Object-Oriented Programming 2

Jerry Zhu

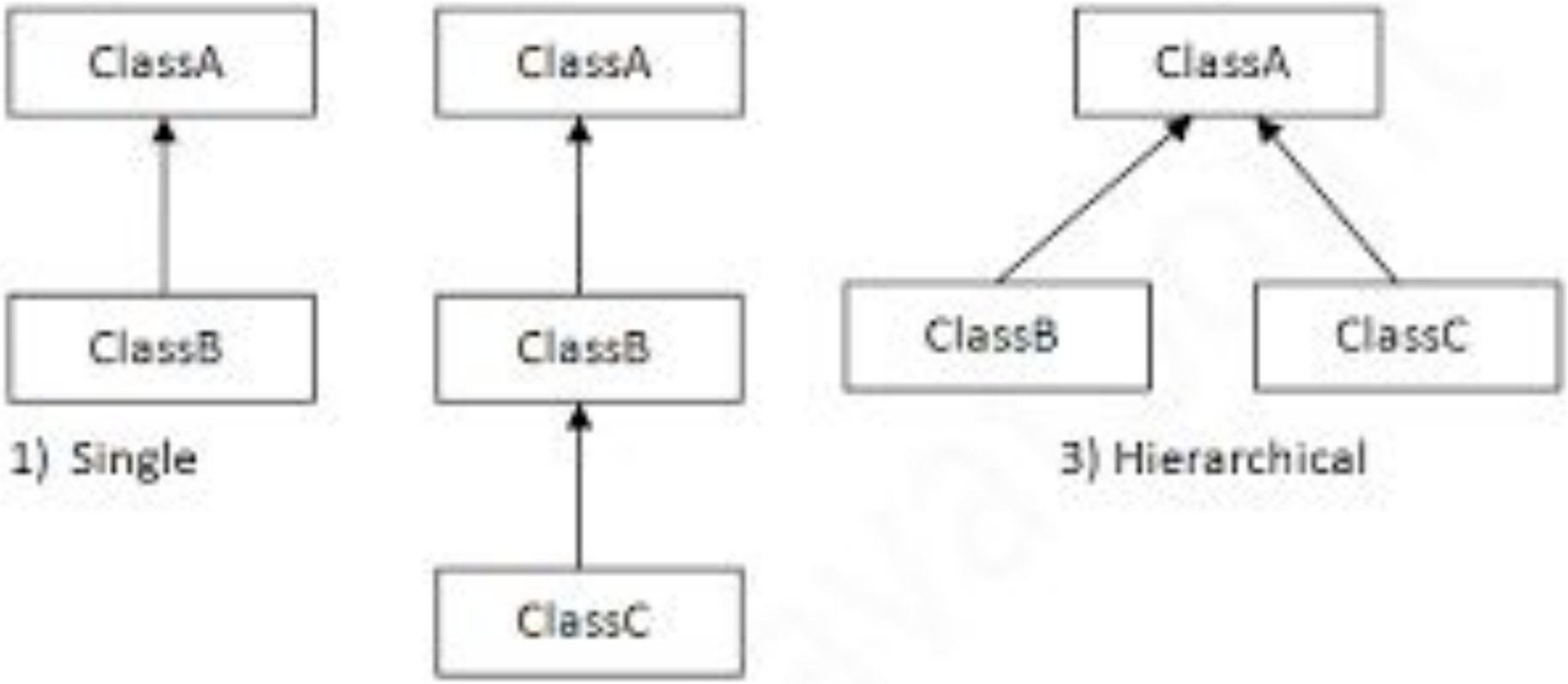
Summary of OOP1

- Literally, it is a style of programming based off of real life objects that contain data and do things on each other.
- Classes are blueprints to make objects.
  - Variables store an attribute of a class.
  - Methods are actions of a class.
    - Parameters are stored in the () of the method header and act as a local variable. A return method returns a value of the specified type.
  - Constructors are special methods to make an object out of a class.
  - Overloaded methods are methods with the same name but different paramenters.

