدرسة المنتسب لها،** يحوي باني مناسب للتوصيف، إضافة لطريقة toString تعطي **اسمه و عمره و اسم مدرسته**
- **تعريف صف Graduate علما أن المتخرج هو طالب وله اختصاص،** يحوي باني مناسب للتوصيف، إضافة لطريقة toString تعطي **اسمه و عمره و اسم مدرسته و اختصاصه**
- **تعريف صف InheritanceApp** يقوم بإنشاء أغراض من الصفوف السابقة و استدعاء الطرق الموجودة فيها

تمثيل علاقة الـ Inheritance باستخدام لغة UML




InheritanceApp
Source Packages
inheritanceapp
Graduate.java
InheritanceApp.java
Person.java
Student.java
Test Packages
Libraries
Test Libraries

صف الأب Person

```
package inheritanceapp;
 public class Person {
    private String name;
    private int age;
    public Person(String name, int age){
        this.name=name;
        this.age=age;
    }
    public String toString(){
      return "Name: "+name+" Age: "+age;
    }
}
```

الصف Student حيث أن الأب المباشر له هوالصف Person

تستخدم الكلمة المحجوزة extends للتعبير عن وراثة الصف Student للصف Person

```
package inheritanceapp;
public class Student extends Person{
  private String schoolName;
  public Student(String name, int age, String schoolName){
    super(name,age);
    this.schoolName=schoolName;
  }
  @Override
  public String toString(){
    return "Student Information\n"+ super.toString()+
           "SchoolName: "+schoolName;
  }
}
```

Super تشير لصف الأب

استدعاء الصف Student لباني صف الأب Person عن طريق التعليمة super

الطريقة toString() في صف الابن Student أعادت تعريف الدالة toString() في صف الأب Person

صف الأب

استدعاء الطريقة toString() من صف الأب Person في صف الابن Student

الصف Graduate و الذي بدوره ابن للصف Student

```
package inheritanceapp;
public class Graduate extends Student {
  private String specialization;
  public Graduate(String name, int age, String schoolName,
                  String specialization){
    super(name, age,schoolName);
    this.specialization=specialization;
  }
  @Override
  public String toString(){
    return "Graduate Information\n"+ super.toString()+
              " Specialization:"+specialization;
  }
}
```

الصف InheritanceApp

```
package inheritanceapp;
 class InheritanceApp {
    public static void main(String[] args) {
       Person p=new Person("Ali",30);
       System.out.println(p.toString());
       Student s=new Student("Sami", 20,"AlSHAM PRIVATE UNIVERSITY");
        System.out.println(s.toString());
       Graduate g=new Graduate("Rami",25, "Damascus UNIVERSITY","IT");
        System.out.println(g.toString());
    }
}
```

نتيجة التنفيذ

Name: Ali Age: 30
Student Information
Name: Sami Age: 20 SchoolName: AlSHAM PRIVATE UNIVERSITY
Graduate Information
Student Information
Name: Rami Age: 25 SchoolName: Damascus UNIVERSITY
Specialization:IT

ملاحظات

- تستخدم الكلمة المحجوزة extends للتعبير عن وراثة صف لصف آخر، فللتعبير أن الصف Student ابن للصف Person نكتب:

```
public class Student extends Person
```

وللتعبير أن الصف Graduate ابن للصف Student نكتب:

```
public class Graduate extends Student
```

ملاحظات حول الصف Student

- الكلمة المحجوزة super تشير إلى صف الأب Person
- **استدعاء باني الصف الابن Student لباني صف الأب Student عن طريق كتابة التعليمة super(name,age); في أول سطر من أسطر باني الابن Student والتي بدورها تعمل على إعطاء قيم أولية لجميع الحقول الموروثة من الأب Person .**
- **الطريقة toString() في صف الابن Student أعادت تعريف الدالة toString() في صف الأب Person.**
- **لاستدعاء الطريقة toString() من صف الأب Person ضمن صف الابن Student تم عن طريق كتابة التعليمة super.toString()**

ملاحظات حول الصف Graduate

- الكلمة المحجوزة super تشير إلى صف الأب Student
- **استدعاء باني الصف الابن Graduate لباني صف الأب Student عن طريق كتابة التعليمة ;super(name, age,schoolName) في أول سطر من أسطر باني الابن Graduate والتي بدورها تعمل على إعطاء قيم أولية لجميع الحقول الموروثة من الأب Student .**
- **الطريقة toString() في صف الابن Graduate أعادت تعريف الدالة toString() في صف الأب Student.**
- **لاستدعاء الطريقة toString() من صف الأب Student ضمن صف الابن Graduate تم عن طريق كتابة التعليمة super.toString()**

