CSE 215: Programming Language II Lab

Sec – 8, Faculty - MUO

Lab Officer: Tanzina Tazreen

Lab – 9

Polymorphism

Polymorphism means "many forms". Polymorphism is the ability of an object to take on many forms. **Inheritance** lets us inherit attributes and methods from another class. **Polymorphism** uses those methods to perform a single action in different ways.

There are 2 kinds of polymorphism:

1. Compile-time polymorphism / method overloading
2. Run-time polymorphism / Method overriding

Method overriding

In inheritance, child class automatically inherits parent class method. But if a child class re-write one of parent class method (to better suit child class need), then it's called **method overriding.**

Rules for Java Method Overriding

1. The method must have the same name as in the parent class
2. The method must have the same parameter as in the parent class.

```
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();
  }
}
```

```
The animal makes a sound
The pig says: wee wee
The dog says: bow wow
```

Dynamic Binding

A method can be implemented in several classes along the inheritance chain. The JVM decides which method is invoked at runtime.

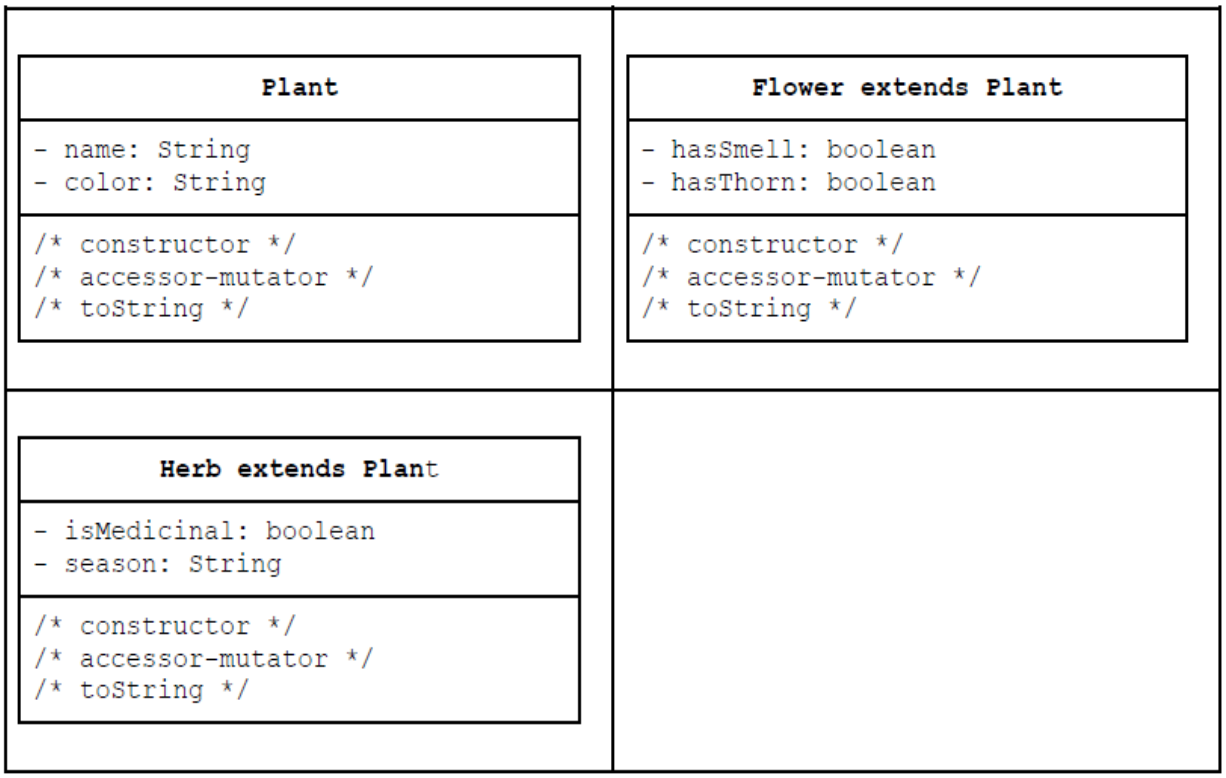
In the above example, Animal is the **declared type/ reference type**, and Pig/Dog is the **actual type.**

