

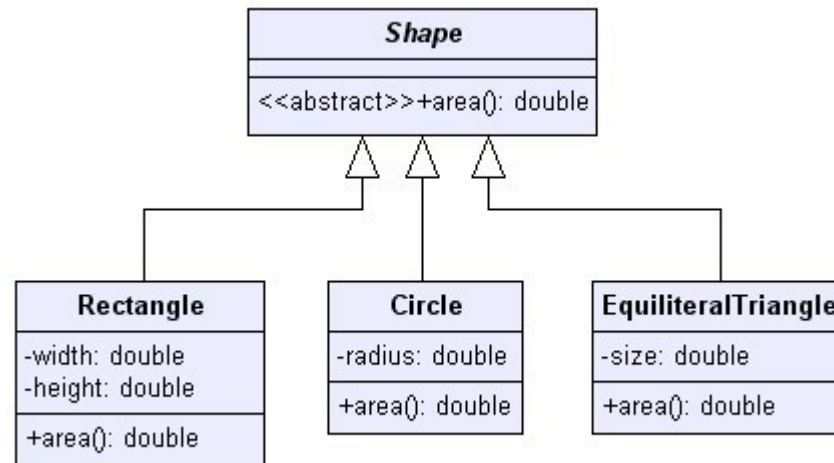
Abstract Class

Abstract Class

- An abstract class is a class we cannot instantiate due to unimplemented method (or methods) it includes.

Except for that limitation it is the same as any other class. The unimplemented method should be marked as an abstract method if it was declared within our abstract class. If it was inherited from one of the base classes then there is no need to add the 'abstract' keyword.

Abstract Class



Abstract Class

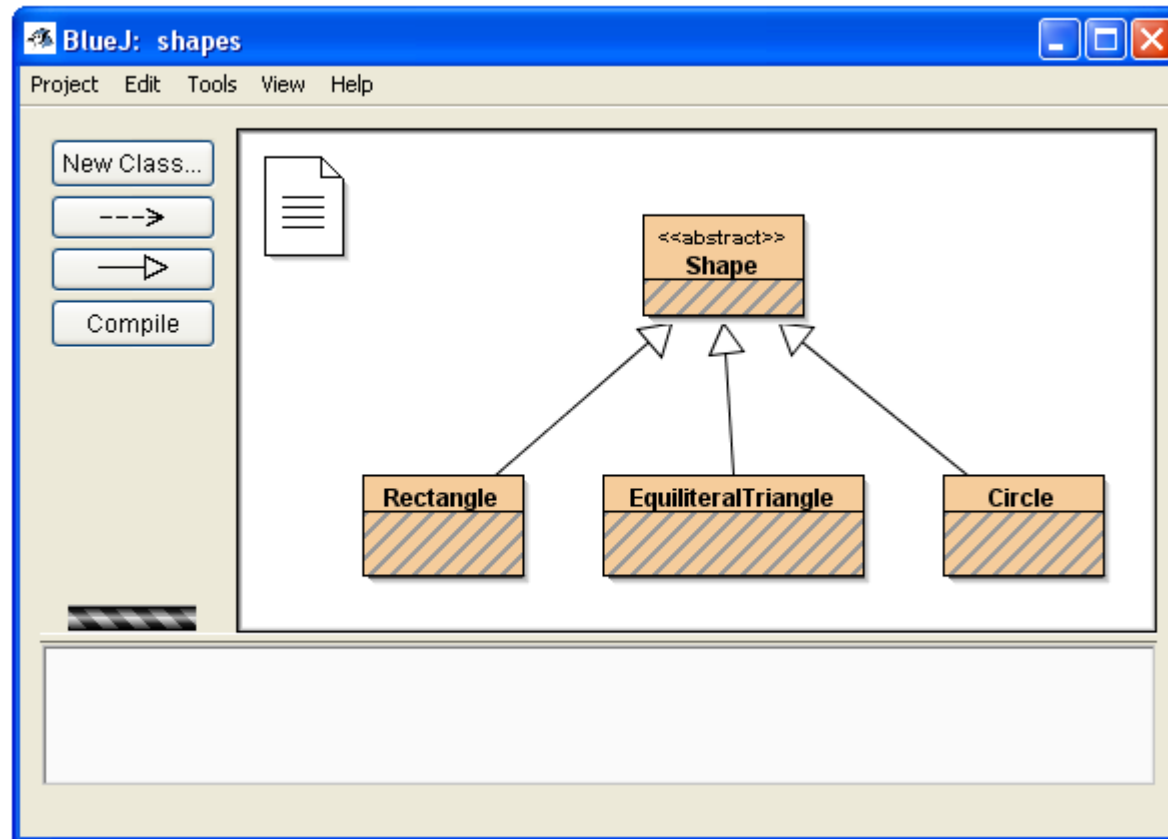
```
abstract class Shape
{
    abstract double area();
}

class Rectangle extends Shape
{
    private double width, height;
    Rectangle(double wVal, double hVal)
    {
        width = wVal;
        height = hVal;
    }
    double area()
    {
        return width * height;
    }
}
```

```
class Circle extends Shape
{
    private double radius;
    Circle(double radVal)
    {
        radius = radVal;
    }
    double area()
    {
        return 3.14 * radius * radius;
    }
}

class EquiTriangle extends Shape
{
    private double size;
    EquiTriangle(double val)
    {
        size = val;
    }
    double area()
    {
        return 0.433 * size * size;
    }
}
```