إعادة تعريف الطريقة Method Overriding

- يمكن للصف الابن أن يقوم بإعادة تعريف طريقة overriding لدى الصف الأب، ولكن يجب أن تكون اسم الطريقة ووسطائها و نوع الإرجاع في الصف الابن مطابقاً تماماً للطريقة المراد إعادة تعريفها في الصف الأب.
- **أي هي: إمكانية كتابة طريقة موجودة في صف الأب ضمن صف الابن بحيث يكون لها نفس الاسم و نفس قائمة الوسطاء و نفس نوع الإرجاع و لكن تقوم بعمل مشابه لطريقة الأب أو مختلفة كليا**
- عندما يقوم الصف الابن بتعديل التعليمات البرمجية لطريقة موروثة من الصف الأب فإننا ندعو ذلك بإعادة تعريف الطريقة Overriding.
- تعد القدرة على إعادة تعريف طرق الصف الأب في الصف الابن واحدة من الإمكانيات الكبيرة التي توفرها البرمجة غرضية التوجه
- إذ يمكن رفع أداء الطرق المستخدمة في صف الأب دون إحداث تغييرات فيه.
- يكفي التصريح عن صف جديد (ابن) يرث من الصف الأب ثم يقوم الصف الابن بإعادة كتابة التعليمات البرمجية لبعض أو جميع الطرق التي ورثها. بهذا الشكل يحتفظ الصف الأب بالطرق الأصلية، بينما يمتلك الصف الابن الطرق الموروثة و المعدلة (التي تم إعادة تعريفها).

الهدف من Method Overriding

- الهدف الحقيقي من لإعادة تعريف طريقة **Overriding** هو إتاحة الفرصة لصف الابن أن يعرف طرق حسب حاجته.

شروط الـ Override للطرق

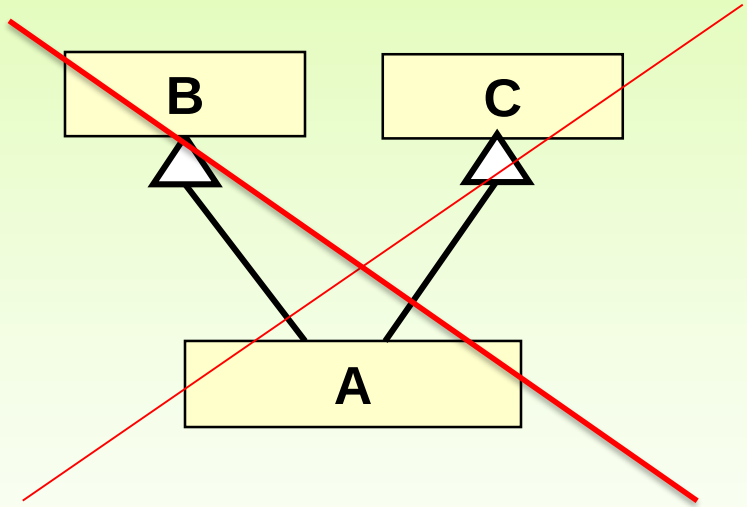
- يجب أن يكون محدد الوصول **Modifier** المستخدم للطريقة الجديدة هو نفسه المستخدم للطريقة القديمة، و يجب أن يكون نوعه **public** أو **protected**
- عدد و نوع وسطاء الطريقة الجديدة يجب أن يطابق عدد و نوع وسطاء الطريقة القديمة.
- نوع الإرجاع للطريقة الجديدة يجب أن يكون نفس نوع الإرجاع لطريقة القديمة.
- الطريقة المعرفة كـ **private** لا يمكن أن نفعل لها **Override**، لأن كلمة **private** تمنع إمكانية الوصول المباشر للدالة من الصف الابن
- الطريقة المعرفة كـ **final** لا يمكن أن نفعل لها **Override** لأن كلمة **final** تمنع تغير محتوى الطريقة بعد تعريفها.
- الطريقة المعرفة كـ **static** لا يمكن أن نفعل لها **Override** و لكن يمكن تعريفها من جديد في أي مكان، لأن كلمة **static** تجعل الطريقة مشتركة بين جميع الصفوف.
- لا يمكن أن نفعل **Override** للباني.

