CS112

Abstract Classes
Chapter 13

Lecture 09

الفصل الدراسي الثاني 1442- Spring 2021

College of Computer Science and Engineering

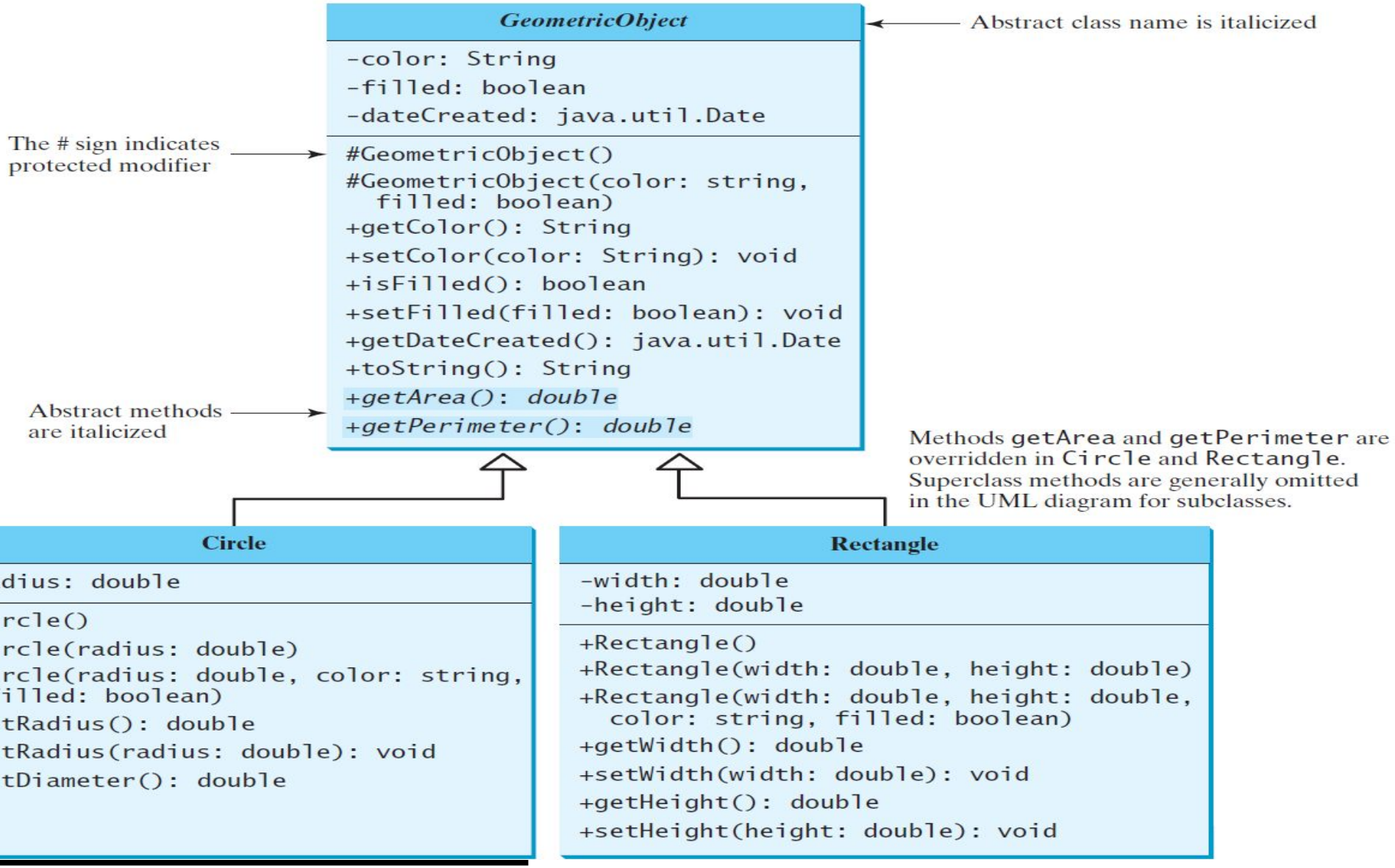
Introduction

- You can use the java.util.Arrays.sort method to sort an array of numbers or strings. Can you apply the same sort method to sort an array of geometric objects?
  - In order to write such code, you have to know about interfaces
- An interface is for defining common behavior for classes (including unrelated classes). Before discussing interfaces, we introduce a closely related subject: 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 be used to create any specific instances . Such a class is referred to as an **abstract class**

