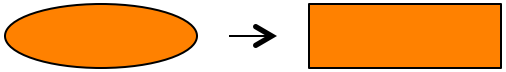
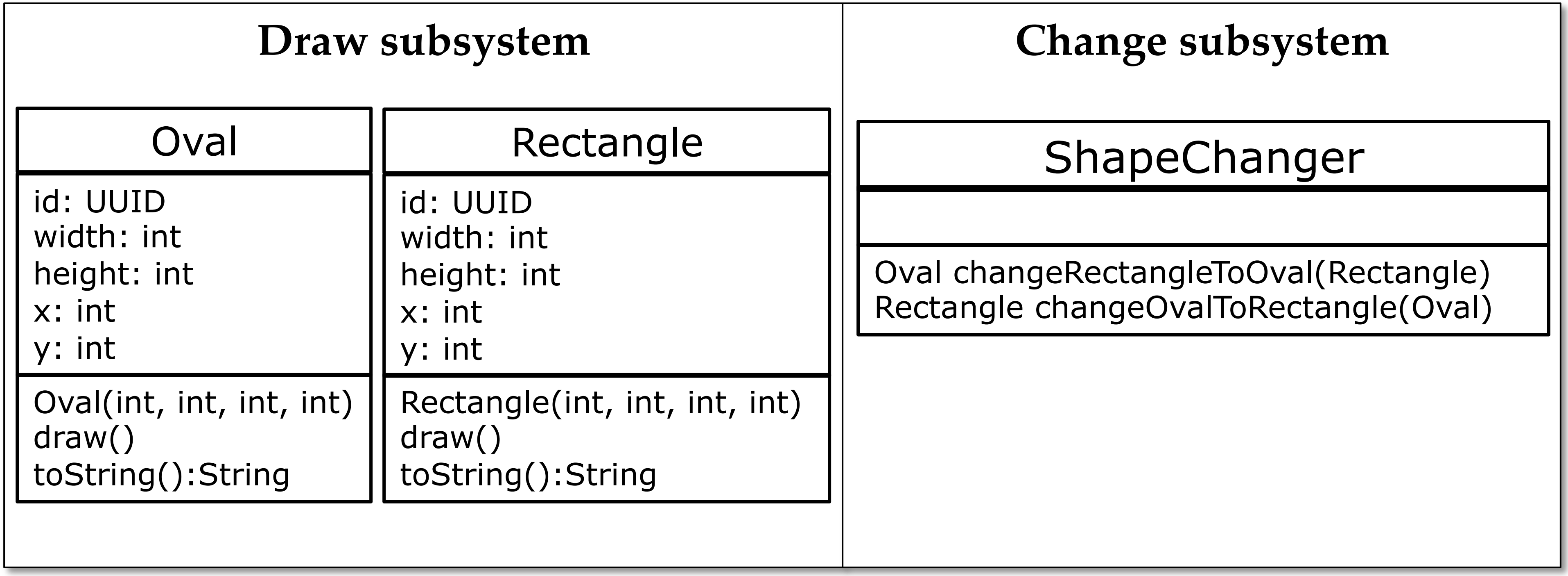
