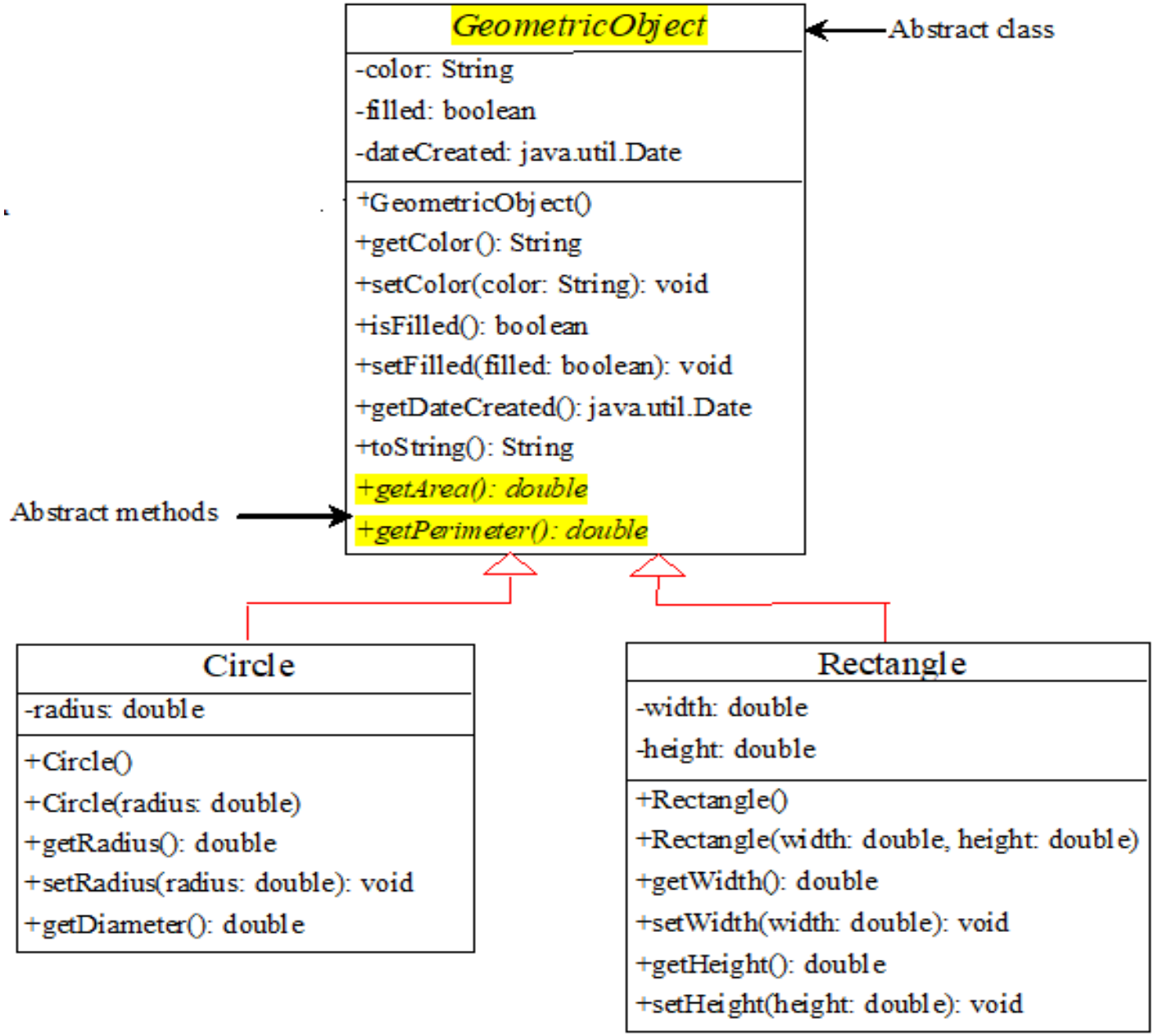
Abstract Class

Abstract class


GeometricObject
Abstract class
-color: String
-filled: boolean
-dateCreated: java.util.Date
+GeometricObject()
+getColor(): String
+setColor(color: String): void
+isFilled(): boolean
+setFilled(filled: boolean): void
+getDateCreated(): java.util.Date
+toString(): String
+getArea(): double
Abstract methods
+getPerimeter(): double
Circle
-radius: double
+Circle()
+Circle(radius: double)
+getRadius(): double
+setRadius(radius: double): void
+getDiameter(): double
Rectangle
-width: double
-height: double
+Rectangle()
+Rectangle(width: double, height: double)
+getWidth(): double
+setWidth(width: double): void
+getHeight(): double
+setHeight(height: double): void

