24. Abstract Classes

Java

Summer 2008

Instructor: Dr. Masoud Yaghini

Outline

- Abstract Classes
- References

Abstract Classes

Abstract Classes

- In the inheritance hierarchy, classes become more specific and concrete with each new subclass.
- If you move from a subclass back up to a superclass, the classes become more general and less specific.
- Class design should ensure that a superclass contains common features of its subclasses.
- Sometimes a superclass is so abstract that it cannot have any specific instances.
- Such a class is referred to as an ***abstract class***.

Abstract Classes

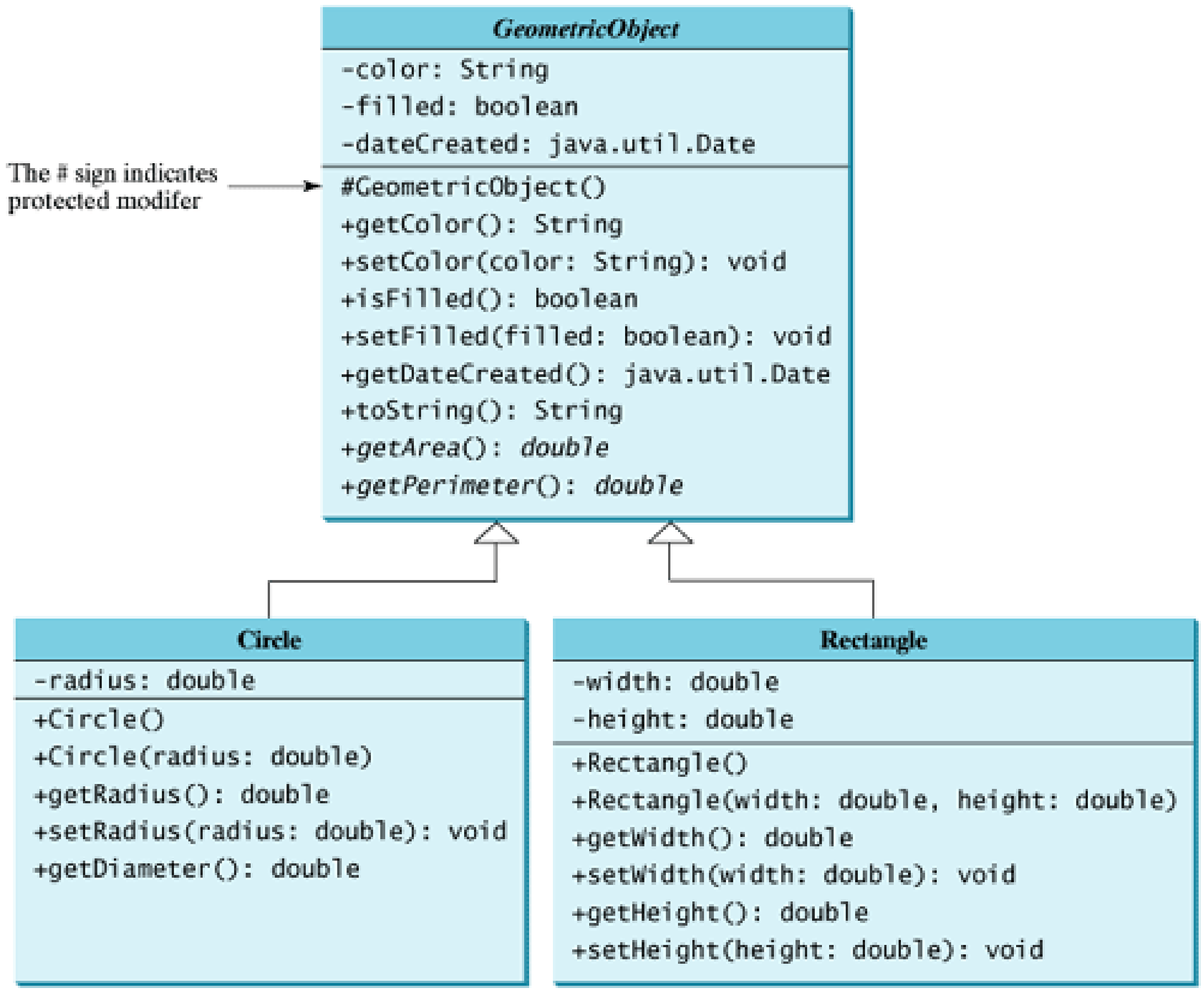
- In the preceding chapter we compute areas and perimeters for all geometric objects
- It is better to declare the getArea() and getPerimeter() methods in the GeometricObject class.
- These methods cannot be implemented in the GeometricObject class because their implementation is dependent on the specific type of geometric object.
- Such methods are referred to as ***abstract methods***.
- A class that contains abstract methods must be declared abstract.

