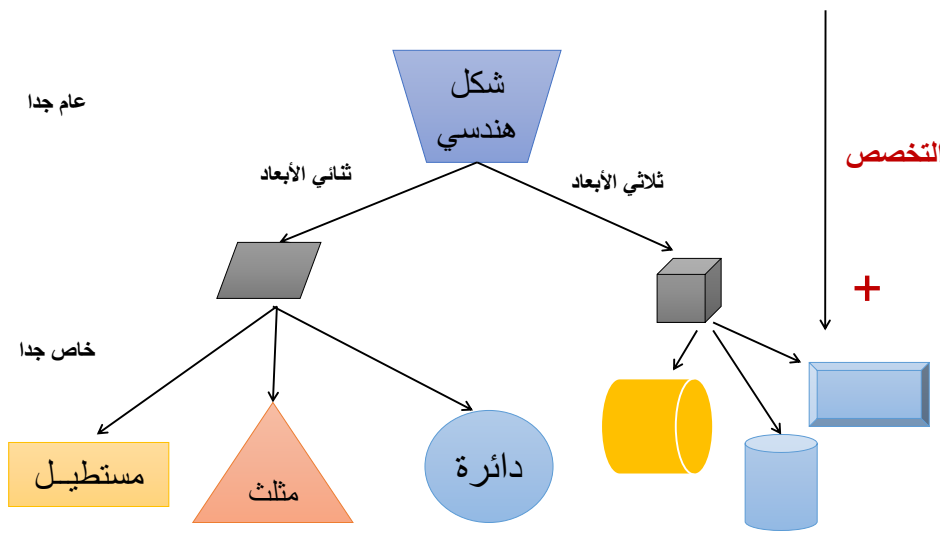
مفهوم تعدد الأشكال Polymorphism

- ✓مفهوم تعدد الأشكال يسمح لنا بكتابة برامج في صورة قابلة للتغيير بشكل واسع النطاق؛ سواء كان التغيير لفئات موجودة مسبقاً أو تغيير مستقبلي لإنتاج برامج جديدة.
- ✓هذه الخاصية تسهل علينا توسيع قدرات نظامنا. وتجعله أكثر مرونه. من خلاله يتم تعريف أصناف عامة جدًا في الصفات والسلوك ثم بعد ذلك يتم تخصيصها أكثر فأكثر من خلال تطبيق مخصص للسلوك وتعريف المزيد من المتغيرات.

مفهوم تعدد الأشكال Polymorphism

✓ Polymorphismيعني قدرة الكائن object على أن يأخذ عدة أشكال. والاستخدام الأكثر شيوعًا للـPolymorphism يحدث عندما يُستخدم أب parent class للإشارة إلى يرث منه child class.

مفهوم تعدد الأشكال Polymorphism

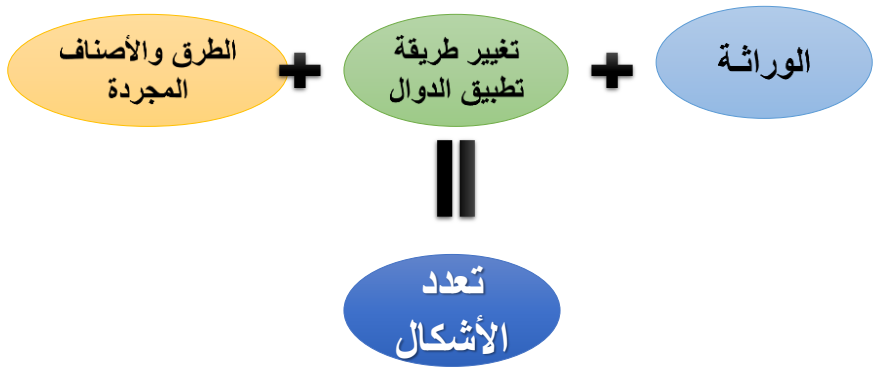


لماذا تعدد الأشكال ؟

- ✓تحقيق لمباديء البرمجة الشيئية .
- ✓توسيع لمفهوم الوراثة.
- ✓إنتاج مشاريع برمجيـة أكثر ليونة و قدرة على التطور.

علاقة تعدد الأشكال بالوراثـة ...

- ✓مفهوم تعدد الأشكال هو مفهوم أكثر عمقًا و تخصصـا من مفهوم الوراثة.



أمثلة نحتاج فيها تعدد الأشكال ؟

مثال : (حركة الحيوانات)

- ✓ عندي صنف الحيوان، ومشتق منه الضفدع، السمكة، الطيور. والمعروف أن كل الحيوانات تتحرك وتغير موضعها ولكن **طريقة** حركتها ومقدارها يتغير من صنف لآخر.
- ✓ في هذه الحالة نستخدم تعدد الأشكال لنقوم بتطبيق الصنف (حيوان) وبه السلوك (حركة) ولكن دون أن نقوم بتعريف كيفية حدوث الحركة ثم نقوم في الأصناف المشتقة بتعريف كيفية حدوث الحركة لكل صنف من الأصناف.

