Polymorphism in Java
تعددية الأشكال في الجافا

Dr. REEMA AL-KAMHA

مقدمة

- **Polymorphism means *many* (poly) *shapes* (morph)**
  - **تعني تعددية الأشكال،** أي القدرة على الظهور بعدة أشكال.
  - في الجافا، تعددية الأشكال تعني وجود **عدة طرق (methods) لها نفس الاسم**.

أنواع تعددية الأشكال Types of polymorphism

- **Compile Time Polymorphism or static binding or early binding**

حيث يتم تحديد الغرض عند ترجمة البرنامج، و يظهر هذا النوع من تعددية الأشكال في التحميل الزائد للدوال Method Overloading، وهي إمكانية كتابة أكثر من طريقة **في نفس الصف** تحمل نفس الاسم على أن نغير في نوع الوسطاء أو عددها أو ترتيبها إذا كانت من أنواع مختلفة.

- **Run Time Polymorphism or dynamic binding, or late binding**

حيث يتم تحديد الغرض وقت التنفيذ البرنامج عوضا عن وقت ترجمته، و يظهر هذا النوع من تعددية الأشكال عند إعادة تعريف الطريقة Method Overriding،
وهي إمكانية إعادة كتابة طريقة موجودة في صف الأب ضمن صف الابن بحيث يكون لها نفس الترويسة (signature)، أي لها نفس الاسم و نفس قائمة الوسطاء و نفس نوع الإرجاع و لكن تقوم بعمل مشابه لطريقة الأب أو مختلفة كليا.

يتم تنفيذ تعددية الأشكال في هذا النوع عن طريق انشاء متغير مرجعي r من الصف الأب و جعله يؤشر على غرض من صفوف الأبناء، و استدعاء الطرق التي تم إعادة كتابتها في الصف الابن من خلال المتغير المرجعيr .

مثال على التحميل الزائد للطرق Overloading

```
class Test {
    public static void main(String args[]) {
        myPrint(5);
        myPrint(5.0);
    }

    static void myPrint(int i) {
        System.out.println("int i = " + i);
    }

    static void myPrint(double d) { // same name, different parameters
        System.out.println("double d = " + d);
    }
}
```

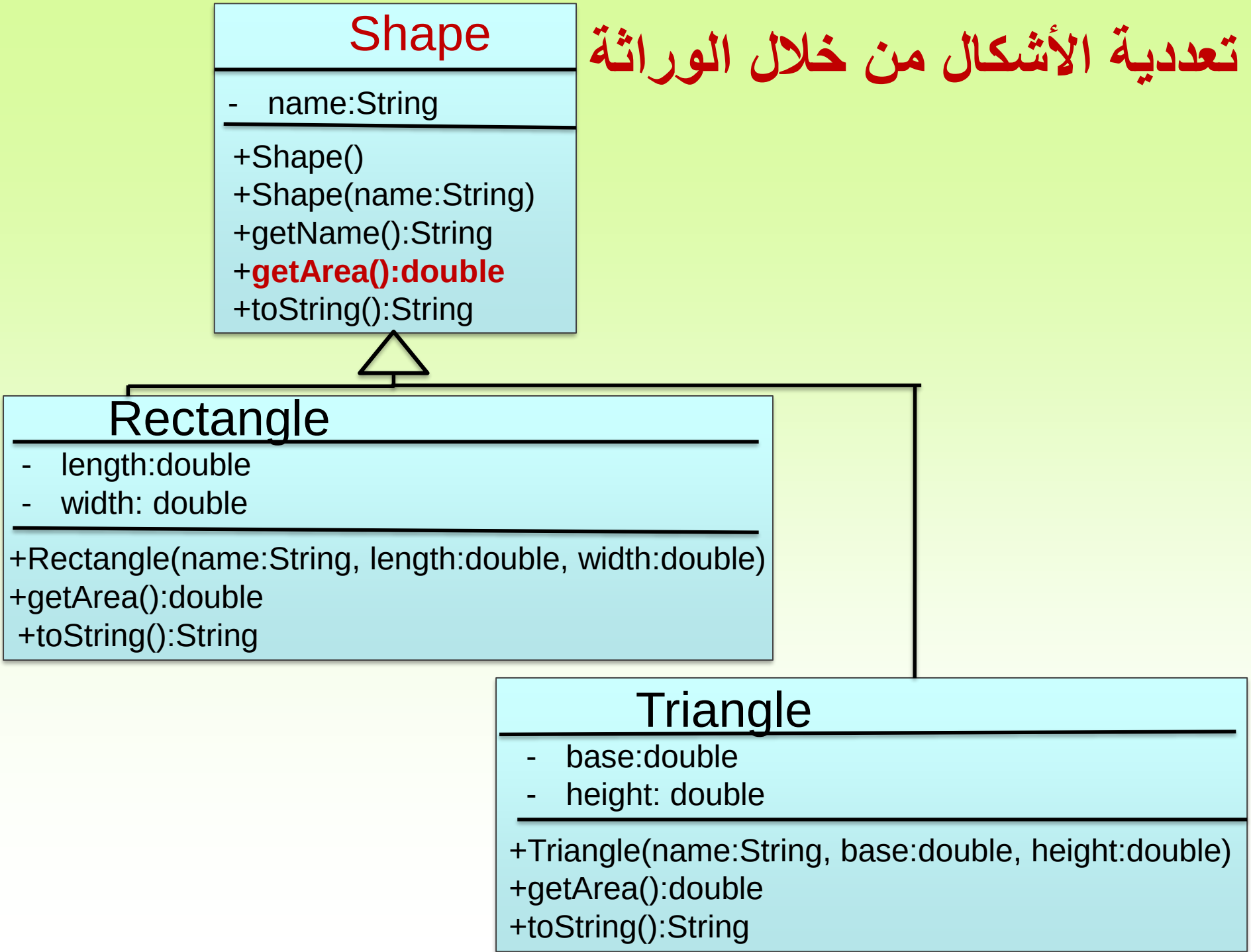
نتيجة التنفيذ:

```
int i = 5
double d = 5.0
```

