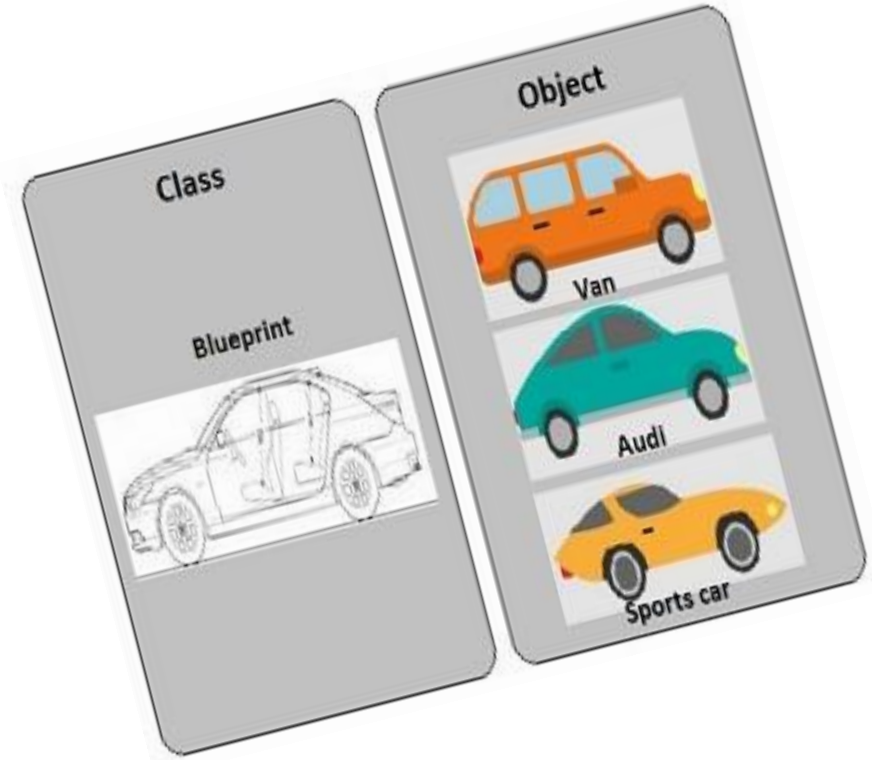
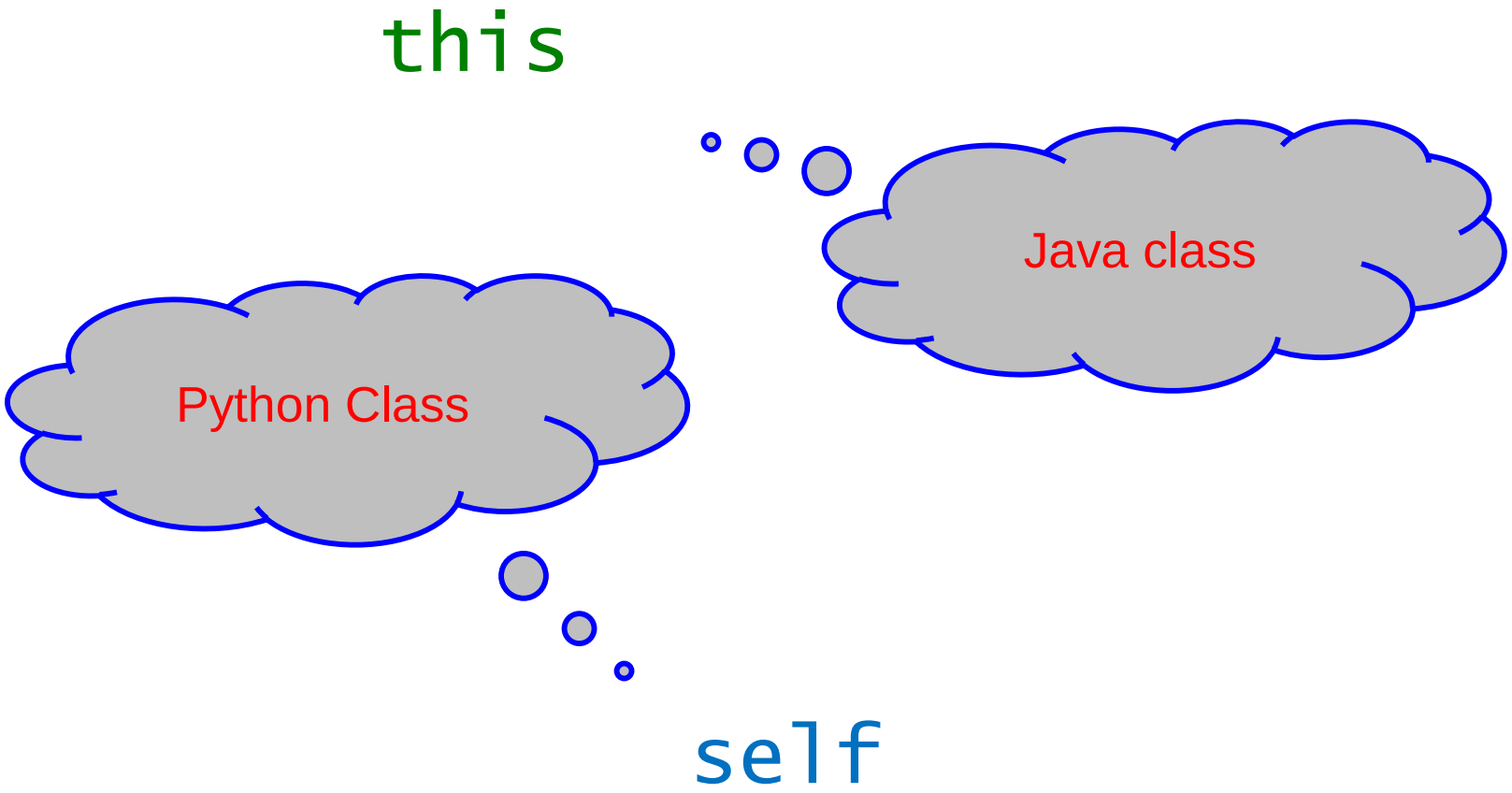