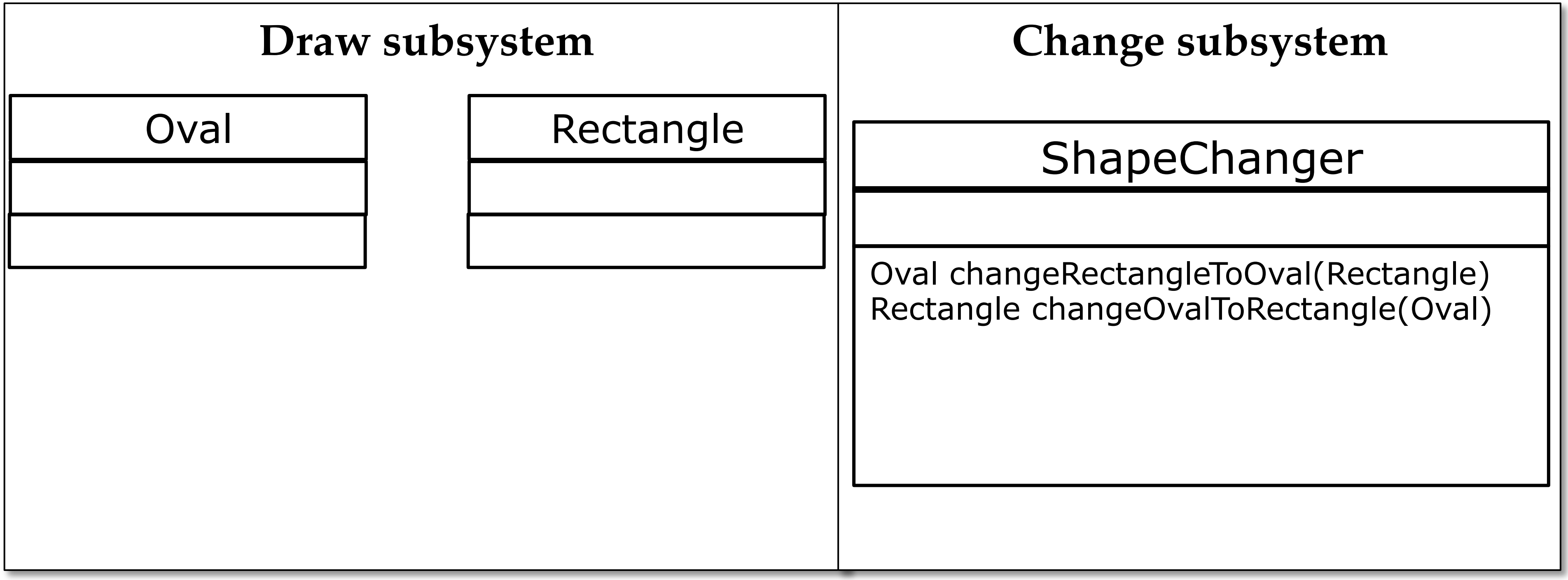
Exercise #1: Adding Code for Changing Shapes (Problem Statement)