Abstract Class

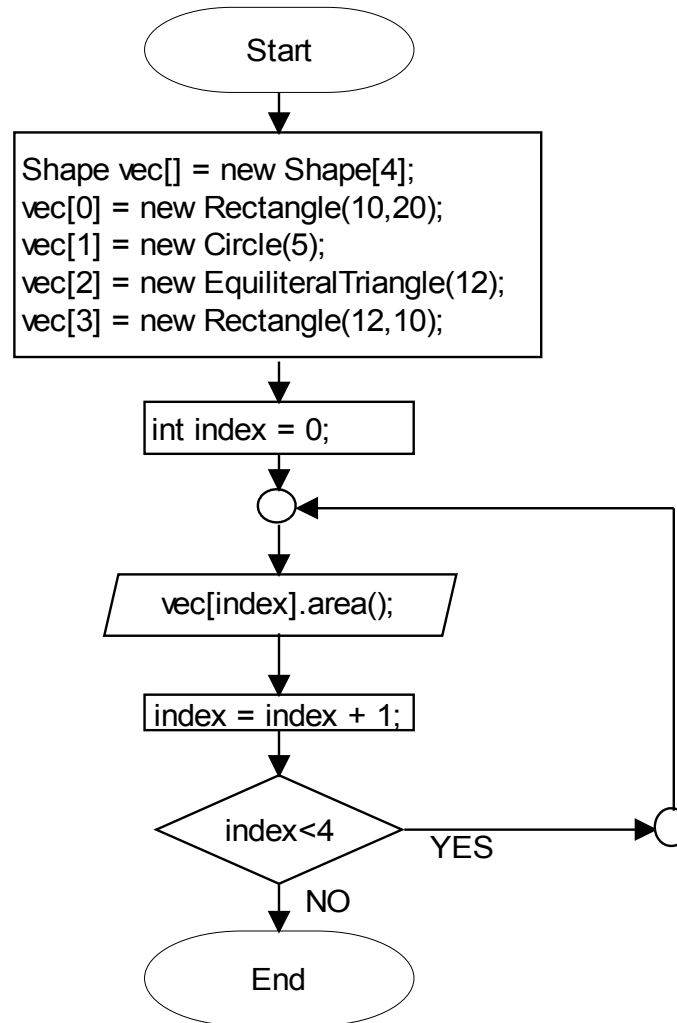


Polymorphism

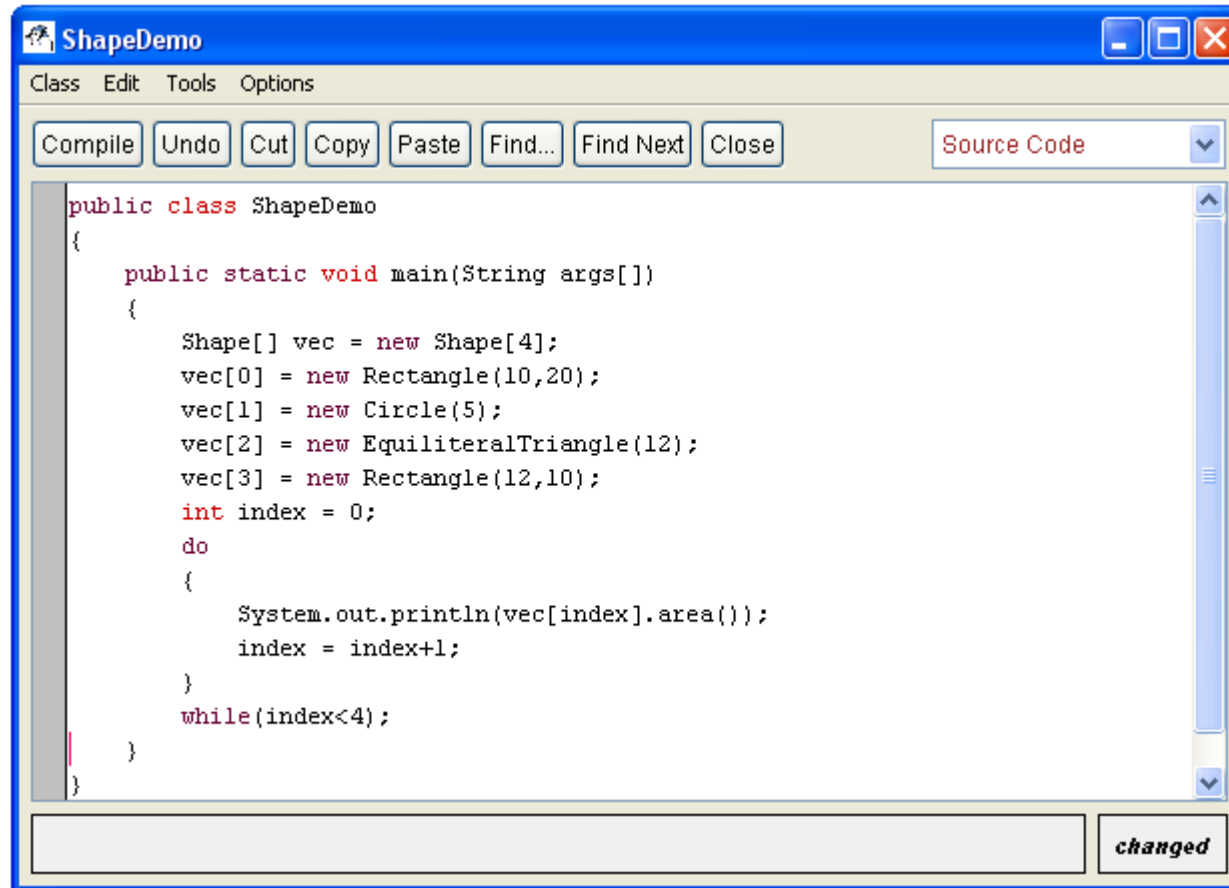
- Polymorphism is a code fragment that each time we execute it performs differently.
- The main purpose of having an abstract class is to allow us to implement polymorphism.

An abstract class is a type the same way any other class is. Having an abstract class we can use variables of its type holding references for objects instantiated from non abstract class that inherits from the abstract class.

Polymorphism



Polymorphism



```
public class ShapeDemo
{
    public static void main(String args[])
    {
        Shape[] vec = new Shape[4];
        vec[0] = new Rectangle(10,20);
        vec[1] = new Circle(5);
        vec[2] = new EquilateralTriangle(12);
        vec[3] = new Rectangle(12,10);
        int index = 0;
        do
        {
            System.out.println(vec[index].area());
            index = index+1;
        }
        while(index<4);
    }
}
```

changed