Java Programming AP Edition
U4C11 Inheritance and Polymorphism

INHERITANCE (SUPER CLASS AND SUBCLASS)

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

Inheritance

Inheritance in Java begins with the relationship between two classes defined like this:

class SubClass extends SuperClass

Inheritance expresses the is a relationship in that SubClass is a (**specialization** of) SuperClass. The extends relation has many of the same characteristics of the implements relationship used for interfaces

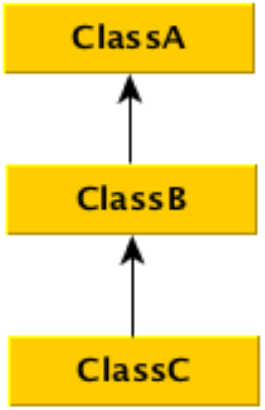
class MyImplementationClass implements MyInterface

As with inheritance, we say that **MyImplementationClass** is a **MyInterface**.

Inheritance

Diagrammatically, these relationships are expressed in UML with the extends as a solid line (white Triangle in some tools) and implements as a dashed line:

The ***is a*** relationship is transitive in that, if we have this hierarchy:



Inheritance

in which
ClassB is a ClassA
ClassC is a ClassB
then, by transitivity:
ClassC is a ClassA
The term base class is also used for superclass, and derived class as subclass.
Being a subclass is also transitive in that we can say that:
ClassC is a subclass of ClassA
The term inheritance expresses the fact that the objects of the subclass inherit all the features of the superclass including data members and functions, although the private data members and functions of the superclass are **not** directly accessible.

What does a subclass inherit?

A **subclass inherits** all the members (fields, methods, and nested classes) from its superclass. **Constructors** are not members, so they are not **inherited** by **subclasses**, but the constructor of the superclass can be invoked from the **subclass**.

Members of a class that are declared private are not directly accessible by subclasses of that class. Only members of a class that are declared **protected** or **public** are accessed directly by subclasses declared in a package other than the one in which the class is declared.

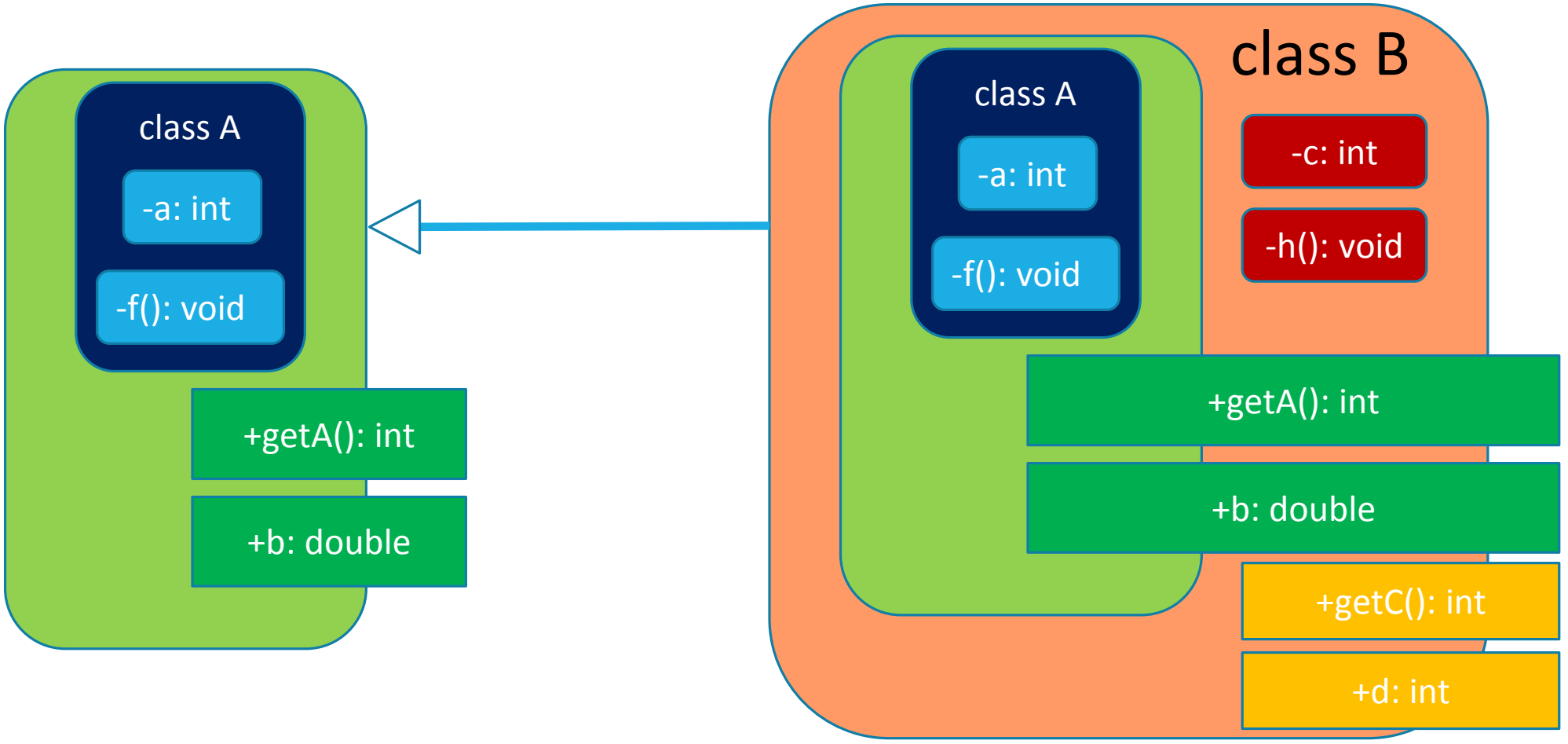
protected: no access by other package but can be inherited.