From Python to Java: writing our own classes to build *custom* data types

Computer Science 112
Boston University

Christine Papadakis-Kanaris

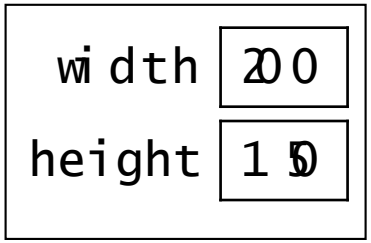
Classes:
Python vs. Java



Example: A Rectangle Class

- Let's say that we want to create a data type for objects that represent rectangles.

- Every Rectangle object should have two variables inside it (*width* and *height*) for the rectangle's dimensions.
  - these variables are referred to as *fields*
  - also known as: attributes, instance variables



- We'll also put functions/methods inside the object.

An Initial `Rectangle` Class

Python

```
class Rectangle:
    def __init__(self, w, h):
        self.width = w
        self.height = h
```

Java

```
public class Rectangle {
    int width;
    int heig
                non-static, no return type
    public Rectangle(int w, int h) {
        this.width = w;
        this.height = h;
    }
}
```

- `__init__` is the *constructor*.
  - used to create objects of the class

- The constructor has the same name as the class.
  - it is non-static
  - It has no return type

An Initial `Rectangle` Class

A constructor returns a reference to the instance of the class (i.e. object) that was created.

Java

```
public class Rectangle {
    int width;
    int heig
                non-static, no return type
    public Rectangle(int w, int h) {
        this.width = w;
        this.height = h;
    }
}
```

- `__init__` is the *constructor*.
  - used to create objects of the class

- The constructor has the same name as the class.
  - it is non-static
  - It has no return type

An Initial `Rectangle` Class

Just as the return is implicit, so is the return type. The implicit data type in this example is: Rectangle!

Java

```
public class Rectangle {
    int width;
    int heig
                non-static, no return type
    public Rectangle(int w, int h) {
        this.width = w;
        this.height = h;
    }
}
```

- `__init__` is the *constructor*.
  - used to create objects of the class

- The constructor has the same name as the class.
  - it is non-static
  - It has no return type

An Initial `Rectangle` Class

Python

```python
class Rectangle:
    def __init__(self, w, h):
        self.width = w
        self.height = h
```

Java

```java
public class Rectangle {
    int width;
    int height;

    public Rectangle(int w, int h) {
        this.width = w;
        this.height = h;
    }
}
```

- `__init__` is the *constructor*.
  - used to create objects of the class
- All methods have a `self` parameter.
  - use it to access the object it*self*!