الكلمة المحجوزة super

- يتم استخدامها للوصول إلى الصف الأب
- تستخدم لاستدعاء باني محدد من البواني الخاصة بالصف الأب، ولكن يجب وضعها كأول تعليمة ضمن الباني في الصف الابن وإلا سيحصل خطأ في الترجمة compiler error.
- تستخدم عندما يقوم صف ابن بإعادة تعريف طريقة m من الصف الأب، و ضمن هذه الطريقة يريد الصف الابن استدعاء الطريقة m المعرفة في الأب للإستفادة منها، عندئذ يتم استدعائها عن طريق الكلمة المفتاحية super

لا تدعم لغة الجافا الصف الذي له أبوان مباشران

يمكن للصف الواحد أن تمتلك أي عدد من الأبناء، و لكن لا تسمح لغة الجافا بالوراثة المتعددة، أي لا تسمح للصف الواحد أن يرث أكثر من أب مباشر واحد. على سبيل المثال، التوريث التالي خاطئ لأن الصف A له أبوان اثنان مباشران هما B و C



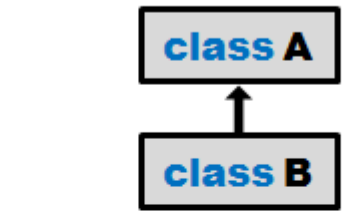
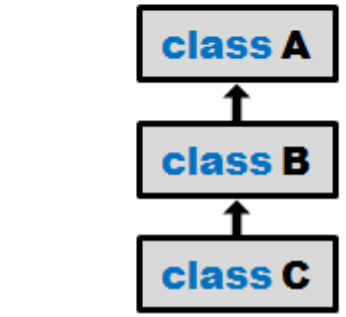
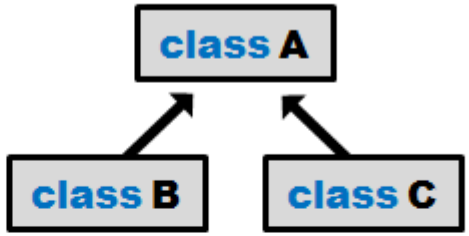
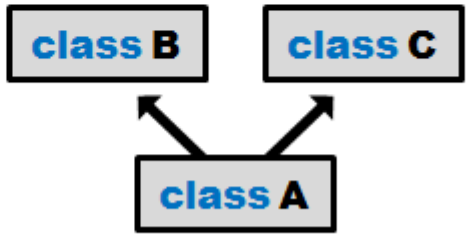
ملاحظات حول البواني

- إذا كان الصف الأب يمتلك أكثر من بان، فإن الصف الابن يستطيع استدعاء أي واحد من هذه البواني.
- لنفرض أن الصف B هو ابن الصف A. إذا أنشأنا غرضا من الصف B باستخدام أحد البواني فيه (في الصف B) فإن ذلك الباني يجب أن يستدعى بان من بواني الصف الأب A و إلا يصدر المترجم خطأ. و هنا نميز عدة حالات:
  - إذا كان الصف الأب A لا يحوي أي باني، عندئذ يقوم مترجم الجافا باستدعاء الباني الافتراضي للأب دون الحاجة لوضع استدعاء صريح (بوساطة super) لهذا الباني.
  - إذا احتوى الصف الأب A باني بدون وسطاء، عندئذ يقوم مترجم الجافا باستدعاء الباني بدون وسطاء للأب دون الحاجة لوضع استدعاء صريح (بوساطة super) لهذا الباني.
  - أما إذا كانت جميع بواني الصف الأب بوسطاء، فإنه يجب أن نضع استدعاءا صريحا لأحد بواني الصف الأب لكي تتم عملية إنشاء غرض من نوع الصف الابن بشكل ناجح

تذكرة

- التعديل على المتغيرات يتم عن طريق تغيير قيمتها
- التعديل على الصفوف يتم بالوراثة
- التعديل على الطرق يتم عن طريق إعادة التعريف

أشكال الوراثة في الجافا

إسمها	شكلها	الكود
Single inheritance		class A { . . . . . } class B extends A { . . . . . }
Multi Level inheritance		class A { . . . . . } class B extends A { . . . . . } class C extends B { . . . . . }
Hierarchical inheritance		class A { . . . . . } class B extends A { . . . . . } class C extends A { . . . . . }
Multiple inheritance		class B { . . . . . } class C { . . . . . } class A ~~extends~~ B, C { . . . . . }

Quick Check

True or False?

A child class may define a method with the same name as a method in the parent.	True
A child class can override the constructor of the parent class.	False
A child class cannot override a `final` method of the parent class.	True
It is considered poor design when a child class overrides a method from the parent.	False

Object Class الصف Object

- هو الصف الأعم في الجافا، فهو أعلى صف أب فيها، و الطرق المعرفة فيه تستخدم في كل صف.

Any object can be cast as an `Object`.