Visibility Modifiers

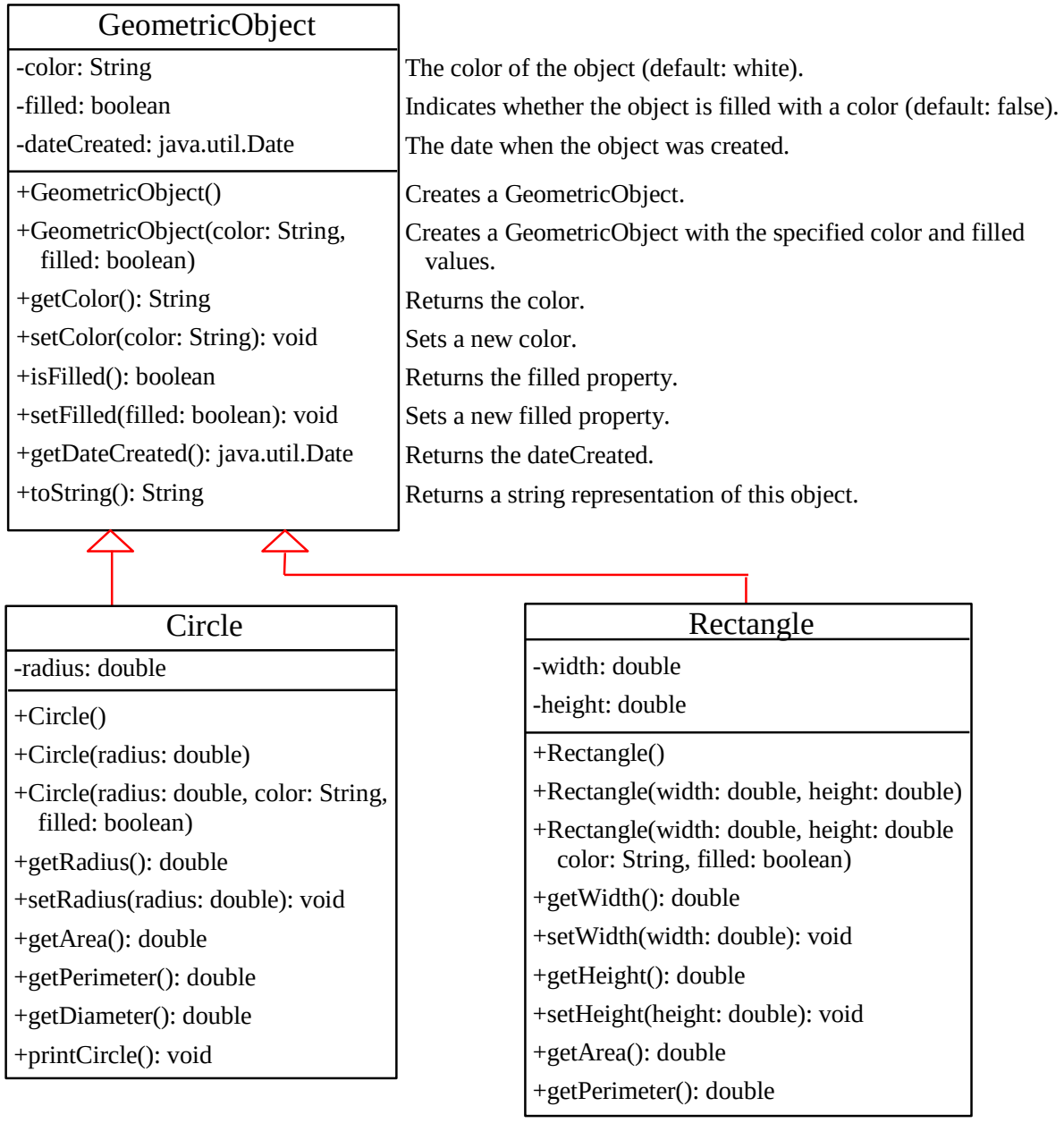
Modifier	Inheritance	Access
+public	All	All
#protected	All	Subclasses
~default (none)	All	Same package
-private	All	Same class

