

Pizza Mania: The best little pizzeria in Texas

- Pizza Mania is having a special on the same kind of pizza (i.e. same thickness and same ingredients). The special comes in two shapes: rectangular (4 inches by 6 inches) for \$4.49 and circular (5 inches in diameter) for \$4.69.
- Which one is the better deal?
 - *How would you write a program to solve this problem?*

What is a “better deal”?

- I *model* the “better deal” concept by comparing the ratio price/area .

OOP Principle No. 0

Objects are the only things that can perform computations.

- I design a Pizza object that can tell me its price and its area.
- The algorithm to compute the area depends on the shape of the Pizza.
 - The Pizza's shape is what we call a "variant".

OOP Principle No. 1

Encapsulate all related variants into an “abstract class”.

- I create `AShape`, an abstract class, to represent the union of all possible shapes, each with its own way of computing its area.
- Classes `Circle` and `Rectangle` are concrete variants of `AShape`.
 - This is an example of what we call the *union pattern*.
 - How does a `Pizza` object compute its area?

OOP Principle No. 2

Program to the (abstract) interface..

- A Pizza object maintains a reference to an abstract AShape object, which, at run time, should be an instance of a specific concrete subclasses of AShape.
 - For this reason, AShape is said to be *polymorphic*.
- At construction time, we hand a Pizza object its price and a concrete AShape instance. The Pizza object stores its price and maintains a reference to the given AShape instance.
 - The Pizza object will forward all requests to compute its area to its AShape reference.

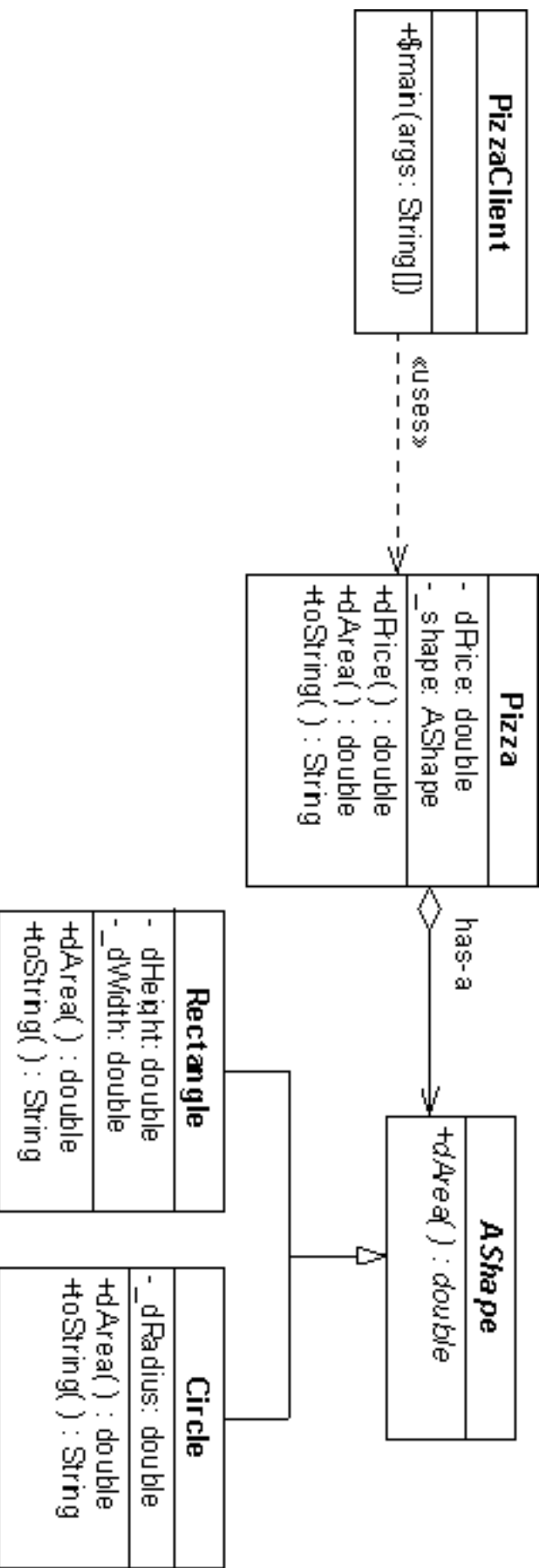
Commentary

- It is important to realize that the Pizza object does **not** know (and does **not** care about) what kind of concrete shape its AShape reference is. This has the effect of reducing the “*coupling*” between the Pizza class and the concrete subclasses of AShape.
 - There is no conditional statement to check for the specific type of shape for the Pizza object to compute its area. This has the effect of reducing code complexity.
 - * This a design pattern called the *strategy pattern*.

The Strategy Pattern

- In general, the strategy pattern consists of a union pattern of *strategies*, and a client class, called the *context*, that contains a reference to the abstract strategy in the union. In our Pizza example, the context is the Pizza class, and the abstract AShape plays the role of the (abstract) strategy.

Pizza



Java Programs

- A Java program consists of one or more classes.

- One of them must be `public` and must have a method with the following *signature*:

```
public static void main(String[] args)
```

- The main method will *instantiate* appropriate objects and send them “messages” (by calling their methods) to perform the desired tasks.