- The constructor has the same name as the class.
  - it is non-static
  - it has no return type
- All non-static methods have an *implicit* parameter called `this`.
  - use it to access *this* object!

An Initial `Rectangle` Class

Python

```
class Rectangle:
    def __init__(self, w, h):
        self.width = w
        self.height = h
```

Java

```
public class Rectangle {
    int width;
    int height;

    public Rectangle(int w, int h) {
        this.width = w;
        this.height = h;
    }
```

Note that there is no `this` parameter that is explicitly passed to the method…

- __i
  - l
    c
- All methods have self parameter.
  - use it to access the object it*self*!

he constructor has the same ame as the class.

- it is non-static
- it has no return type

- All non-static methods have an *implicit* parameter called `this`.
  - use it to access *this* object!

An Initial `Rectangle` Class

Python

```
class Rectangle:
    def __init__(self, w, h):
        self.width = w
        self.height = h
```

Java

```
public class Rectangle {
    int width;
    int height;

    public Rectangle(int w, int h) {
        this.width = w;
        this.height = h;
    }
```

… In java, it is implicitly passed to every **non-static** method of the class.

- __i
  - l
    c
- All methods have self parameter.
  - use it to access the object it*self*!

he constructor has the same ame as the class.

- it is non-static
- it has no return type

- All non-static methods have an *implicit* parameter called `this`.
  - use it to access *this* object!

An Initial `Rectangle` Class

Python

```
class Rectangle:
    def __init__(self, w, h):
        self.width = w
        self.height = h
```

Java

```
public class Rectangle {
    int width;
    int height;

    public Rectangle(int w, int h) {
        this.width = w;
        this.height = h;
    }
}
```

The `this` parameter can be **explicitly** ..

- pa
  - use it to a
    the object it*self*!

The constructor has the same name as the class.

- it is non-static
- it has no return type

- All non-static methods have an *implicit* parameter called `this`.
  - use it to access *this* object!

An Initial `Rectangle` Class

Python

```python
class Rectangle:
    def __init__(self, w, h):
        self.width = w
        self.height = h
```

Java

```java
public class Rectangle {
    int width;
    int height;

    public Rectangle(int w, int h) {
        width = w;
        height = h;
    }
}
```

The `this` parameter can be explicitly or **implicitly** used to access data members of the object.

- The constructor has the same name as the class.
  - it is non-static
  - it has no return type

- All non-static methods have an *implicit* parameter called `this`.
  - use it to access *this* object!

- pa
  - use it to a
    the object it*self*!

An Initial `Rectangle` Class

Python

```
class Rectangle:
    def __init
```

Java

```
public class Rectangle {
    int width;
    int height;

    ublic Rectangle(int w, int h) {
        width = w;
        height = h;
```

Static methods are methods of the class. They have *class level* scope and cannot be called on instances (i.e. objects) of the class.

- `__init` ... *uctor*.
  - used to create objec of the class
- All methods have a `self` parameter.
  - use it to access the object it*self*!

- The constructor has the same name as the class.
  - it is non-static
  - it has no return type
- All non-static methods have an *implicit* parameter called `this`.
  - use it to access *this* object!

An Initial `Rectangle` Class

Python

```
class Rectangl
    d
```

Java

```
public class Rectangle {
    int width;
    int height;

    public Rectangle(int w, int h) {
        width = w;
        height = h;
```

Since static methods are not called on an object, there is no associated *this* reference (or pointer) to pass to them.

- `__in` ... *uctor*.
  - used to create obje of the class
- All methods have a `self` parameter.
  - use it to access the object it*self*!

- The constructor has the same name as the class.
  - it is non-static
  - it has no return type
- All non-static methods have an *implicit* parameter called `this`.
  - use it to access *this* object!

An Initial `Rectangle` Class

Python

```
class Rectangle:
    def __init
```

Java

```
public class Rectangle {
    int width;
    int height;

    ublic Rectangle(int w, int h) {
        width = w;
        height = h;
```

And again the reason why constructors, like all instance methods of a class, cannot be declared as static methods!

