CSE 215: Programming Language II Lab

Sec – 8, Faculty - MUO

Lab Officer: Tanzina Tazreen

Lab – 10

Abstract Class & polymorphism

Abstraction in Java

Abstraction is a process of hiding the implementation details and showing only functionality to the user. it shows only essential things to the user and hides the internal details, for example, sending SMS where you type the text and send the message. You don't know the internal processing about the message delivery.

There are two ways to achieve abstraction in java

1. Abstract class
2. Interface

Abstract class

To create an abstract class, you need to declare a class abstract, using the **abstract** keyword.

- Abstract class has to have at least one abstract method (abstract method is a method with the keyword abstract and without a body, only the header).
- An abstract class may or may not have all abstract methods. Some of them can be concrete methods
- If an abstract class cannot be instantiated (you cannot create object of that class).
- To use an abstract class, you have to inherit it from another class using the keyword "**extend**".
- If you inherit an abstract class, you have to implement all the abstract methods in it. thus making overriding compulsory
- An abstract class can have parameterized constructors and the default constructor is always present in an abstract class.
- It can have static methods also.

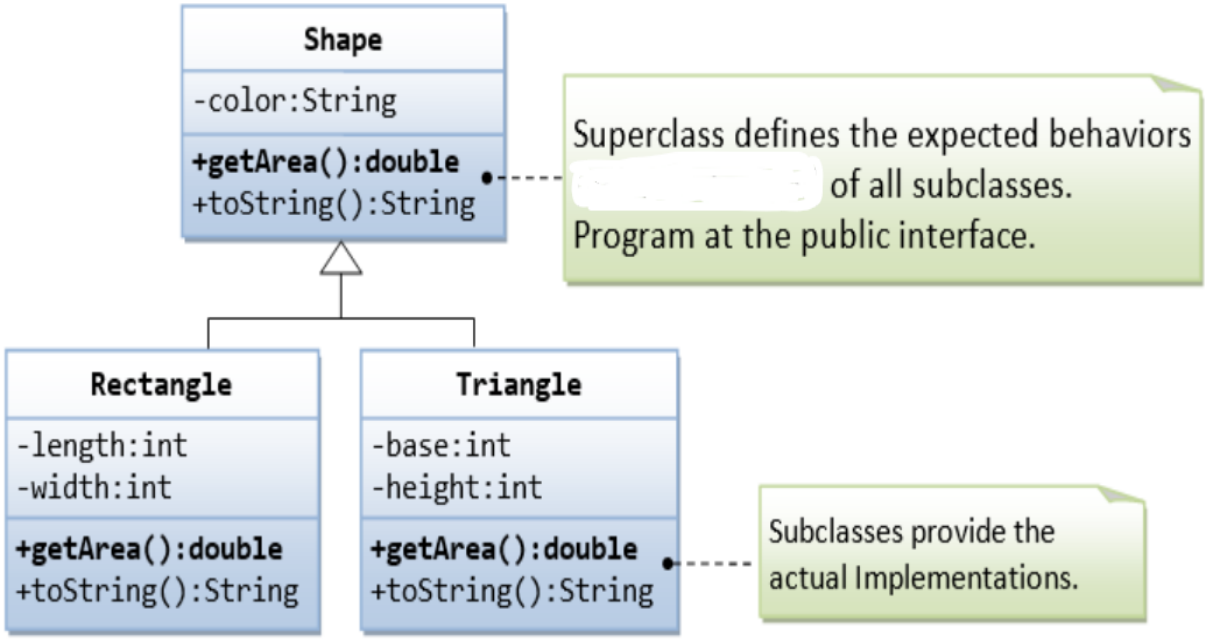
When to use abstract classes and abstract methods

There are situations in which we will want to define a superclass that declares the structure of a given abstraction without providing a complete implementation of every method. Sometimes we will want to create a superclass that only defines a generalization form that will be shared by all of its subclasses, leaving it to each subclass to fill in the details.

```
// Abstract class
abstract class Animal
{
  // Abstract method (does not have a body)
  public abstract void animalSound();
  // Regular method
  public void sleep()
  {
    System.out.println("Zzz");
  }
}

// Subclass (inherit from Animal)
class Pig extends Animal
{
  public void animalSound()
  {
    // The body of animalSound() is provided here
    System.out.println("The pig says: wee wee");
  }
}

class Dog extends Animal
{
  public void animalSound()
  {
    // The body of animalSound() is provided here
    System.out.println("The Dog says: vow vow");
  }
}


class Main {
  public static void main(String[] args)
  {
```

```
    //abstract class can be used to create an object
    // Animal an = new Animal();
    Pig myPig = new Pig(); // Create a Pig object
    myPig.animalSound();
    myPig.sleep();
  }
}
```

Tasks:



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.

Make the "Shape" class an abstract class by making "getArea()" an abstract method.

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.