Example 1: GeometricObject Class



Abstract Classes and Abstract Methods (1)

- 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 classes are like regular classes, but you cannot create instances of abstract classes using the new operator
- An abstract method is defined without implementation. Its implementation is provided by the subclasses. A class that contains abstract methods must be defined as abstract

Abstract Classes and Abstract Methods (2)

- Therefore, the following statement, which creates an array whose elements are of GeometricObject type, is correct.

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

Constructor of an Abstract Class

- The constructor in the abstract class is defined as protected, because it is used only by subclasses
- When you create an instance of a concrete subclass, its superclass's constructor is invoked to initialize data fields defined in the superclass

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.

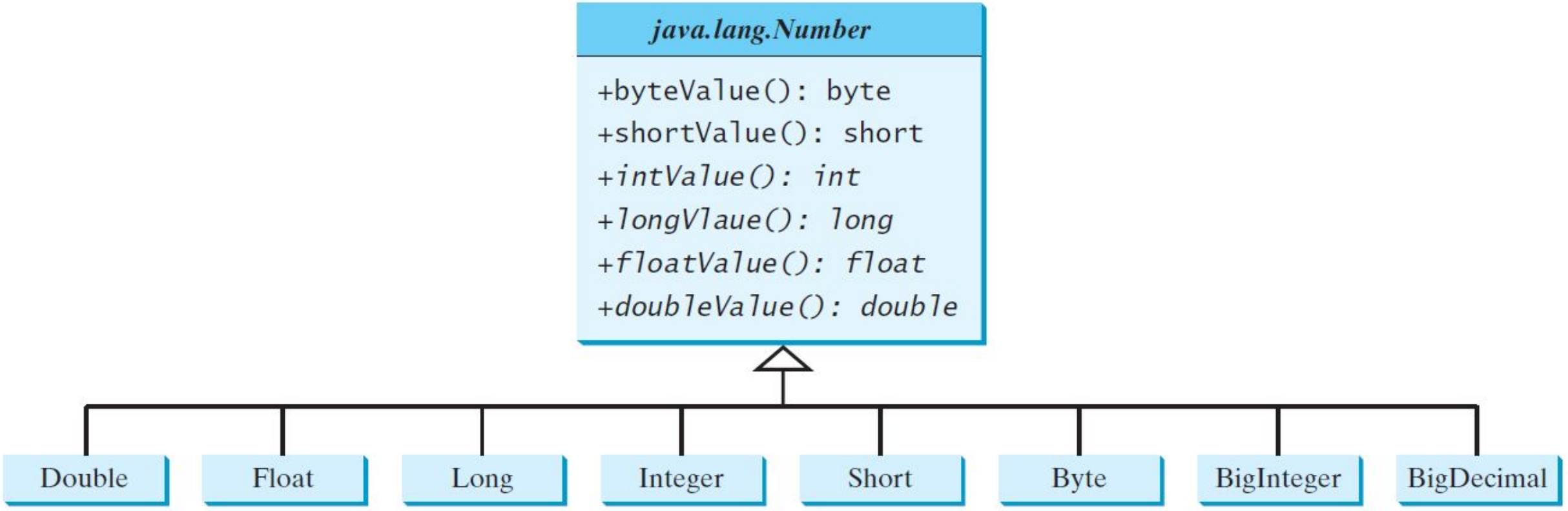
Abstract Class without Abstract Method

- 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
- 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

Example 2: Number Class (1)