Abstract class

- A class that contains abstract methods must be abstract
- An abstract method cannot be contained in a nonabstract class.
- If a subclass of an abstract superclass does not implement all the abstract methods, the subclass must be declared abstract.
- In other words, in a nonabstract subclass extended from an abstract class, all the abstract methods must be implemented, even if they are not used in the subclass.

Abstract class

- An abstract class cannot be instantiated using the new operator, but you can still define its constructors, which are invoked in the constructors of its subclasses.
- For instance, the constructors of GeometricObject are invoked in the Circle class and the Rectangle class.

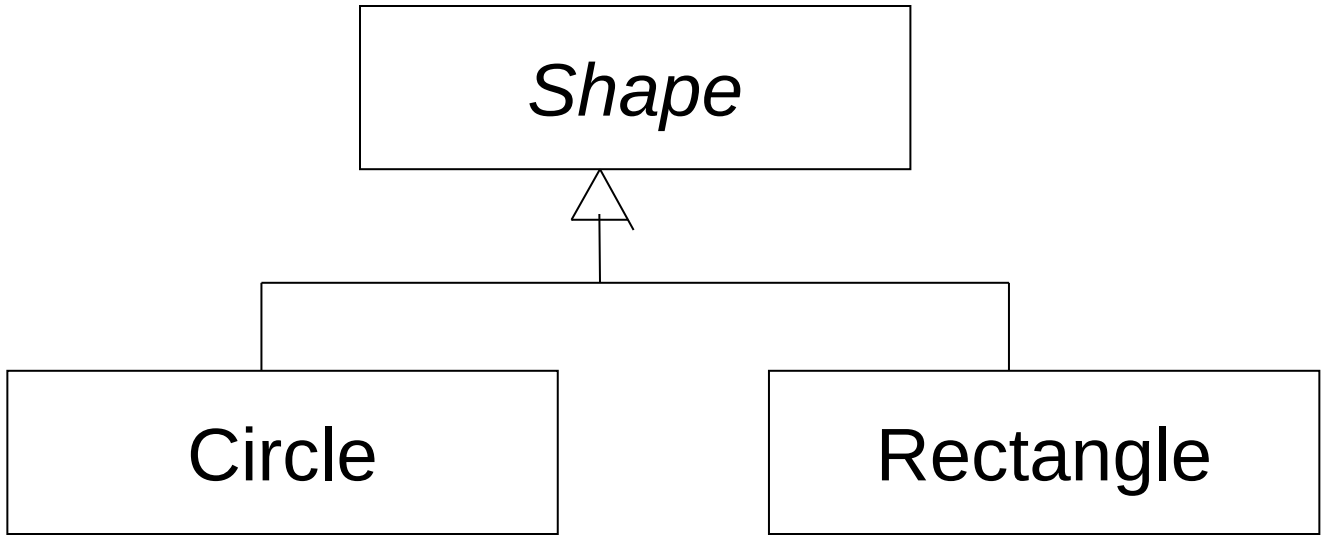
Abstract class

- You cannot create an instance from an abstract class using the new operator, but an abstract class can be used as a data type.
- Therefore, the following statement, which creates an array whose elements are of GeometricObject type, is correct.

```
GeometricObject[] geo = new    GeometricObject[10];
```

Example

- Shape is a abstract class.



Example

```
public abstract class Shape {
        public abstract double area();
        public void move() { // non-abstract method
            // implementation
        }
}
```

- Shape s =new Shape(); //wrong

Example

```
public Circle extends Shape {
   protected double r;
   protected static final double PI =3.1415926535;
   public Circle() { r = 1.0; )
   public double area() { return PI * r * r; }
...
}
public Rectangle extends Shape {
   protected double w, h;
   public Rectangle() { w = 0.0; h=0.0; }
   public double area() { return w * h; }
}
```

- Shape s =new Rectangle(); //ok
- Shape s=new Circle(); //ok