- `__init` ... *uctor*.
  - used to create objec of the class
- All methods have a `self` parameter.
  - use it to access the object it*self*!

- The constructor has the same name as the class.
  - it is non-static
  - it has no return type
- All non-static methods have an *implicit* parameter called `this`.
  - use it to access *this* object!

An Initial `Rectangle` Class

Python

```
class Rectangle:



    def __init__(self, w, h):
        self.width = w
        self.height = h
        width = w
        height = h
```

Java

```
public class Rectangle {



    public Rectangle(int w, int h) {
        width = w;
        height = h;
    }
}
```

- The fields are defined by assigning something to them in the constructor.

No Scope issue:

`self.width` is a data member
`width` is local variable

An Initial `Rectangle` Class

Python

```
class Rectangle:



    def __init__(self, w, h):
        self.width = w
        self.height = h
```

Java

```
public class Rectangle {
    int width;
    int height;

    public Rectangle(int width, int h)
        this.width = width;
        height = h;
    }
}
```

- The fields are defined by assigning something to them in the constructor.

- The fields (data members) must be declared separately.
  - outside of any method
  - usually near the class header
  - the constructor assigns their initial values

Adding Functionality to an Object

Python

```
class Rectangle:
    // do not declare
    // the fields!

    def __init__(self, w, h):
        self.width = w
        self.height = h


    def grow(self, dw, dh):
        self.width += dw
        self.height += dh
```

Java

```
public class Rectangle {
    int width;
    int height;

    public Rectangle(int w, int h) {
        this.width = w;
        this.height = h;
    }

    public void grow(int dw, int dh) {
        this.width += dw;
        this.height += dh;
    }
}
```

- `self` *is* in the param list.
- it gets its value from the called object
  - ex: `r1.grow(50,10)`

Adding Functionality to an Object

Python

```
class Rectangle:
    // do not declare
    // the fields!

    def __init__(self, w, h):
        self.width = w
        self.height = h


    def grow(self, dw, dh):
        self.width += dw
        self.height += dh
```

Java

```
public class Rectangle {
    int width;
    int height;

    public Rectangle(int w, int h) {
        this.width = w;
        this.height = h;
    }
                    non-static
    public void grow(int dw, int dh) {
        this.width += dw;
        this.height += dh;
```

this

- `self` *is* in the param list.
- it gets its value from the called object
  - ex: `r1.grow(50,10)`

- `this` is *not* in the parameter list
- it also gets its value from the called object
  - ex: `r1.grow(50,10)`

Adding Functionality to an Object:

a method to compute the area

Python

```python
class Rectangle:
    // do not declare
    // the fields!

    def __init__(self, w, h):
        self.width = w
        self.height = h


   def grow(self, dw, dh):
        self.width += dw
        self.height += dh


   def area(self):
       return self.width
              * self.height
```

Java

```java
public class Rectangle {
    int width;
    int height;

    public Rectangle(int w, int h) {
        this.width = w;
        this.height = h;
    }

    public void grow(int dw, int dh) {
        this.width += dw;
        this.height += dh;
    }

    public int area() {
        return this.width * this.height;
    }
}
```

Simplifying the Client Program

```
public class RectangleClient {
    public static void main(String[] args) {
        Rectangle r1 = new Rectangle(100, 50);
        Rectangle r2 = new Rectangle(20, 80);

        int area1 = r1.width * r1.height;
        System.out.println("r1's area = " + area1);

        int area2 = r2.width * r2.height;
        System.out.println("r2's area = " + area2);

        // grow both rectangles
        r1.width += 50;  r1.height += 10;
        r2.width += 5;   r2.height += 30;

        System.out.println("r1: " + r1.width + " x " + r1.height);
        System.out.println("r2: " + r2.width + " x " + r2.height);
    }
}
```

Simplifying the Client Program

```
public class RectangleClient {
    public static void main(String[] args) {
        Rectangle r1 = new Rectangle(100, 50);
        Rectangle r2 = new Rectangle(20, 80);

        int area1 = r1.width * r1.height;
        System.out.println("r1's area = " + area1);

        int area2 = r2.width * r2.height;
        System.out.println("r2's area = " + area2);

        // grow both rectangles
        r1.width += 50;  r1.height += 10;
        r2.width += 5;   r2.height += 30;

        System.out.println("r1: " + r1.width + " x " + r1.height);
        System.out.println("r2: " + r2.width + " x " + r2.height);
    }
}
```