تعددية الأشكال من خلال إعادة كتابة الطريقة

- تستخدم إعادة كتابة الطريقة في الوراثة و الصف المجرد و الواجهات من خلال جعل متغير مرجعي من صف الأب يغير سلوكه وفقا للغرض الذي يؤشر عليه من صف الأبناء.
- **ملاحظة:** يحقق تعددية الأشكال من هذا النمط مبدأ الـ Upcasting وهو التحويل من نوع الصف الابن إلى نوع الصف الأب، أي: انشاء متغير مرجعي r من الصف الأب و جعله يؤشر على غرض من صفوف الأبناء، و استدعاء الطرق التي تم إعادة كتابتها في الصف الابن من خلال المتغير المرجعيr .

تعددية الأشكال من خلال الوراثة



الصف الرئيسي PolymorphismShapeApp

```
package polymorphismshapeapp;
public class PolymorphismShapeApp {
  public static void main(String[] args) {
    Shape r=new Rectangle("rectangle",5,3);
    System.out.println("Area="+r.getArea());
    System.out.println(r.toString());
    r=new Triangle("triangle",10,5);
    System.out.println("Area="+r.getArea());
    System.out.println(r.toString());
  }
}
```

تعريف متغير مرجعي r من نوع الصف الأب Shape، و جعله يؤشر على غرض من صف الابن Rectangle

ملاحظات:

- المتغير مرجعي r يستطيع أن يستدعى الدالة getArea الموجودة في صف الابن Rectangle
- المتغير مرجعي r يستطيع أن يستدعى الدالة toString الموجودة في صف الابن Rectangle
- لاحظ أن المتغير المرجعي من نوع الصف الأب يمكنه فقط استدعاء الطرق من صف الابن التي تم إعادة كتابتها فيه، أي لا يمكنه استدعاء طريقة موجودة في صف الابن و غير موجودة في صف الأب.

ملاحظات

- تم تعريف متغير مرجعي r من نوع الصف الأب Shape، و جعله يؤشر على غرض من صف الابن Rectangle
- المتغير مرجعي r يستطيع أن يستدعى الدالة getArea الموجودة في صف الابن Rectangle
- المتغير مرجعي r يستطيع أن يستدعى الدالة toString الموجودة في صف الابن Rectangle
- الطريقة getArea في صف الأب Shape تم إعادة كتابتها في صف الابن Rectangle
- الطريقة toString في صف الأب Shape تم إعادة كتابتها في صف الابن Rectangle
- المتغير المرجعي r من نوع الصف الأب Shape، يؤشر على غرض من صف الابن Triangle
- المتغير مرجعي r يستطيع أن يستدعى الدالة getArea الموجودة في صف الابن Triangle
- المتغير مرجعي r يستطيع أن يستدعى الدالة toString الموجودة في صف الابن Triangle
- الطريقة getArea في صف الأب Shape تم إعادة كتابتها في صف الابن Triangle
- الطريقة toString في صف الأب Shape تم إعادة كتابتها في صف الابن Triangle
- **لاحظ أن المتغير المرجعي من نوع الصف الأب يمكنه فقط استدعاء الطرق من صف الابن التي تم إعادة كتابتها فيه، أي لا يمكنه استدعاء طريقة موجودة في صف الابن و غير موجودة في صف الأب.**

