Objects - Classes

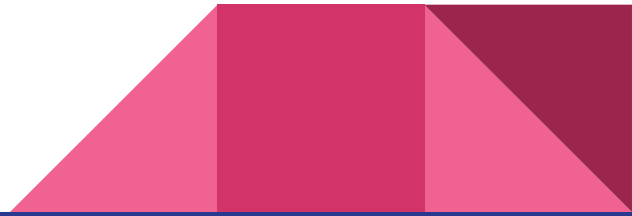
Part 1

Intro to objects

An **object** can be defined as a data field.

An **object** has both **state** and **behavior**.

- **State**: data about that object
- **Behavior:** actions that can be performed on the object.

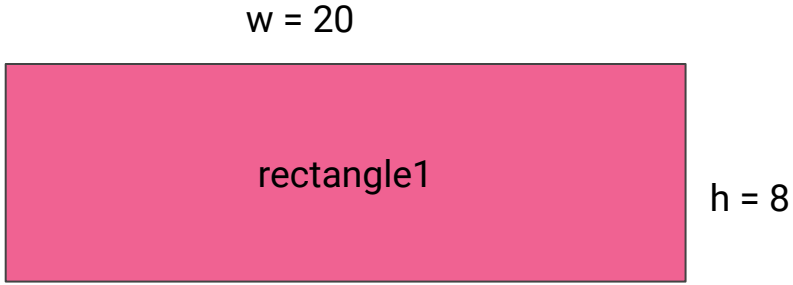


Object Oriented Programming

This rectangle is an example of an object.

A rectangle has some **attributes**: height and width

The **state** is data about that object. In this case we know that rectangle1 has a width of 20 and height of 8.



The **behavior** of an object is the actions that can be performed on the object. What could be an action for rectangle1?

Computing the area or the perimeter.

Object Oriented Programming

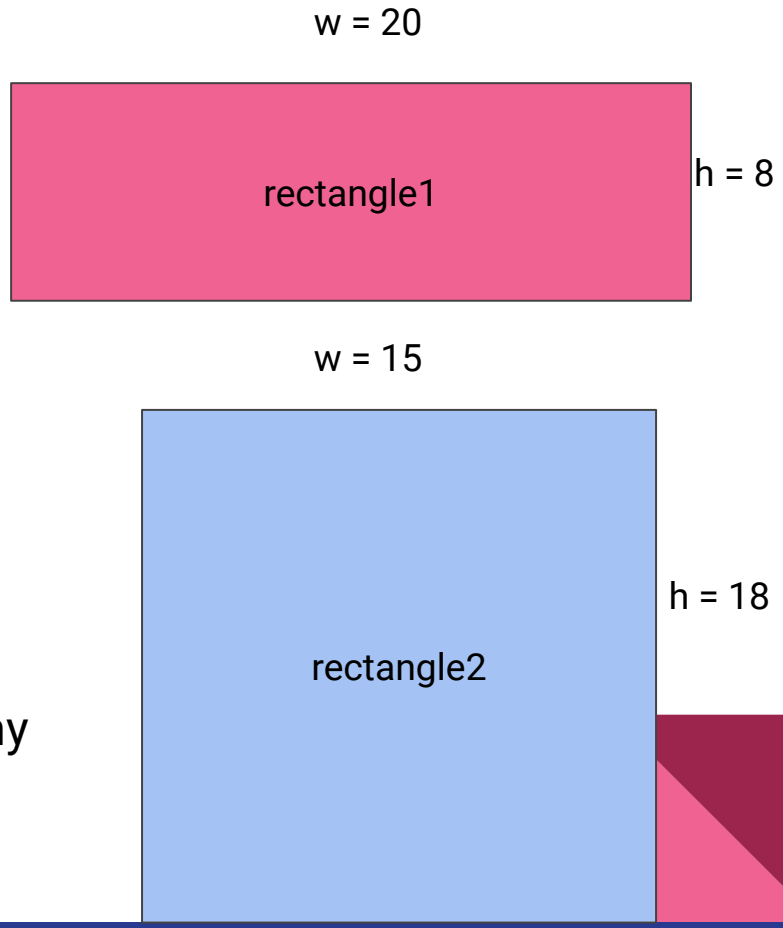
This rectangle is an example of an object.

We can know some things about it like its height and width.

Here is another rectangle:

This is a different object that has different height and width.

If both are rectangle objects, they should have many of the same behaviors and the same types of state but not necessarily the same state values.



Intro to Classes

Objects are created from **classes**.

A **class** is a template for creating an object. It holds information about what a class object state and behavior can be and how can be used in programs.