Accessing an Objects of subclasses

(Subclass is another kind of Wrapper Class)



Superclasses and Subclasses



Demo Programs:

(TestCircleRectangle.java GeometricObject.java CircleFromSimpleGeometricObject.java RectangleFromSimpleGeometricObject.java)

Go BlueJ!!!

Are superclass's Constructor Inherited?

- **No. They are not inherited.**
- **They are invoked explicitly or implicitly.**
- **Explicitly using the super keyword.**

A constructor is used to construct an instance of a class. Unlike properties and methods, a superclass's constructors are not inherited in the subclass. They can only be invoked from the subclasses' constructors, using the keyword super. ***If the keyword super is not explicitly used, the superclass's no-arg constructor is automatically invoked.***

Superclass's Constructor Is Always Invoked

A constructor may invoke an overloaded constructor or its superclass's constructor. If none of them is invoked explicitly, the compiler puts super() as the first statement in the constructor. For example,

```
public A() {
}
```

is equivalent to

```
public A() {
  super();
}
```

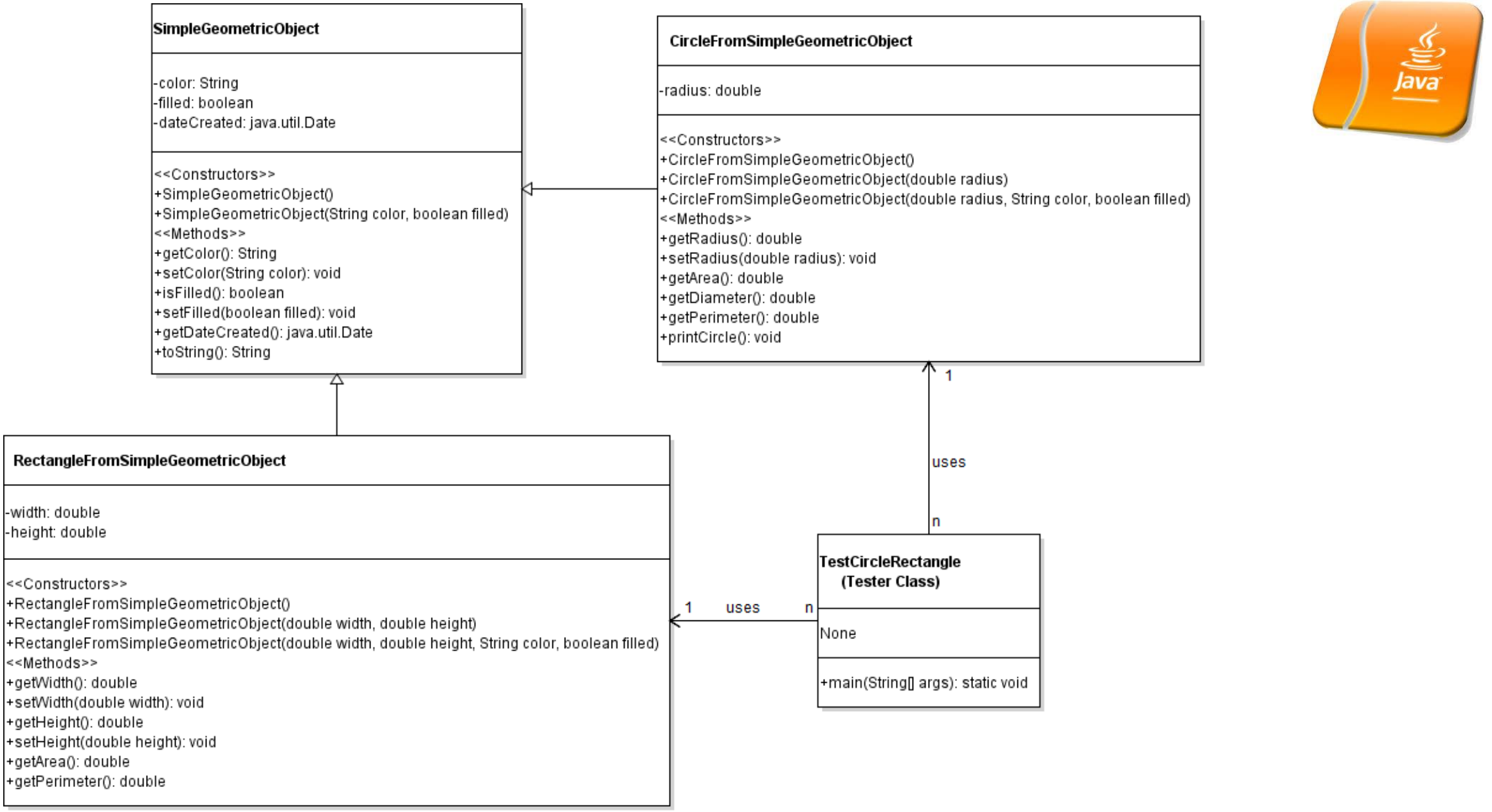
```
public A(double d) {
  // some statements
}
```

is equivalent to

```
public A(double d) {
  super();
  // some statements
}
```

