Encapsulation: is basically making a variable inside a class private, in order for it to be only accessed within the class, for example, you'll only be able to change the variable name if you use get and set functions inside the class. Encapsulation is used for a better control of your class attributes and methods.

Code example: (W3SCHOOLS SOURCE)

Example

```
public class Main {
  public static void main(String[] args) {
    Person myObj = new Person();
    myObj.name = "John";  // error
    System.out.println(myObj.name); // error
  }
}
```

Run Example »

```
MyClass.java:4: error: name has private access in Person
    myObj.name = "John";
          ^
MyClass.java:5: error: name has private access in Person
    System.out.println(myObj.name);
                       ^
2 errors
```

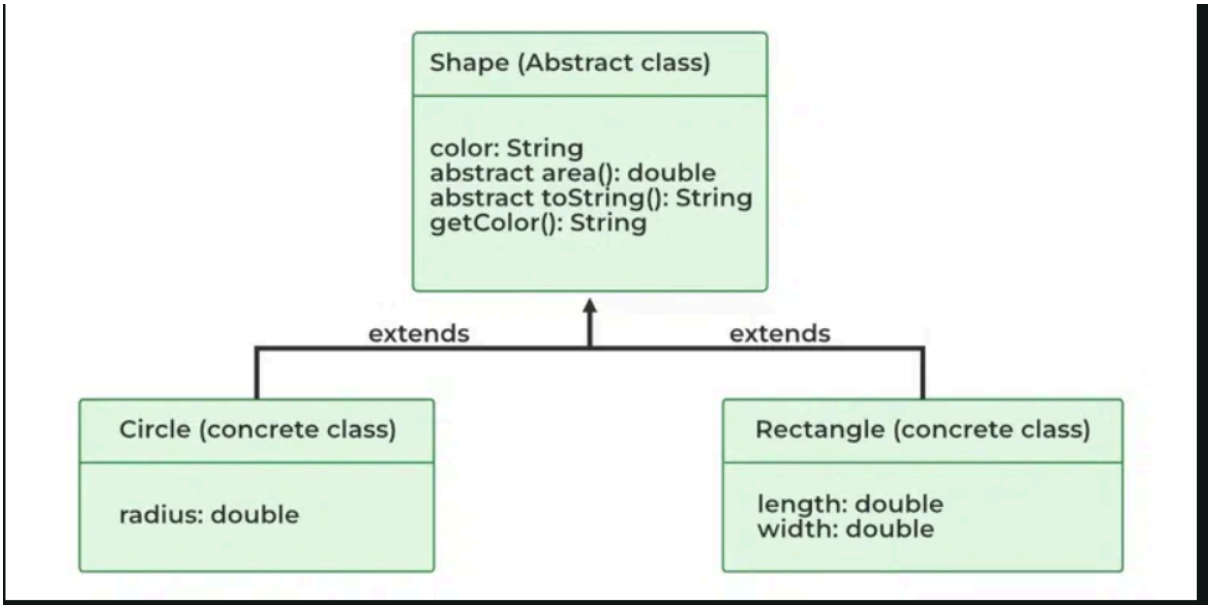
Inheritance: Inheritance is using a code to extend a class making another class, as an example, imagine you have a really long code and you wish to reuse a class you have, but don't want to go back and edit everything, you can simply extend the class by using class "Name of new class" extend "class you wish to extend". This term is divided by two terms, subclass, and superclass, subclass being the class that is extending, and superclass being the class that you already made.

w3schools example below:

```
class Vehicle {
  protected String brand = "Ford";        // Vehicle attribute
  public void honk() {                    // Vehicle method
    System.out.println("Tuut, tuut!");
  }
}

class Car extends Vehicle {
  private String modelName = "Mustang";    // Car attribute
  public static void main(String[] args) {

    // Create a myCar object
    Car myCar = new Car();

    // Call the honk() method (from the Vehicle class) on the myCar object
    myCar.honk();

    // Display the value of the brand attribute (from the Vehicle class) and the value of the modelName from the Car class
    System.out.println(myCar.brand + " " + myCar.modelName);
  }
}
```

Abstraction: Abstraction in java is somewhat linked with superclasses and subclasses, abstraction is simply making a superclass that has a common template for derived subclasses and is the process of hiding details and only showing the essential information to the user. For example, let's say you make a "Vehicle" superclass, and the superclass has multiple abstract methods without a body to see what the car does in case of each vehicle type, but let's say it has an "accelerate" abstract method, you'd have to make a abstract superclass that cannot be accessed directly in the main method, and you must have a subclass that inherits the Vehicle class, and it also defines what the methods inside your abstract class do, say in this case, you define an accelerate abstract method in your superclass with no body, and define what accelerating actually does in your subclass, which could be a Boat subclass, a Car subclass, or a Plane subclass.

Image taken from: https://www.geeksforgeeks.org/abstraction-in-java-2/



Polymorphism: Polymorphism is linked to the use of subclasses and superclasses, polymorphism means “many forms” and it means that a class, can have multiple forms using inheritance, for example i can have a superclass called “Animal” that has many subclasses, or “forms” like for example a Dog extension or a Pig extension subclass. This is used when you want to reuse code, that being your superclass.

w3schools example:

Example

```
class Animal {
  public void animalSound() {
    System.out.println("The animal makes a sound");
  }
}

class Pig extends Animal {
  public void animalSound() {
    System.out.println("The pig says: wee wee");
  }
}

class Dog extends Animal {
  public void animalSound() {
    System.out.println("The dog says: bow wow");
  }
}

class Main {
  public static void main(String[] args) {
    Animal myAnimal = new Animal();  // Create a Animal object
    Animal myPig = new Pig();  // Create a Pig object
    Animal myDog = new Dog();  // Create a Dog object
    myAnimal.animalSound();
    myPig.animalSound();
    myDog.animalSound();
  }
}
```