For our rectangles example:

- We must create a class rectangle

The Rectangle Class

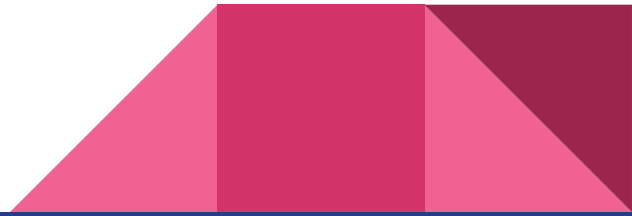
The Rectangle class:

{

Every rectangle must have a width and height.

The area and perimeter of every rectangle can be computed using values of width and height.

}



Rectangle Class

The class is the template or skeleton used to create an object. When filled out correctly, the result is an object itself

Rectangle Object
Template (Class)

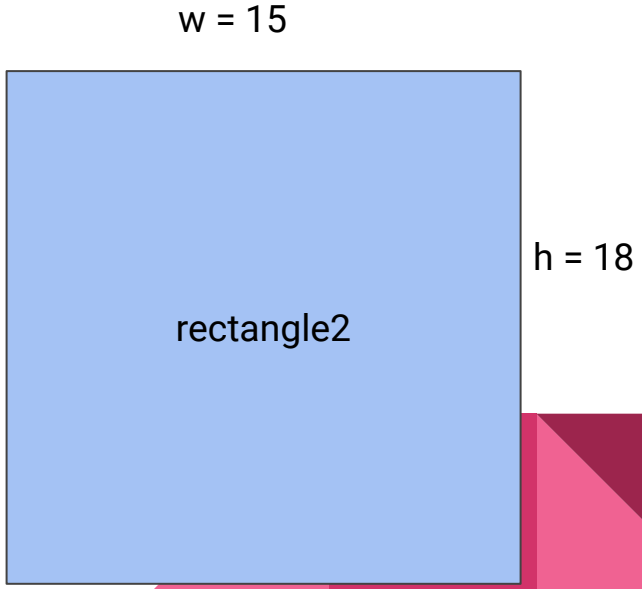
objectName

width	
height	

Rectangle Object

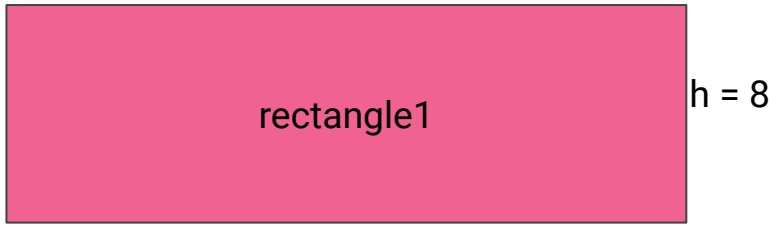
rectangle2

width	15
height	18

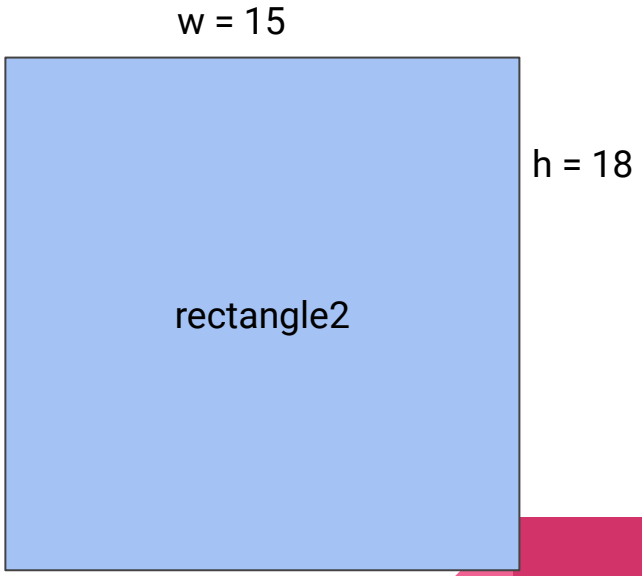


Rectangle object

When an object is created, it is referred to as an **instance** of the class that it belongs to. Separated of other objects from the same class.