Inheritance

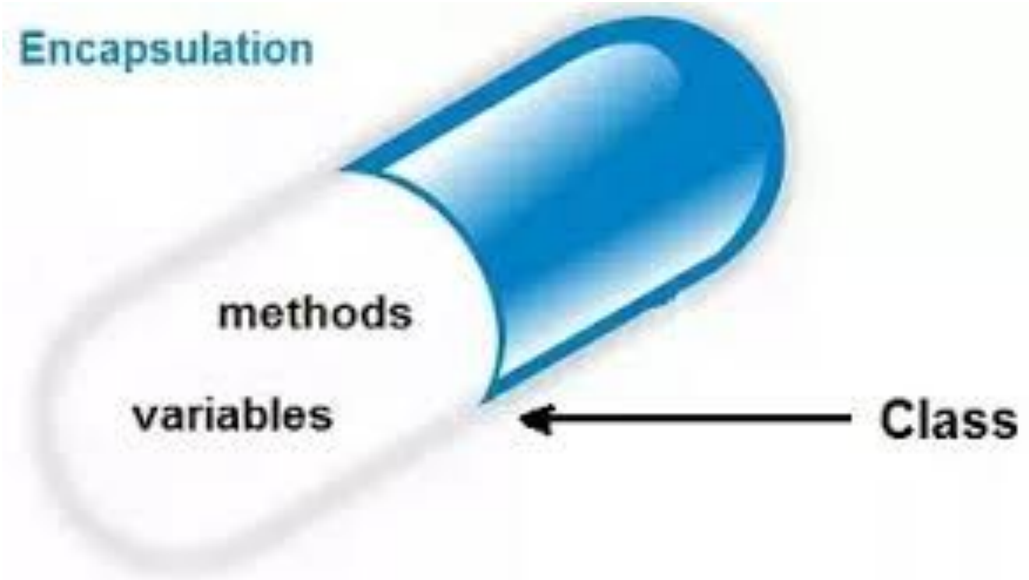
- Inheritance describes how the attributes and public methods are inherited by the subclass from its superclass/parent class.
- Think of the subclass is a specialized superclass.
- To me, the value of inheritance is that it removes the need for repetitive code. If two classes share many similarities, inheritance eliminates the need to write the class for them individually. For example, I do not need to write separate Shape and Rectangle classes by declaring the same variables and methods that Shape has in Rectangle again.

Inheritance Cont.



Encapsulation

- In Java, encapsulation is when an object’s attributes(instance variables) and its behaviours(methods) are wrapped together in class.
- To visualize this, think of a capsule(class) in which it stores all parts of an object.
- Its significance is that it allows classes to make objects, allowing OOP to work.



Information Hiding

- In Java, one uses information hiding through declaring the fields to be private.
- This is key because often times in OOP, the programmer must make numerous classes and have objects cooperate in a very intertwined manner, and sometimes, if the fields are all public, the information can be confused(local information is used in other objects, etc.)
- Information comes in to solve the problem through limiting the **scope** of data so that they remain hidden from others to avoid confusion.
  - So they can be changed without any ill effects (loose coupling).

Getters, Setters, Mutators

- Although the instance variables of an object needs to be private for information hiding, their value needs to be accessed or modified sometimes.
- This is done through these 3 types of methods.
  - Getters: methods to return the value of a field.
  - Setters: methods to set the value of a field.
  - Mutators: methods to alter the value of a field in some way without setting a completely new one.