- Imagine a legacy system with an existing subsystem for *drawing* shapes
- New requirement: Support also *changing* of shapes
  - Example: Change an oval into a rectangle 
- The manager says: "Do not touch the existing code!"



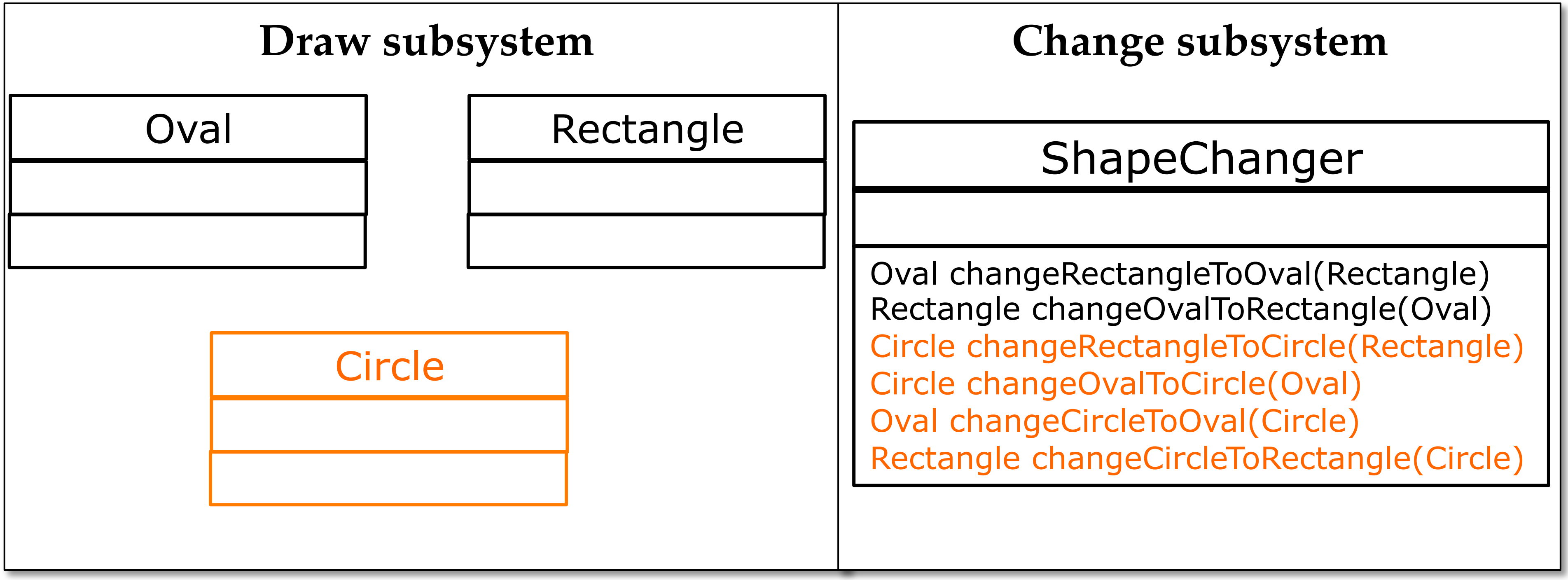
Exercise #1: Adding Code for Changing Shapes (Problem Statement)