width	20
height	8

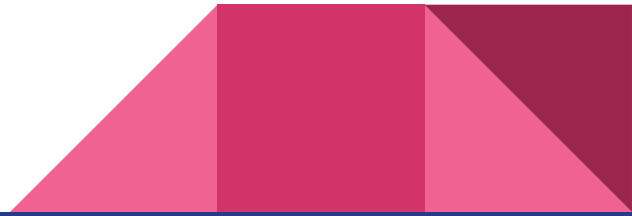


width	15
height	18

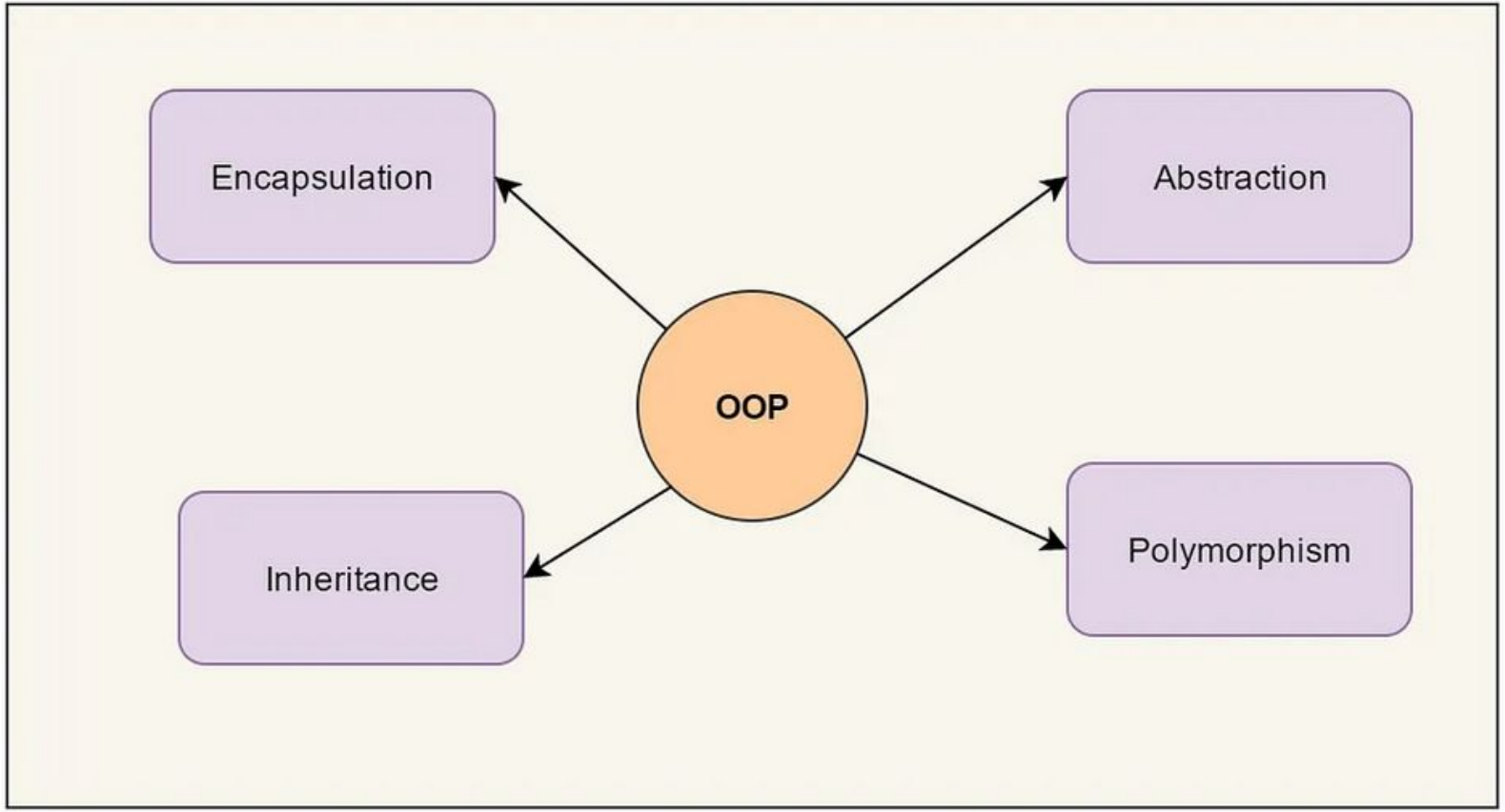
Object Oriented Programming

Using object type variables in programming is often referred to as Object-Oriented Programming (OOP).

OOP is a computer programming model that relies on the concept of **classes** and **objects**. It is used to structure a software program into simple, reusable pieces of code blueprints (classes), which are used to create individual instances of objects.