Object myPig 's actual type is Pig, and Animal is reference type. So, Which animalSound()method is invoked is determined by myPig's actual type (not Reference type).
the animalSound() method of the Pig class is called automatically, even though the declared type is Animal. **This is known as "dynamic binding" or "late binding", where the method to invoke along an inheritance chain is determined by the actual type of the declared object at runtime. And this is true for any number of classes along the inheritance chain.**

A subclass is a specialization of its superclass; every instance of a subclass is also an instance of its superclass, but not vice versa. It means that you can refer to a subclass using the superclass type.

Task:

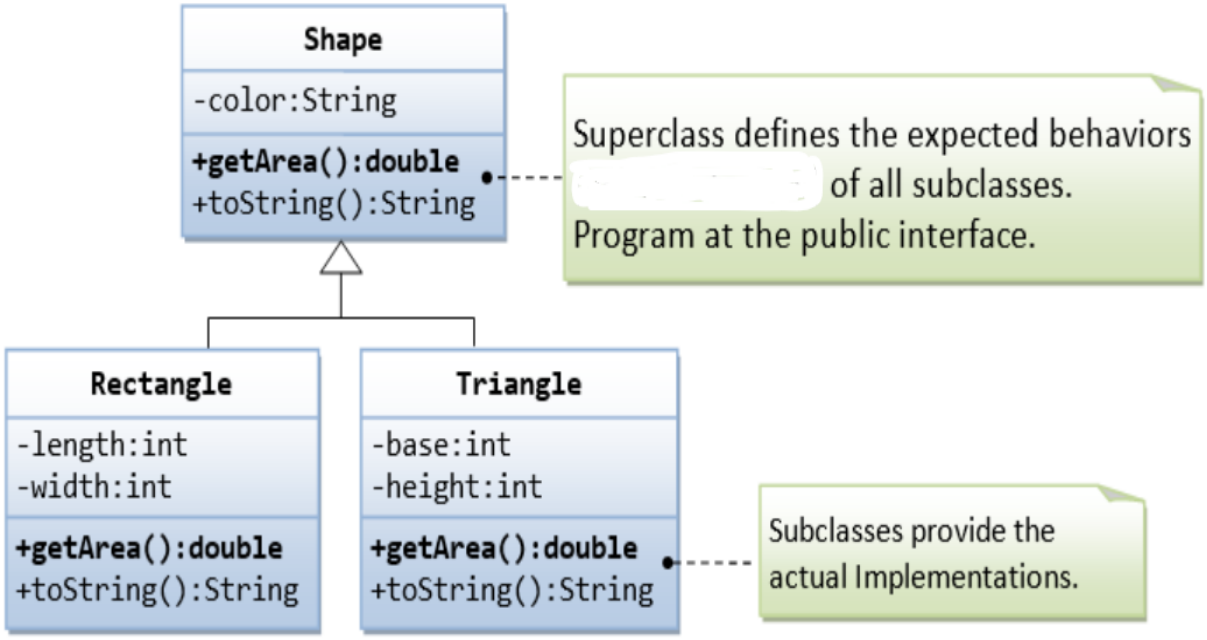
1.



In main method, create an array of Plant objects and implement these methods:
static void add(Plant [] plants, Plant p) // to add new plant object into the array
static void remove(Plant [] plants, String n) // remove a plant given its name
static Plant search(Plant [] plants, String n) // search for a plant given its name
static void display(Plant [] plants) // display all Plant objects

*** In Java **accessors are used to get the value of a private field and mutators are used to set the value of a private field**. Accessors are also known as getters and mutators are also known as setters***

2.



There are many kinds of shapes, such as triangle, rectangle and so on. We should design a superclass called Shape, which defines the public behaviors of all the shapes. For example, we would like all the shapes to have a method called getArea(), which returns the area of that particular shape.

At last, in the Driver class, create references of Shape, and assigned them instances of both super class and subclasses. Then call the getArea() and toString() for all these objects.