- The functional decomposition leads to the code implementing change functionality being in a separate subsystem
- For each combination of Shapes code implementing change functionality has to be written


Draw subsystem
Oval
Rectangle
Change subsystem
ShapeChanger
Oval changeRectangleToOval(Rectangle)
Rectangle changeOvalToRectangle(Oval)

Exercise #1: Adding Code for Changing Shapes (Problem Statement)

- The functional decomposition leads to the code implementing change functionality being in a separate subsystem
- For each combination of Shapes code implementing change functionality has to be written
  - Especially adding new shapes becomes expensive

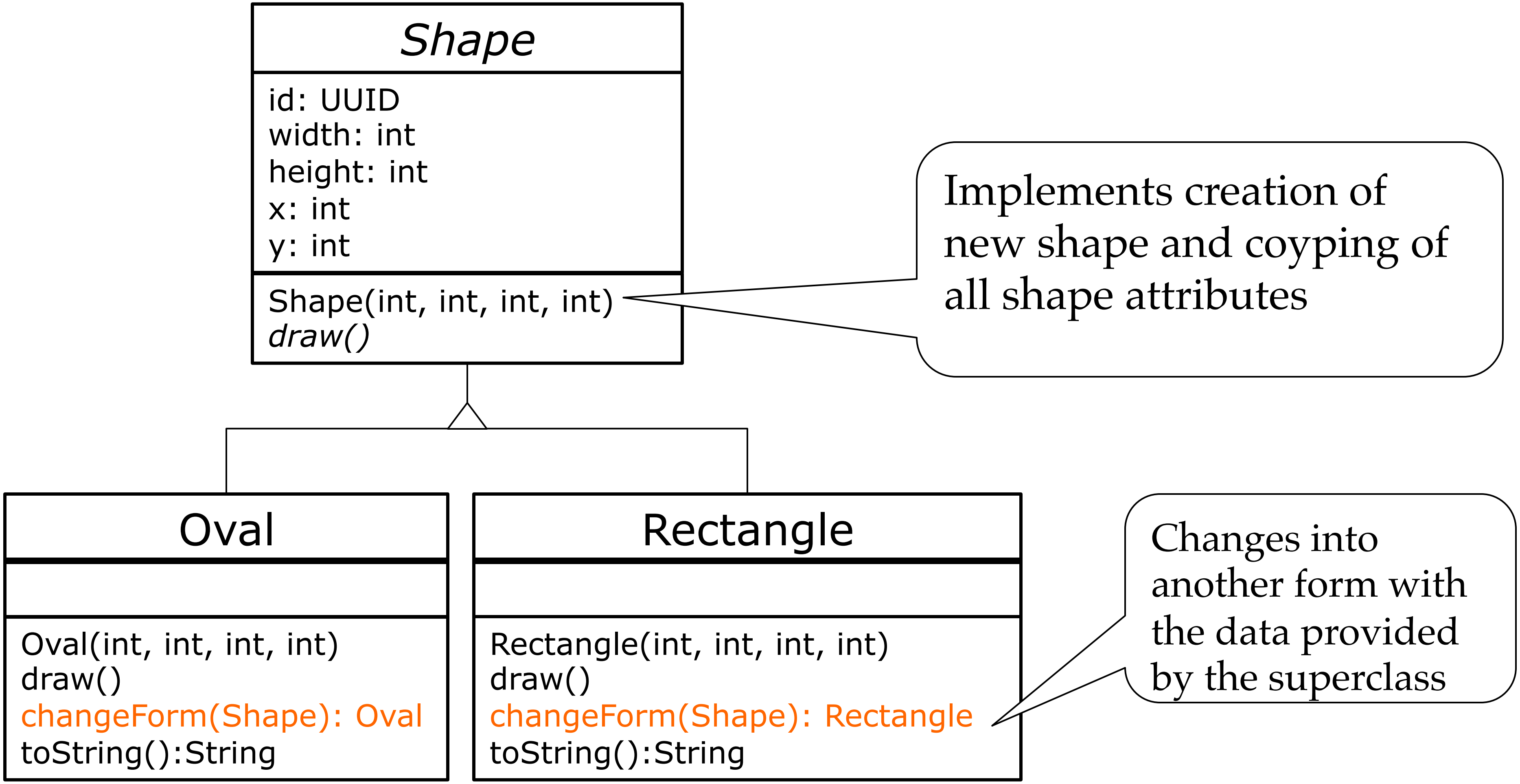

Draw subsystem
Oval
Rectangle
Circle
Change subsystem
ShapeChanger
Oval changeRectangleToOval(Rectangle)
Rectangle changeOvalToRectangle(Oval)
Circle changeRectangleToCircle(Rectangle)
Circle changeOvalToCircle(Oval)
Oval changeCircleToOval(Circle)
Rectangle changeCircleToRectangle(Circle)

Task #1: Adding Code for Changing Shapes

15 min

- Re-engineer the legacy code to produce a refactored solution
  - Step 1: Introduce an abstract Shape class
  - Step 2: Introduce a method changeForm(Shape) that transforms one shape into another
  - Step 3: Delete ShapeChanger
    - The new solution does not use the ShapeChanger class anymore
- Benefit
  - Shape instance can be changed at runtime
  - After you are done, adding a new shape subclass should be possible without changing the code for the existing shapes.