طريقة أخرى لكتابة الصف الرئيسي PolymorphismShapeApp
للتعبير عن تعددية الأشكال من خلال المتجهات

تعريف متجه r كل عنصر منه من نوع صف الأب Shape

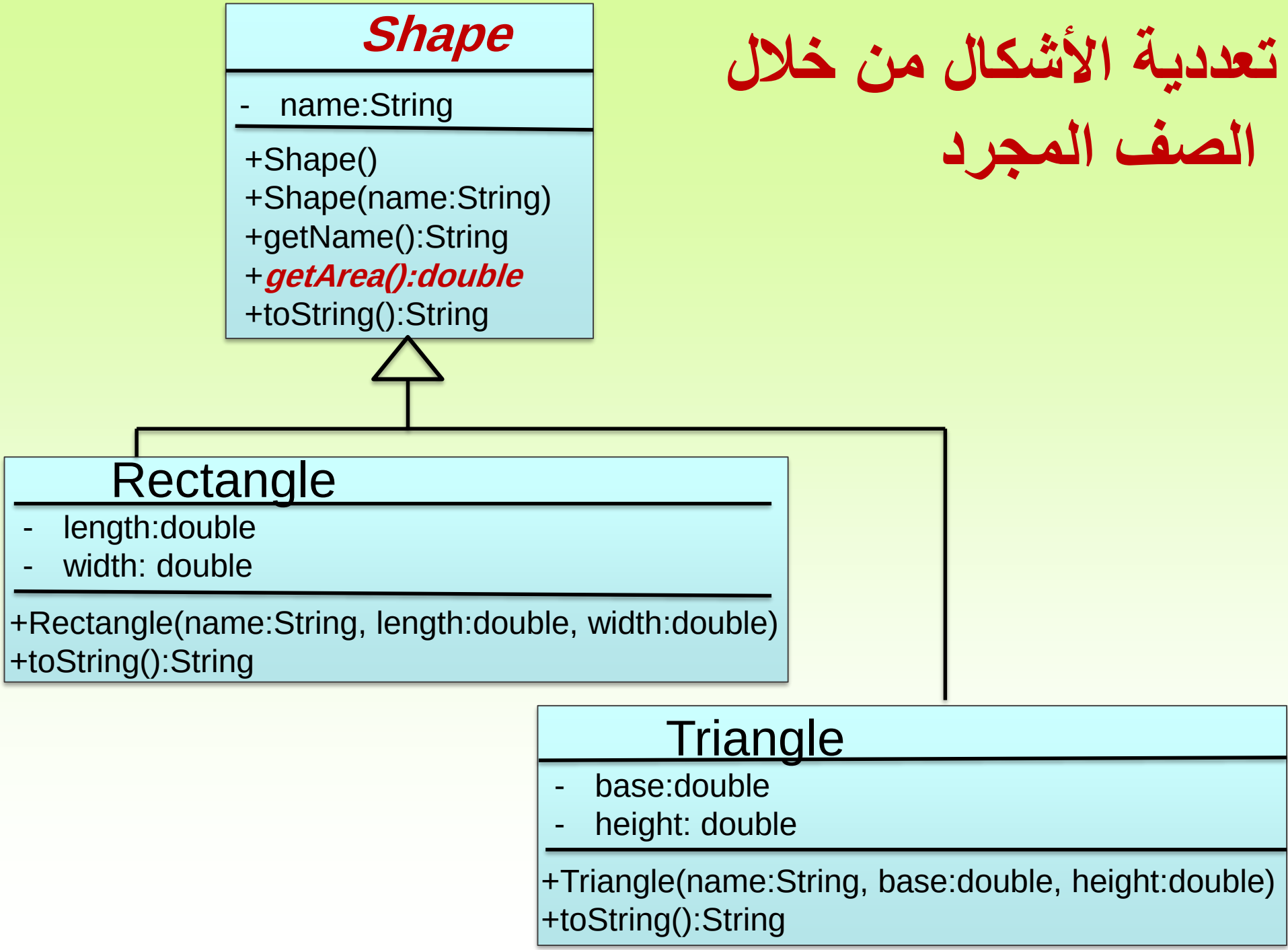
```
package polymorphismshapeapp;
public class PolymorphismShapeApp {
  public static void main(String[] args) {
    Shape[] r = new Shape[2];
    r[0] = new Rectangle("rectangle",5,3);
    r[1] = new Triangle("triangle",10,5);
    for (int i = 0; i < r.length; i++) {
      System.out.println("Area="+r[i].getArea());
      System.out.println(r[i].toString());
    }
  }
}
```