Simplifying the Client Program

```
public class RectangleClient {
    public static void main(String[] args) {
        Rectangle r1 = new Rectangle(100, 50);
        Rectangle r2 = new Rectangle(20, 80);

        int area1 = r1.area();
        System.out.println("r1's area = " + area1);

        int area2 = r2.width * r2.height;
        System.out.println("r2's area = " + area2);

        // grow both rectangles
        r1.width += 50;  r1.height += 10;
        r2.width += 5;   r2.height += 30;

        System.out.println("r1: " + r1.width + " x " + r1.height);
        System.out.println("r2: " + r2.width + " x " + r2.height);
    }
}
```

Simplifying the Client Program

```
public class RectangleClient {
    public static void main(String[] args) {
        Rectangle r1 = new Rectangle(100, 50);
        Rectangle r2 = new Rectangle(20, 80);

        int area1 = r1.area();
        System.out.println("r1's area = " + area1);

        int area2 = r2.width * r2.height;
        System.out.println("r2's area = " + area2);

        // grow both rectangles
        r1.width += 50;  r1.height += 10;
        r2.width += 5;   r2.height += 30;

        System.out.println("r1: " + r1.width + " x " + r1.height);
        System.out.println("r2: " + r2.width + " x " + r2.height);
    }
}
```

Simplifying the Client Program

```
public class RectangleClient {
    public static void main(String[] args) {
        Rectangle r1 = new Rectangle(100, 50);
        Rectangle r2 = new Rectangle(20, 80);

        int area1 = r1.area();
        System.out.println("r1's area = " + area1);

        int area2 = r2.area();
        System.out.println("r2's area = " + area2);

        // grow both rectangles
        r1.width += 50;  r1.height += 10;
        r2.width += 5;   r2.height += 30;

        System.out.println("r1: " + r1.width + " x " + r1.height);
        System.out.println("r2: " + r2.width + " x " + r2.height);
    }
}
```

Simplifying the Client Program

```
public class RectangleClient {
    public static void main(String[] args) {
        Rectangle r1 = new Rectangle(100, 50);
        Rectangle r2 = new Rectangle(20, 80);

        int area1 = r1.area();
        System.out.println("r1's area = " + area1);

        int area2 = r2.area();
        System.out.println("r2's area = " + area2);

        // grow both rectangles
        r1.width += 50;  r1.height += 10;
        r2.width += 5;   r2.height += 30;

        System.out.println("r1: " + r1.width + " x " + r1.height);
        System.out.println("r2: " + r2.width + " x " + r2.height);
    }
}
```

Simplifying the Client Program

```
public class RectangleClient {
    public static void main(String[] args) {
        Rectangle r1 = new Rectangle(100, 50);
        Rectangle r2 = new Rectangle(20, 80);

        int area1 = r1.area();
        System.out.println("r1's area = " + area1);

        int area2 = r2.area();
        System.out.println("r2's area = " + area2);

        // grow both rectangles
        r1.grow(50, 10);
        r2.width += 5;   r2.height += 30;

        System.out.println("r1: " + r1.width + " x " + r1.height);
        System.out.println("r2: " + r2.width + " x " + r2.height);
    }
}
```

Simplifying the Client Program

```
public class RectangleClient {
    public static void main(String[] args) {
        Rectangle r1 = new Rectangle(100, 50);
        Rectangle r2 = new Rectangle(20, 80);

        int area1 = r1.area();
        System.out.println("r1's area = " + area1);

        int area2 = r2.area();
        System.out.println("r2's area = " + area2);

        // grow both rectangles
        r1.grow(50, 10);
        r2.width += 5;   r2.height += 30;

        System.out.println("r1: " + r1.width + " x " + r1.height);
        System.out.println("r2: " + r2.width + " x " + r2.height);
    }
}
```

