LISKOV SUBSTITUTION PRINCIPLE

LSP

- If `S` is a subtype of `T`, then objects of type `S` may be substituted for objects of type `T`, without altering any of the desired properties of the program.
- “`S` is a subtype of `T`”?

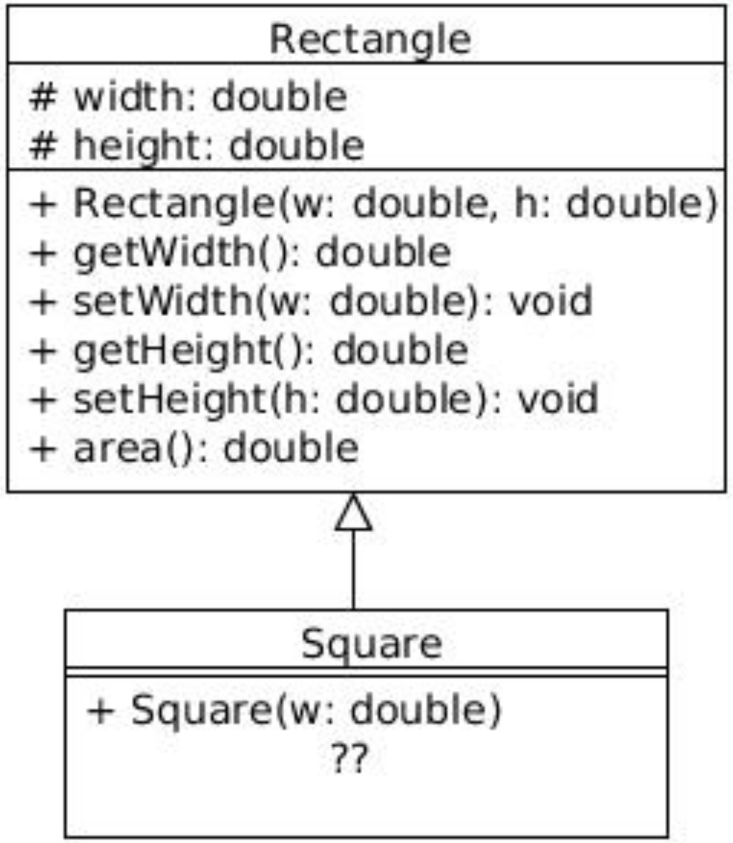
  In Java, this means that `S` is a child class of `T`, or `S` *implements* interface `T`.

- “A program that uses an interface must not be confused by an implementation of that interface.” [Uncle Bob]

LISKOV SUBSTITUTION PRINCIPLE

Example

- Mathematically, a square “is a” rectangle.
- In object-oriented design, it is not the case that a `Square` “is a” `Rectangle`!
- This is because a `Rectangle` has *more* behaviours than a `Square`, not less.
- The LSP is related to the Open/Closed principle: the subclasses should only extend (add behaviours), not modify or remove them.



Rectangle

width: double
height: double

+ Rectangle(w: double, h: double)
+ getWidth(): double
+ setWidth(w: double): void
+ getHeight(): double
+ setHeight(h: double): void
+ area(): double

Square

+ Square(w: double)
??