يمكن وضع أي صف من صفوف الأبناء في المتجه r، و الذي هو من نوع صف الأب Shape

التنفيذ

```
The Area= 15.0
the name of the shape is rectangle Length=5.0 width=3.0 The area = 15.0
The Area= 25.0
the name of the shape is triangle base= 10.0 height=5.0 The area = 25.0
```

تعددية الأشكال من خلال الصف المجرد



الصف الرئيسي
PolymorphismAbstractShapeApp

```
package polymorphismabstractshapeapp;
import java.util.*;
public class PolymorphismAbstractShapeApp {
  public static void main(String[] args) {
    Shape r=new Rectangle("rectangle",5,3);
    System.out.println("The Area= "+r.getArea());
    System.out.println(r.toString());
    r=new Triangle("triangle",10,5);
    System.out.println("The Area= "+r.getArea());
    System.out.println(r.toString());
  }
}
```

ملاحظات

- تم تعريف متغير مرجعي r من نوع صف الأب المجرد Shape، و جعله يؤشر على غرض من صف الابن Rectangle
- المتغير مرجعي r يستطيع أن يستدعى الطريقة المجردة getArea الموجودة في صف الابن Rectangle
- الطريقة المجردة getArea في صف الأب Shape تم إعادة كتابتها في صف الابن Rectangle
- الطريقة toString في صف الأب Shape تم إعادة كتابتها في صف الابن Rectangle
- المتغير المرجعي r من نوع الصف الأب Shape، يؤشر على غرض من صف الابن Triangle
- المتغير مرجعي r يستطيع أن يستدعي الطريقة المجردة getArea الموجودة في صف الابن Triangle
- الطريقة getArea في صف الأب Shape تم إعادة كتابتها في صف الابن Triangle
- الطريقة toString في صف الأب Shape تم إعادة كتابتها في صف الابن Triangle
- **لاحظ أن المتغير المرجعي من نوع صف الأب المجرد يمكنه فقط استدعاء الطرق المجردة و غير المجردة من صف الابن التي تم إعادة كتابتها فيه، أي لا يمكنه استدعاء طريقة موجودة في صف الابن و غير موجودة في صف الأب.**

طريقة أخرى لكتابة الصف الرئيسي PolymorphismAbstractShapeApp
للتعبير عن تعددية الأشكال من خلال المتجهات Polymorphic Array