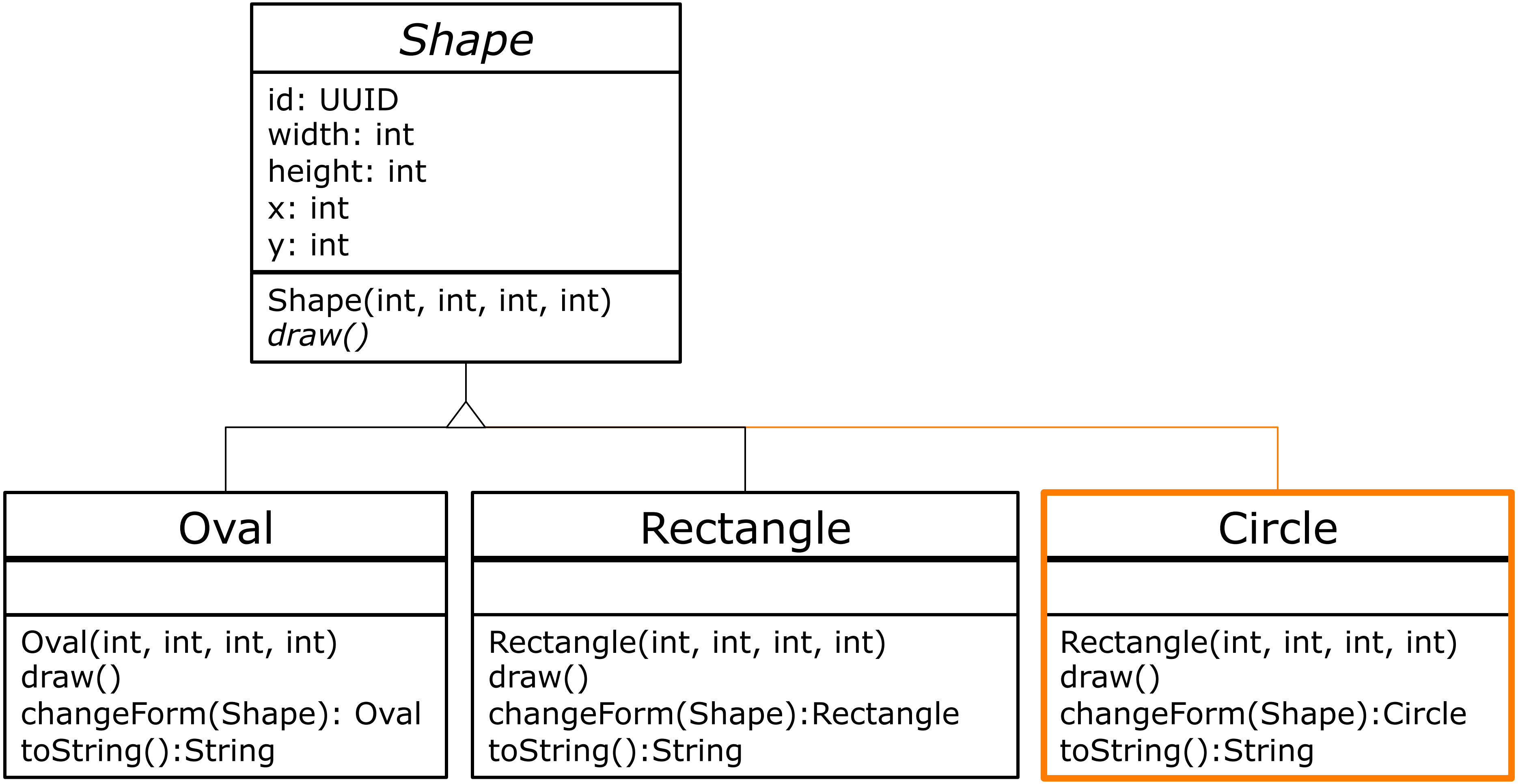
Task #1: Adding Code for Changing Shapes



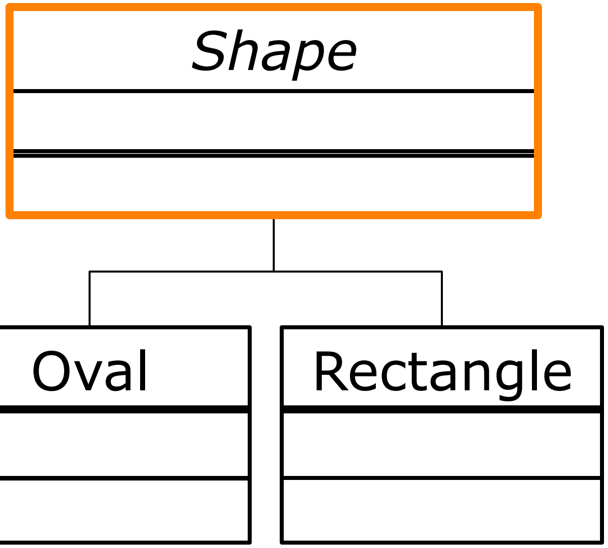
Implements creation of new shape and coyping of all shape attributes