أمثلة نحتاج فيها تعدد الأشكال ؟

مثال : (حساب مساحة الأشكال الهندسية)

- ✓ الـclass العام اسمه شكل هندسي(Shape) له أبعاد ومساحة ومحيط، لوقمنا باشتقاق مربع، مستطيل، دائرة، اسطوانة منه. ولكل منهم مساحة ومحيط سيتم حسابهم بطريقة مختلفة. فحساب مساحة المربع تختلف عن طريقة حساب مساحة الدائرة، وتختلف عن حساب طريقة حساب مساحة المستطيل، وهكذا.
- ✓ في المثال التالي سنطبق ذلك ونستخدم مفهوم تعدد الاشكال لنقوم بتطبيق الـclass (شكـــــل هندسي Shape) وبه دالة حساب المساحة area وسنقوم باشتقاق أصناف classes مختلفة (دائرة circle) ومستطيل rectangle وفيها سيتم توضيح كيفية حساب المساحة لكل صنف من الأصناف بالطريقة المناسبة له.

مثال:

```
class  Shape{
        public void area(){
            System.out.println("Shape");

        }
    }
class  Circle extends Shape{
    double r=2;
    @Override
        public void area(){
            System.out.println("Circle area=" + 3.14 * r * r);
        }
    }
class  rectangle extends Shape{
    double length=20,widthr=10;
    @Override
        public void area(){
            System.out.println("rectangle area=" + length*widthr);
        }
    }
class Test {
        public void area (Shape  s){
            s.area();
        }
}
```

مثال:

```
1   package poly_java;
2   public class Poly_Java {
3       public static void main(String[] args) {
4           Shape s = new Shape();
5           Circle c=new Circle();
6           rectangle r=new rectangle();
7           s.area();
8           c.area();
9           r.area();
10
11          Test t =new Test();
12          t.area(s);
13          t.area(c);
14          t.area(r);
15                  }
16  }
```

الكلاس العام شكل هندسي Shape: يحوي دالة لحساب المساحة لا تحوي الا جملة طباعة

```
class  Shape{
        public void area(){
            System.out.println("Shape");

        }
    }
```

1

الكلاس دائرة وهو كلاس يرث الكلاس Shape ويحوي دالة area لحساب مساحة الدائرة بعمل overriding للدالة area الموجودة في الكلاس الأب Shape

```
class  Circle extends Shape{
    double r=2;
    @Override
        public void area(){
            System.out.println("Circle area=" + 3.14 * r * r);
        }
    }
```

2

الكلاس مستطيل وهو كلاس يرث الكلاس Shape ويحوي دالة area لحساب مساحة المستطيل بعمل overriding للدالة area الموجودة في الكلاس الأب Shape. **لاحظ الفرق في عمل نفس الدالة في كل كلاس**

```
class  rectangle extends Shape{
    double length=20,widthr=10;
    @Override
        public void area(){
            System.out.println("rectangle area=" + length*widthr);
        }
    }
```

3

لاحظ معامل الدالة (الباراميتر) **هو من النوع Shape** أي الكلاس الأساسي الذي تشتق منه كل الكلاسات الأخرى، أي الكلاس الأساسي والكائنات المشتقة منه، هنا تعدد الاشكال مطبق على الدالة area لتقوم باعمال مختلفة باختلاف الـobject المُستدعى لتنفذ الدالة المناسبة

```
class Test {
        public void area (Shape  s){
            s.area();
        }
}
```

4

```
public class Poly_Java {
    public static void main(String[] args) {
        Shape s = new Shape();
        Circle c=new Circle();
        rectangle r=new rectangle();
        s.area();
        c.area();
        r.area();

        Test t =new Test();
        t.area(s);
        t.area(c);
        t.area(r);
            }
}
```

5

هنا لم يطبق
Polymorphism

```
run:
Shape
Circle area=12.56
rectangle area=200.0
```

هنا طبق
Polymorphism

```
Shape
Circle area=12.56
rectangle area=200.0
```

مثال آخر: تعدد الأشكال لدالة الجمع

```
public class A {
   public void Add(int x,int y){
      System.out.println(x+y);
   }
}
public class B extends A{
   @Override
   public void Add(int x,int y){
      System.out.println(x+""+y);
   }
}

 public class D {
    public void Add (A adder ,int x,int y){
       adder.Add(x, y);
    }
 }
```

```
package polymorphism2;
public class Polymorphism2 {
   public static void main(String[] args) {
      D d =new D();
      A a= new A();
      B  b=new B();
      a.Add(10,20);
      b.Add(10,20);
      d.Add(a,50,25);
      d.Add(b,50,25);
   }
}
```