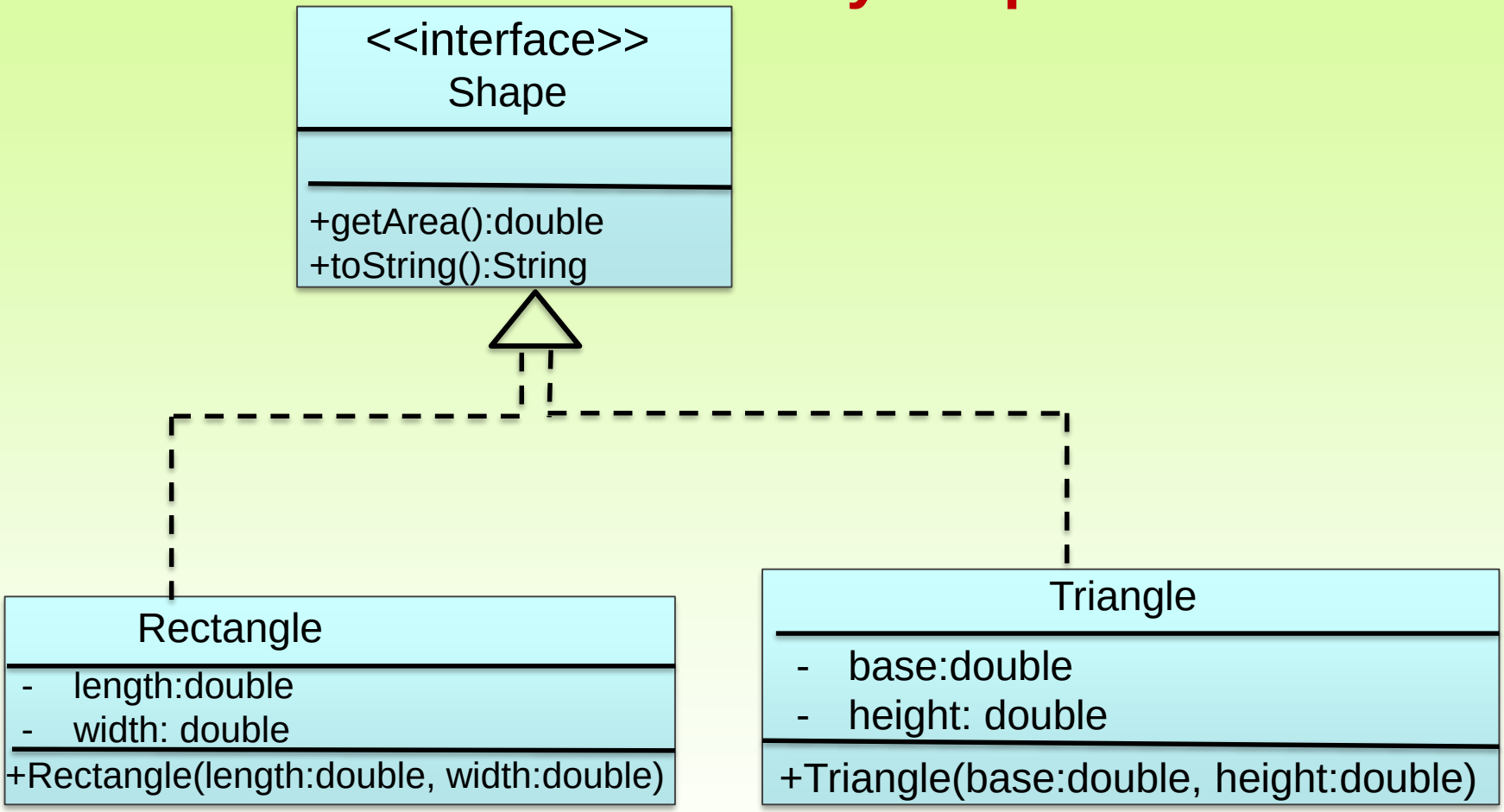
```
package polymorphismabstractshapeapp;
import java.util.*;
public class PolymorphismAbstractShapeApp {
   public static void main(String[] args) {
      Shape[] r = new Shape[2];
      r[0] = new Rectangle("rectangle",5,3);
      r[1] = new Triangle("triangle",10,5);
      for(int i = 0; i < r.length; i++) {
         System.out.println("Area="+r[i].getArea());
         System.out.println(r[i].toString());
      }
   }
}
```

التنفيذ

```
The Area= 15.0
the name of the shape is rectangle Length=5.0 width=3.0 The area = 15.0
The Area= 25.0
the name of the shape is triangle base= 10.0 height=5.0 The area = 25.0
```

تعددية الأشكال من خلال الواجهات

Polymorphism via Interfaces



الصف الرئيسي

PolymorphismInterfaceShapeApp

```
package polymorphisminterfaceshapeapp;
public class PolymorphismInterfaceShapeApp {
  public static void main(String[] args) {
    Shape r=new Rectangle(5,3);
    System.out.println("The Area= "+r.getArea());
    System.out.println(r.toString());
    r=new Triangle(10,5);
    System.out.println("The Area= "+r.getArea());
    System.out.println(r.toString());
  }
}
```

طريقة أخرى لكتابة الصف الرئيسي PolymorphismInterfaceShapeApp للتعبير عن تعددية الأشكال من خلال المتجهات

```
package polymorphisminterfaceshapeapp;
public class PolymorphismInterfaceShapeApp {
  public static void main(String[] args) {
    Shape[] r = new Shape[2];
    r[0] = new Rectangle(5,3);
    r[1] = new Triangle(10,5);
    for(int i = 0; i < r.length; i++) {
      System.out.println("Area="+r[i].getArea());
      System.out.println(r[i].toString());
    }
  }
}
```