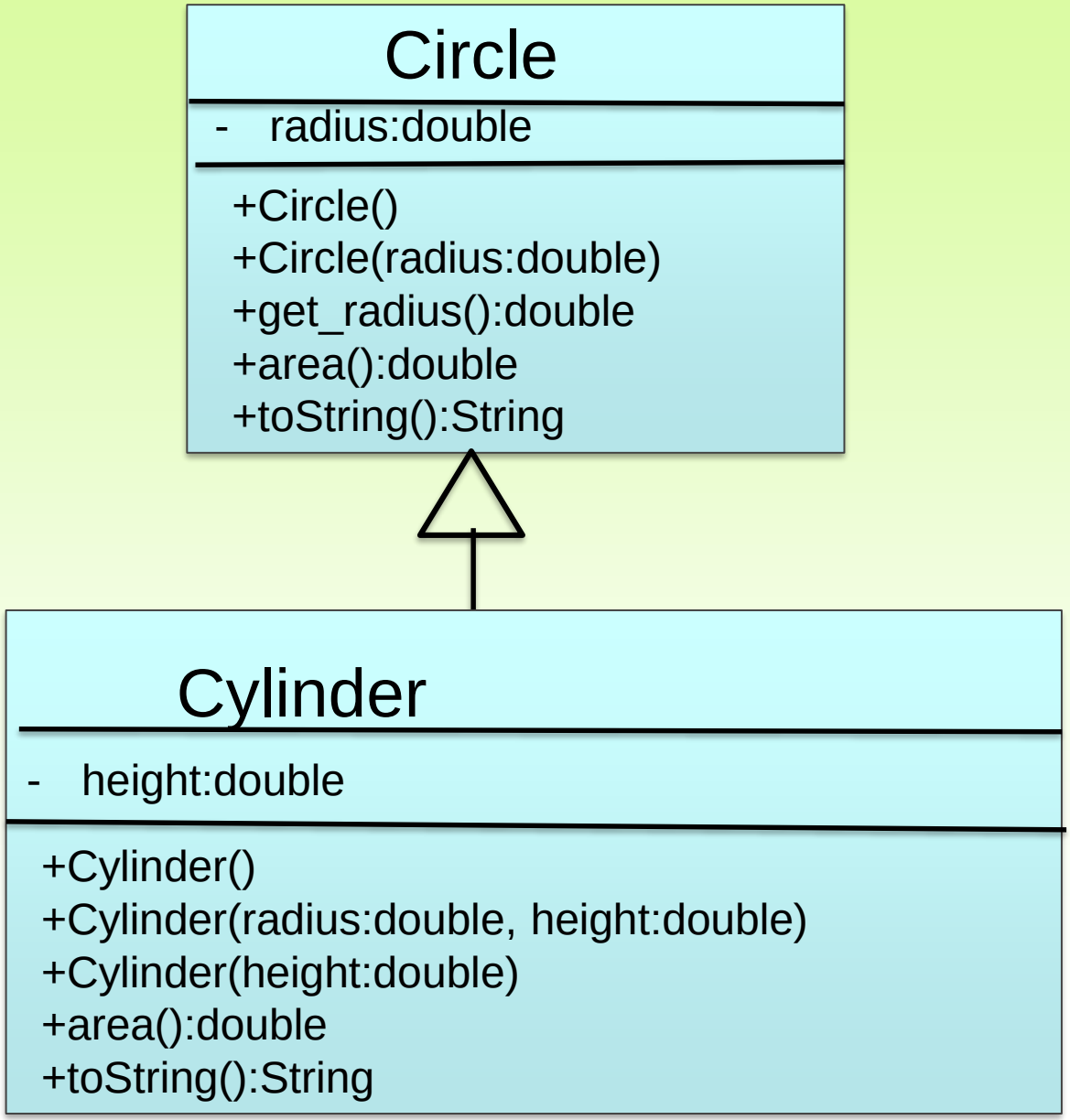
- لإنشاء غرض من الصف Object نكتب:

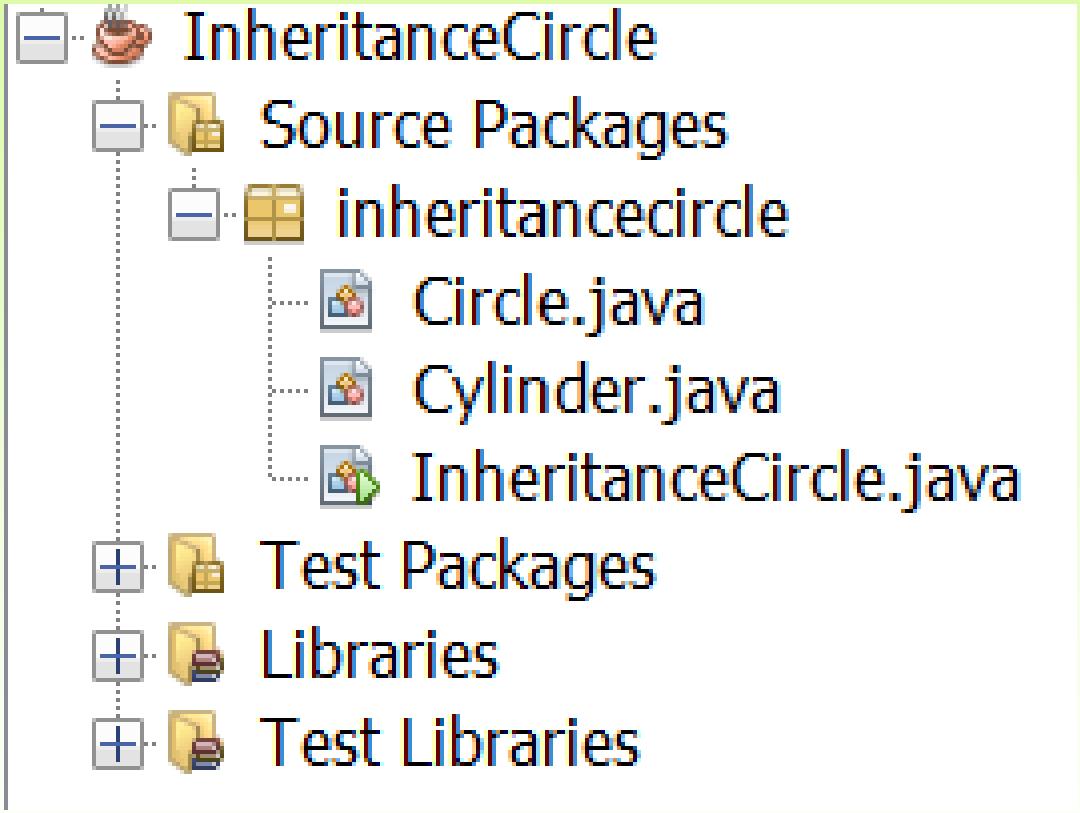
```
Object ob=new Object();
```

البرامج المطلوب قراءتها و تنفيذها و فهمها

- InheritanceApp
- InheritanceCircle
- InheritanceEmployee

تمثيل علاقة الـوراثة لبرنامج InheritanceCircle




InheritanceCircle
Source Packages
inheritancecircle
Circle.java
Cylinder.java
InheritanceCircle.java
Test Packages
Libraries
Test Libraries

صف الأب Circle

```java
package inheritancecircle;
public class Circle {
    private double radius;
    public Circle(){
     radius=0;
    }
    public Circle(double radius){
     this.radius=radius;
    }
    public double get_radius() {
     return radius;
    }
    public double area(){
     return Math.PI * radius*radius;
    }
    @Override
    public String toString(){
     return "The Circle information\n The radius of the circle= "
             + radius+" Area of the circle= "+area();
    }
}
```

صف الأبن Cylinder

```
package inheritancecircle;
public class Cylinder extends Circle {
  private double height;
  public Cylinder(){
  }
  public Cylinder(double radius, double height){
    super(radius);
    this.height=height;
  }
  public Cylinder(double height){
    this.height=height;
  }
  public double area(){
    return 2*Math.PI * get_radius()*height;
  }
  public String toString(){
    return "The Cyinder Information \n radius of cylinder= "
            +get_radius()+" The height = "+height
            +" Area of cylinder= "+area();
  }
}
```

الصف InheritanceCircle

```
package inheritancecircle;
public class InheritanceCircle {
    public static void main(String[] args) {
        Circle r=new Circle(2);
        System.out.println(r.toString());
        Cylinder c=new Cylinder(2,4);
        System.out.println( c.toString());
        Circle r1=new Circle();
        System.out.println(r1.toString());
        Cylinder c1=new Cylinder(4.0);
        System.out.println( c1.toString());


    }
}
```

نتيجة التنفيذ

The Circle information
The radius of the circle= 2.0 Area of the circle= 12.566370614359172
The Cyinder Information
radius of cylinder= 2.0 The height = 4.0 Area of cylinder= 50.26548245743669
The Circle information
The radius of the circle= 0.0 Area of the circle= 0.0
The Cyinder Information
radius of cylinder= 0.0 The height = 4.0 Area of cylinder= 0.0

تمثيل علاقة الوراثة لبرنامج InheritanceEmployee