Java Programming AP Edition
U4C13 Abstract Classes and Interfaces

ABSTRACT CLASSES

ERIC Y. CHOU, PH.D. IEEE SENIOR MEMBER

Abstract Classes and Interfaces Templates and Shell of Data

Yes data fields No instantiation.

Abstract Class is CORE

Interface is Outer Skin

No data fields No instantiation.

Abstract Class

Pointer Yes, no objects, with data/methods (Conceptual Class)

A class can be declared with the abstract qualifier, e.g.:

public abstract class SuperClass

When this is added, it means that **no object** can be instantiated from this class and so must have subclasses for it to be useful.

Experiment with abstract

Using the classes from the basic package above:

1. Edit SuperClass, changing the declaration line to the following and save the changes:public abstract class SuperClass { When you save the changes, you'll see that the Driver class now has an error.
2. Edit Driver and observe NetBeans' response. Fix it by commenting out the flagged line://new SuperClass(), Re-run the driver program observe the only change is reflected by the missing object.
3. Reset SuperClass and Driver back to their original states.

Note that the top member function in SuperClass is unaffected by making the class abstract. An abstract class is similar to an interface in that it must be extended to be used, but unlike an interface in that it usually **does** have functionality whereas an interface has no functionality, only prototypes.

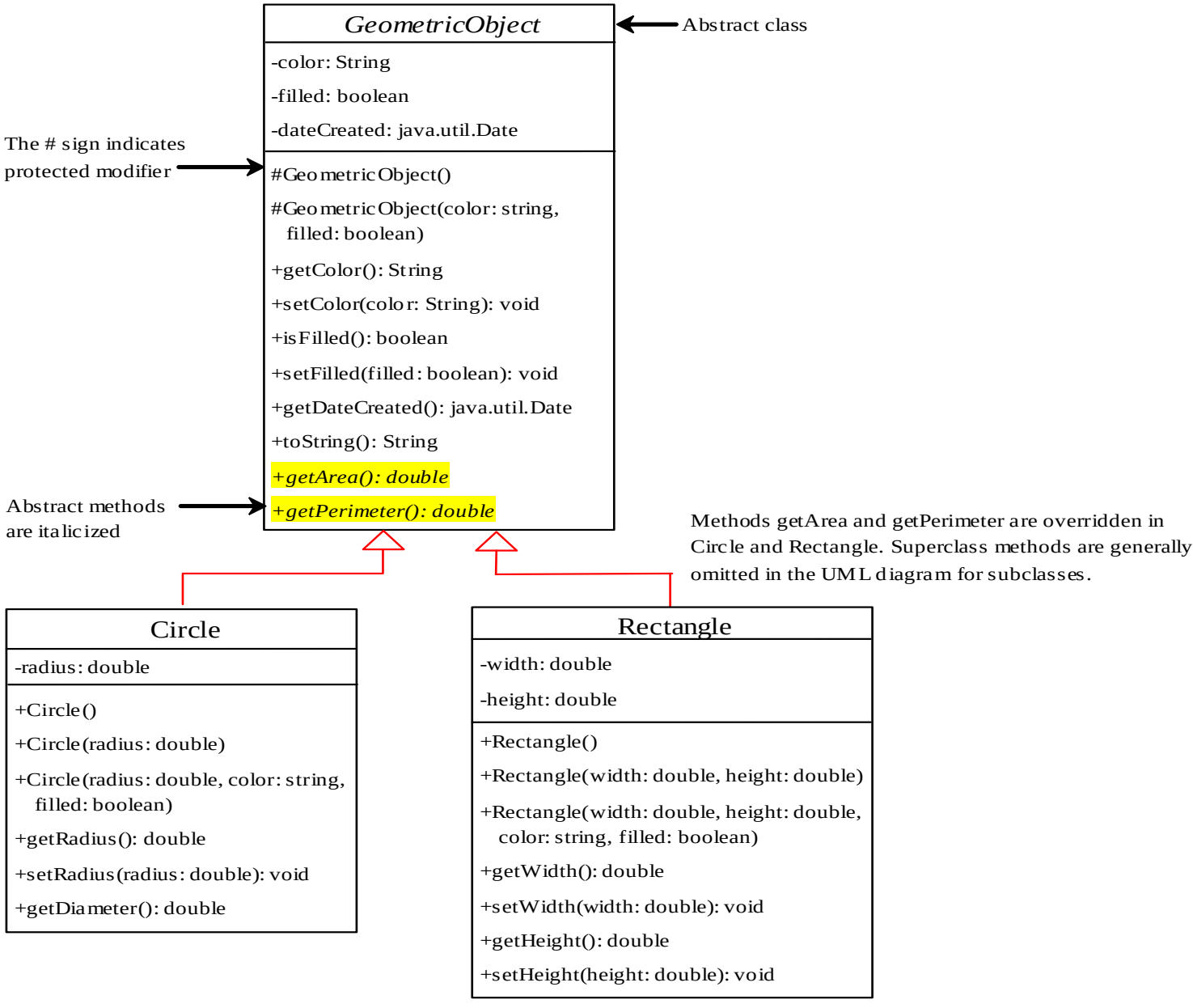
Motivations

You learned how to write simple programs to display GUI components. Can you write the code to respond to user actions such as clicking a button?

Demo Program: HandleEvent.java

Abstract Classes and Abstract Methods

Demo Program:
GeometricObject.java
Circle.java
Rectangle.java
TestGeometricObject.java



Abstract method in abstract class

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 defined 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 Methods undefined method (like headers of methods)

Object cannot be created from 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 without abstract method

A class that contains abstract methods must be abstract. However, it is possible to define 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.

(Template Class of Classes)

Superclass of Abstract Class May Be Concrete

A subclass can be abstract even if its superclass is concrete. For example, the Object class is concrete, but its subclasses, such as GeometricObject, may be abstract.

Concrete method overridden to be abstract

A subclass can override a method from its superclass to define it abstract. This is rare, but useful when the implementation of the method in the superclass becomes invalid in the subclass. In this case, the subclass must be defined abstract.

(Disable concrete methods to set new definition in subclasses.)

Abstract Class as Type

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];

Final classes

Declaring a class final means that it cannot be extended; it is effectively the exact opposite of abstract. **Final classes** are the "**true leaves**" in an inheritance tree diagram. As a simple experiment with our basic example, add final onto the declaration line of SubClass1:

```
public final class SubClass1 {
```

Save the changes and observe that the class declaration in the file SubSubClass1 is now flagged with the obvious error message. Again, undo the change in SubClass1 to fix it.