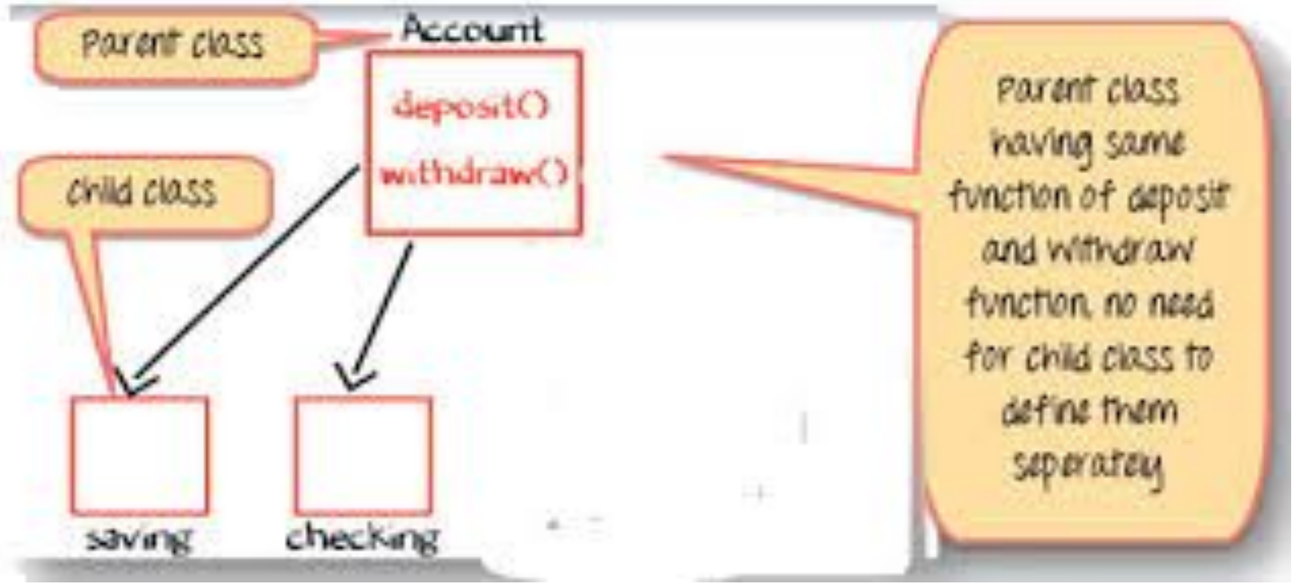
Examples of the Said Methods

Getters	Setters	Mutators

public double getHeight()
{
  return height;
}
	public void setHeight(double h)
	{
	  height = h;
	}
		public void growTaller()
		{
		  height++;
		}

Polymorphism

- A polymorphism is the practice of declaring an object as a type of another compatible object.
- It is significant in that it allows us to create versatile software designs.
- An object reference can refer to an object of its class, or to an object of any class related through inheritance.



Polymorphism Cont.

- I find that the most application for it is that say you have an array of Animal objects. In it, using polymorphism, you can store Ducks, Dogs, etc. If you wanted to make them all use the method speak(), it will do the speak() method for each type(E.g. the dog barks, the duck quacks...)

Abstraction

- Abstraction is the practice of selecting data from a larger pool to define, specifically, for the the object.
- The most common usage of this is when a superclass is declared abstract, waiting for its subclasses to define specific characteristics for it.
- Example: Shape class that came up in the UMLs is the best example for me.
- Think about it logically. “Shape d = new Shape();” doesn’t really sense.That is why we use abstraction. We let its subclasses make objects:

  Triangle t = new Triangle(); Circle c = new Circle();
- The Shape class would have abstract methods like calculateArea(). Since each shape has different formulas, we let the specific shapes define that method.

Personal Reflection

- OOP1 laid the foundation of Object-Oriented Programming by explaning the very basics(classes, objects, constrcutors, etc.) OOP2 is where we start to delve into high-level OOP.
- Of the concepts, I found polymorphism to be the most difficult to understand because for a long time, I could not see the value in it. Otherwise, this unit was more intuitive than the others but then again, this has been only the tip of the iceberg the broad field of computer science.
- I did not have too much trouble with the Schools assignment in OOP2 apart from that the instructions get confusing for me sometimes.