Changes into another form with the data provided by the superclass

Task #1: Adding Code for Changing Shapes



Solution Task #1



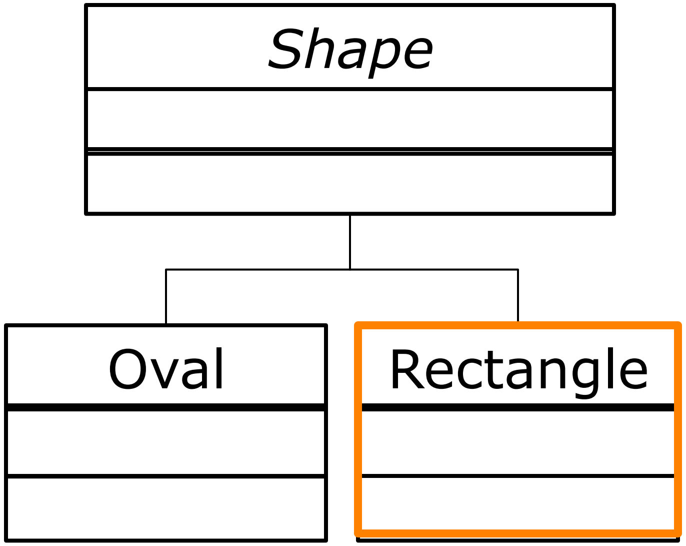
```
public abstract class Shape {
    protected int width;
    protected int height;
    protected int xCoordinate;
    protected int yCoordinate;
    protected UUID id;

    public Shape(int width, int height, int x, int y) {
        id = UUID.randomUUID();
        this.width = width;
        this.height = height;
        this.xCoordinate = x;
        this.yCoordinate = y;
    }

    public abstract void draw();
}
```

Extract attributes

Set all attributes in constructor

Add an abstract method for drawing.

Solution Task #1

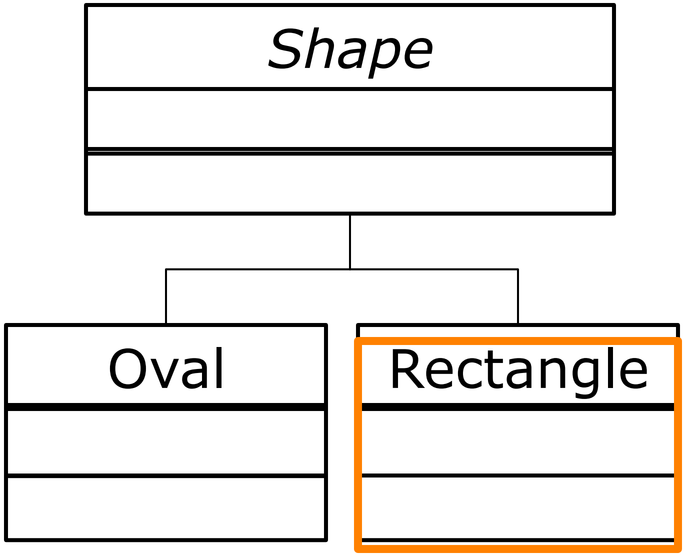


```
public class Rectangle extends Shape {

    public Rectangle(int width, int height, int x, int y) {
        super(width, height, x, y);
    }

    @Override
    public void draw() {
        System.out.println("Drawing: " + this.toString());
    }

...
```

Create Rectangle using constructor of superclass.

Implement draw method. Here we only write the object info to the console.

Solution Task #1



```
...

    @Override
    public String toString() {
        return "Rectangle [width=" + width + ", height=" + height
                + ", xCoordinate=" + xCoordinate + ", yCoordinate="
                + yCoordinate + ", id=" + id + "]";
    }
```

Override the toString method to show all relevant information.

```
    public static Rectangle changeForm(Shape s) {
        return new Rectangle(s.width, s.height, s.xCoordinate,
s.yCoordinate);
    }
}
```

Create a new Rectangle to change any Shape into a Rectangle.

Solution Task #1

```java
public static void main(String[] args) throws Exception {
    // change oval to rectangle
    Oval o = new Oval(10,20,30,40);
    System.out.println("Changing oval to rectangle");
    System.out.println("Before change: " + o);
    Rectangle resultingRectangle  = Rectangle.changeForm(o);
    resultingRectangle.draw();

    System.out.println();

    // change rectangle to oval
    Rectangle r = new Rectangle(1,2,3,4);
    System.out.println("Changing rectangle to oval");
    System.out.println("Before change: " + r);
    Oval resultingOval = Oval.changeForm(r);
    resultingOval.draw();
}
```