The abstract Modifier

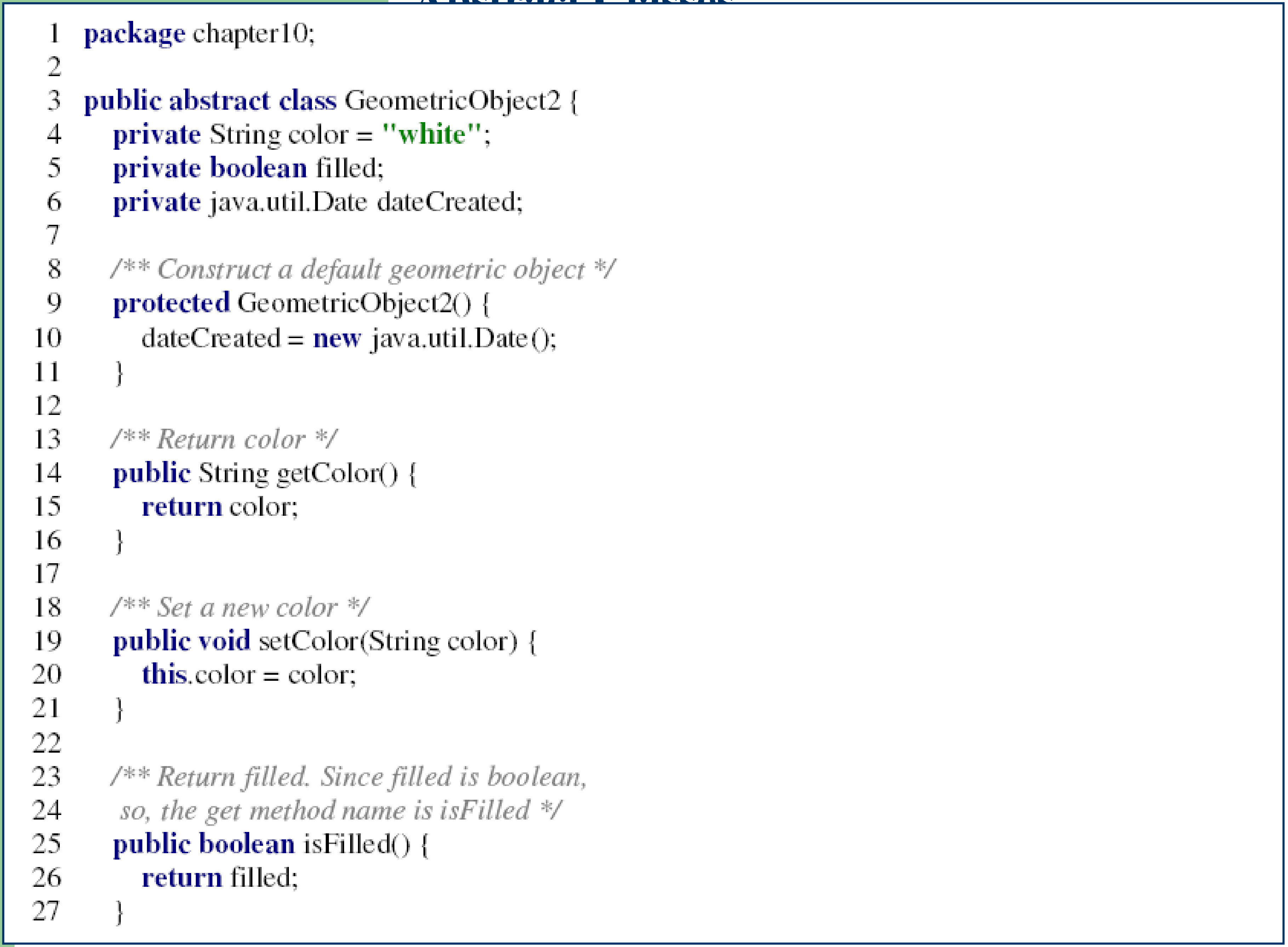
- The abstract class
  - Cannot be instantiated (you cannot create instances of abstract classes)
  - Should be extended and implemented in subclasses
- The abstract method
  - Method signature without implementation
  - Its implementation is provided by the subclasses.

Abstract Classes

The new GeometricObject class contains abstract methods



Abstract Classes



```
package chapter10;

public abstract class GeometricObject2 {
   private String color = "white";
   private boolean filled;
   private java.util.Date dateCreated;

   /** Construct a default geometric object */
   protected GeometricObject2() {
      dateCreated = new java.util.Date();
   }

   /** Return color */
   public String getColor() {
      return color;
   }

   /** Set a new color */
   public void setColor(String color) {
      this.color = color;
   }

   /** Return filled. Since filled is boolean,
    so, the get method name is isFilled */
   public boolean isFilled() {
      return filled;
   }
```

Abstract Classes

```

  /** Set a new filled */
  public void setFilled(boolean filled) {
    this.filled = filled;
  }

  /** Get dateCreated */
  public java.util.Date getDateCreated() {
    return dateCreated;
  }

  /** Return a string representation of this object */
  public String toString() {
    return "created on " + dateCreated + "\ncolor: " + color +
        " and filled: " + filled;
  }

  /** Abstract method getArea */
  public abstract double getArea();

  /** Abstract method getPerimeter */
  public abstract double getPerimeter();
}
```

Abstract Classes

```java
package chapter10;

public class TestGeometricObject {
  /** Main method */
  public static void main(String[] args) {
    // Declare and initialize two geometric objects
    GeometricObject2 geoObject1 = new Circle(5);
    GeometricObject2 geoObject2 = new Rectangle(5, 3);

    System.out.println("The two objects have the same area? " +
        equalArea(geoObject1, geoObject2));

    // Display circle
    displayGeometricObject(geoObject1);

    // Display rectangle
    displayGeometricObject(geoObject2);
  }

```

Abstract Classes

```java
    /** A method for comparing the areas of two geometric objects */
    public static boolean equalArea(GeometricObject2 object1,
                                    GeometricObject2 object2) {
      return object1.getArea() == object2.getArea();
    }

    /** A method for displaying a geometric object */
    public static void displayGeometricObject(GeometricObject2 object) {
      System.out.println();
      System.out.println("The area is " + object.getArea());
      System.out.println("The perimeter is " + object.getPerimeter());
    }
}
```

Abstract Classes

- 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 Classes

- A class that contains abstract methods must be abstract.
- However, it is possible to declare an abstract class that contains no abstract methods.
- In this case, you cannot create instances of the class using the new operator.
- This class is used as a base class for defining a new subclass.

References

References

- Y. Daniel Liang, **Introduction to Java Programming**, Sixth Edition, Pearson Education, 2007. (Chapter 10)

The End