Four Principles of Object Oriented Programming

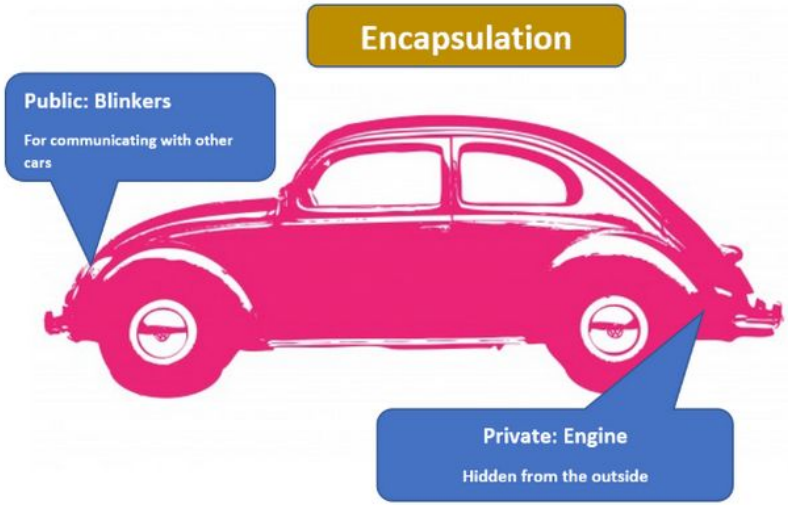


OOP Principle: ENCAPSULATION

- It is the mechanism of hiding important data implementation. Only selected information is exposed to the outside world.

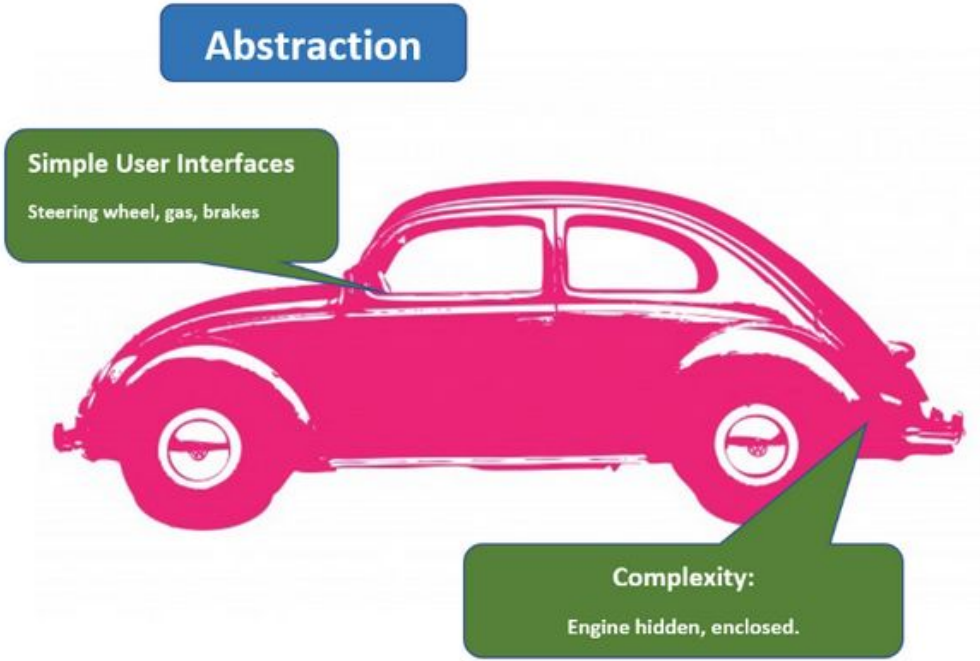
  - Instance variables are kept private
  - Getters and Setters methods are made public

- This adds a layer of security where the developer chooses what data can be seen on an object by exposing that data through public methods in the class definition.



OOP Principle: ABSTRACTION

- It express the intent of the class rather than the actual implementation.
- One class should not know the inner details of another in order to use it.



OOP Principle: INHERITANCE

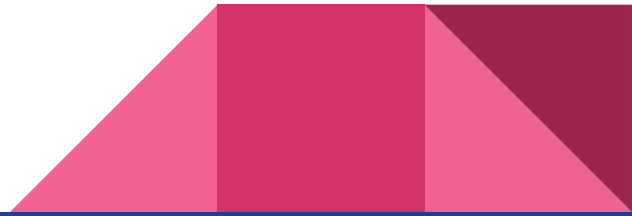
Child classes inherit data and behaviors from the parent class.

OOP Principle: POLYMORPHISM

Many methods can do the same task.

Class Structure

- Fields, also known **instance variables**.
- Methods (getters, setters, other).
- Constructors: needed to create objects.



Declaring Classes

public class Rectangle {

1. Instance Variable

2. Constructors

3. Methods

}

Access Modifiers:

Public: Other classes can access the field

Private: Only its own class can access the field

Exercise

A big part of using classes in Java is thinking about the design of the class. You'll need to figure out what information needs to be in the blueprint. For instance, in our Rectangle class, we needed to know the width and height.

This exercise is a free response question. Imagine that someone comes to you and asks you to design a class that represents a Pizza.

1. What instance variables should the Pizza class have?
2. What do the instance variables represent? What are the types of those instance variables?
3. Methods

Save here:
.../APCSA_1/apcsa-assignments-YourUsername/classwork/10_classes/pizza_class.txt