التنفيذ

```
The Area= 15.0
For the Rectangle Length=5.0 Width=3.0 The area = 15.0
The Area= 25.0
For the Triangle Base=10.0 Height=5.0 The area = 25.0
```

تعددية الأشكال من خلال تمرير متغير مرجعي من نوع الصف الأب كوسيط في طريقة Polymorphic Argument

تم إنشاء الصف ShapeInfo وإضافة الطريقتين area و getInfo له و كلاهما تستقبل وسيطا من نوع صف الأب Shape

```
package polymorphismargumentshapeapp;
public class ShapeInfo {
   double area(Shape s){
    return s.getArea();
   }
   String getInfo(Shape s){
    return s.toString();
   }
}
```

الصف الرئيسي
PolymorphismArgumentShapeApp

```
package polymorphismargumentshapeapp;
public class PolymorphismArgumentShapeApp {
  public static void main(String[] args) {
        Rectangle r=new Rectangle("rectangle",5,3);
        Triangle t=new Triangle("triangle",10,5);
        ShapeInfo s=new ShapeInfo();
        System.out.println("Area="+s.area(r));
        System.out.println(s.getInfo(r));
        System.out.println("Area="+s.area(t));
        System.out.println(s.getInfo(t));
  }
}
```

التنفيذ

```
The Area= 15.0
For the Rectangle Length=5.0 Width=3.0 The area = 15.0
The Area= 25.0
For the Triangle Base=10.0 Height=5.0 The area = 25.0
```

التعليمة instanceof

`instanceof`: determines whether a reference variable that points to an object is of a particular class type

تستخدم هذه التعليمة لاختبار فيما إذا كان غرض ما a من نوع صف B.

الصيغة العامة لاستخدام هذه التعليمة هي:

Object instanceof Class

حيث Object هو الغرض و Class هو الصف. تعيد هذه التعليمة true إذا كان Object من نوع الصف Class و false خلاف ذلك.

مثال على استخدام التعليمة instanceof في الصف الرئيسي PolymorphismShapeApp

```
package polymorphismshapeapp;
public class PolymorphismShapeApp {
  public static void main(String[] args) {
    Shape[] r = new Shape[2];
    r[0] = new Rectangle("rectangle",5,3);
    r[1] = new Triangle("triangle",10,5);
    for(int i = 0; i < r.length; i++) {
        if(r[i] instanceof Rectangle)
            System.out.println("Rectangle");
        else if(r[i] instanceof Triangle)
            System.out.println("triangle");
        System.out.println("Area="+r[i].getArea());
        System.out.println(r[i].toString());
    }
```

مثال برنامج ReviewApp

اكتب برنامجا بلغة الجافا للتوصيف التالي:

- **تعريف صف مجرد اسمه Employee**، له رقم و اسم و اسم القسم الذي يعمل به، إضافة لحقل working لتبيان فيما إذا كان على رأس عمله أم لا. يحوي الصف باني مناسب للتوصيف. يعرف الصف طريقة مجردة setPerformedDuity ، كما يعرف الطريقة info والتي ترجع معلومات الموظف.
- **تعريف صف Doctor علما أنه موظف، له اختصاص. كما** يحوي باني مناسب للتوصيف، إضافة للطريقة prescribeMedicine التي تطبع العبارة "Write a prescribe medicine"، و الطريقة diagonsePatient التي تطبع العبارة "diagnose Patient" والطريقة info التي ترجع معلومات الطبيب، و الطريقة setPerformedDuity التي تستدعي كل الطرق السابقة
- **تعريف صف Nurse علما أنها موظفة. كما** يحوي باني مناسب للتوصيف، إضافة للطريقة drawBlood التي تطبع العبارة "drawing Bood"، و الطريقة checkVitalSign التي تطبع العبارة "Chacking Vital Sign" و الطريقة setPerformedDuity التي تستدعي كل الطرق السابقة، إضافة للطريقة info الموجودة في الصف Employee.
- **تعريف صف HospitalManagement** يحوي الطريقة callupon والتي تستقبل غرض من صف موظف كوسيط وتستدعي الطريقة setPerformedDuity
- الصف الرئيسي ReviewAppيقوم بإنشاء أغراض من الصفوف السابقة و استدعاء الطرق الموجودة فيها.

البرامج المطلوب قراءتها و تنفيذها و فهمها

- **PolymorphismShapeApp**
- **PolymorphismAbstractShapeApp**
- **PolymorphismInterfaceShapeApp**
- **PolymorphismArgumentShapeApp**
- **ReviewApp**