Simplifying the Client Program

```
public class RectangleClient {
    public static void main(String[] args) {
        Rectangle r1 = new Rectangle(100, 50);
        Rectangle r2 = new Rectangle(20, 80);

        int area1 = r1.area();
        System.out.println("r1's area = " + area1);

        int area2 = r2.area();
        System.out.println("r2's area = " + area2);

        // grow both rectangles
        r1.grow(50, 10);
        r2.grow(5, 30);

        System.out.println("r1: " + r1.width + " x " + r1.height);
        System.out.println("r2: " + r2.width + " x " + r2.height);
    }
}
```

Simplifying the Client Program

```
public class RectangleClient {
    public static void main(String[] args) {
        Rectangle r1 = new Rectangle(100, 50);
        Rectangle r2 = new Rectangle(20, 80);

        int area1 = r1.area();
        System.out.println("r1's area = " + area1);

        int area2 = r2.area();
        System.out.println("r2's area = " + area2);

        // grow both rectangles
        r1.grow(50, 10);
        r2.grow(5, 30);

        System.out.println("r1: " + r1.width + " x " + r1.height);
        System.out.println("r2: " + r2.width + " x " + r2.height);
    }
}
```

Simplifying the Client Program

```
public class RectangleClient {
    public static void main(String[] args) {
        Rectangle r1 = new Rectangle(100, 50);
        Rectangle r2 = new Rectangle(20, 80);

        int area1 = r1.area();
        System.out.println("r1's area = " + area1);

        int area2 = r2.area();
        System.out.println("r2's area = " + area2);

        // grow both rectangles
        r1.grow(50, 10);
        r2.grow(5, 30);

        System.out.println("r1: " + r1.getWidth() + " x " + r1.getHeight());
        System.out.println("r2: " + r2.getWidth() + " x " + r2.getHeight());
    }
}
```

Converting an Object to a String

- The `toString()` method allows objects to be displayed in a human-readable format.
  - it returns a string representation of the object

- This method is called *implicitly* when you attempt to print an object or when you perform string concatenation:

```
Rectangle r1 = new Rectangle(10, 20, 100, 80);
System.out.println(r1);
```

equivalent to:

```
System.out.println(r1.toString());
```

Revised Client Program

```
public class RectangleClient {
    public static void main(String[] args) {
        Rectangle r1 = new Rectangle(100, 50);
        Rectangle r2 = new Rectangle(20, 80);

        int area1 = r1.area();
        System.out.println("r1's area = " + area1);

        int area2 = r2.area();
        System.out.println("r2's area = " + area2);

        // grow both rectangles
        r1.grow(50, 10);
        r2.grow(5, 30);

        System.out.println("r1: " + r1.getWidth() + " x " + r1.getHeight());
        System.out.println("r2: " + r2.getWidth() + " x " + r2.getHeight());
    }
}
```

Revised Client Program

```
public class RectangleClient {
    public static void main(String[] args) {
        Rectangle r1 = new Rectangle(100, 50);
        Rectangle r2 = new Rectangle(20, 80);

        int area1 = r1.area();
        System.out.println("r1's area = " + area1);

        int area2 = r2.area();
        System.out.println("r2's area = " + area2);

        // grow both rectangles
        r1.grow(50, 10);
        r2.grow(5, 30);

        System.out.println("r1: " + r1);
        System.out.println("r2: " + r2.getWidth() + " x " + r2.getHeight());
    }
}
```

Revised Client Program

```
public class RectangleClient {
    public static void main(String[] args) {
        Rectangle r1 = new Rectangle(100, 50);
        Rectangle r2 = new Rectangle(20, 80);

        int area1 = r1.area();
        System.out.println("r1's area = " + area1);

        int area2 = r2.area();
        System.out.println("r2's area = " + area2);

        // grow both rectangles
        r1.grow(50, 10);
        r2.grow(5, 30);

        System.out.println("r1: " + r1);
        System.out.println("r2: " + r2);
    }
}
```

Revised Client Program

```
public class RectangleClient {
    public static void main(String[] args) {
        Rectangle r1 = new Rectangle(100, 50);
        Rectangle r2 = new Rectangle(20, 80);

        int area1 = r1.area();
        System.out.println("r1's area = " + area1);

        int area2 = r2.area();
        System.out.println("r2's area = " + area2);

        // grow both rectangles
        r1.grow(50, 10);
        r2.grow(5, 30);

        System.out.println("r1: " + r1);
        System.out.println("r2: " + r2);
    }
}
```