Violet UML for Inheritance

Go Violet UML Editor!!!


SimpleGeometricObject
-color: String
-filled: boolean
-dateCreated: java.util.Date
<<Constructors>>
+SimpleGeometricObject()
+SimpleGeometricObject(String color, boolean filled)
<<Methods>>
+getColor(): String
+setColor(String color): void
+isFilled(): boolean
+setFilled(boolean filled): void
+getDateCreated(): java.util.Date
+toString(): String
CircleFromSimpleGeometricObject
-radius: double
<<Constructors>>
+CircleFromSimpleGeometricObject()
+CircleFromSimpleGeometricObject(double radius)
+CircleFromSimpleGeometricObject(double radius, String color, boolean filled)
<<Methods>>
+getRadius(): double
+setRadius(double radius): void
+getArea(): double
+getDiameter(): double
+getPerimeter(): double
+printCircle(): void
1
uses
n
RectangleFromSimpleGeometricObject
-width: double
-height: double
<<Constructors>>
+RectangleFromSimpleGeometricObject()
+RectangleFromSimpleGeometricObject(double width, double height)
+RectangleFromSimpleGeometricObject(double width, double height, String color, boolean filled)
<<Methods>>
+getWidth(): double
+setWidth(double width): void
+getHeight(): double
+setHeight(double height): void
+getArea(): double
+getPerimeter(): double
1
uses
n
TestCircleRectangle
(Tester Class)
None
+main(String[] args): static void