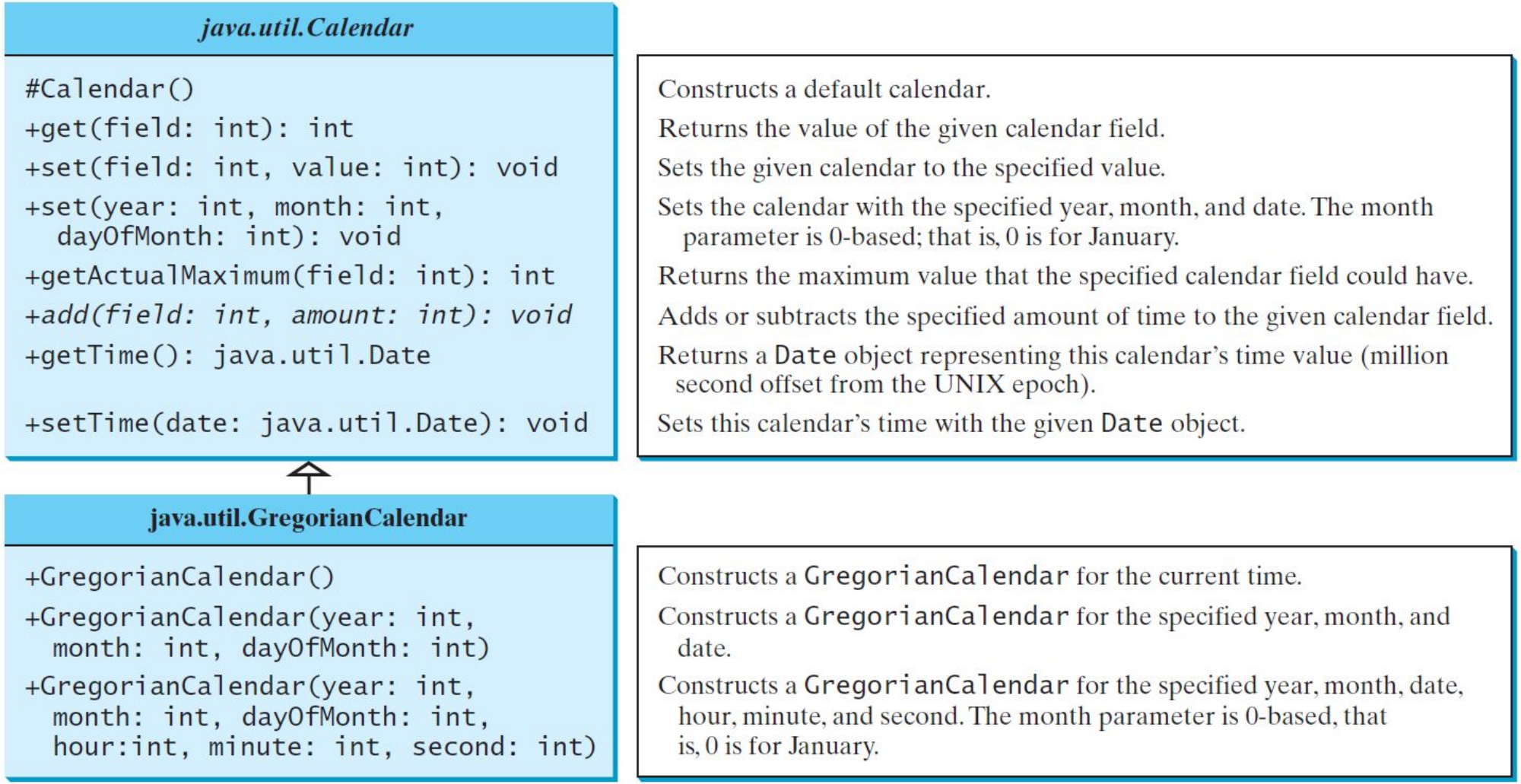


Example 2: Number Class (2)

- Since the intValue(), longValue(), floatValue(), and doubleValue() methods cannot be implemented in the Number class, they are defined as abstract methods in the Number class
- The Number class is therefore an abstract class
- The byteValue() and shortValue() method are implemented from the intValue() method as follows:

```
public byte byteValue() {
    return (byte)intValue();
}
public short shortValue() {
    return (short)intValue();
}
```

Example 3: The Abstract Calendar Class and Its GregorianCalendar Subclass (1)



Example 3: The Abstract Calendar Class and Its GregorianCalendar Subclass (2)

- An instance of java.util.Date represents a specific instant in time with millisecond precision
- java.util.Calendar is an abstract base class for extracting detailed information such as year, month, date, hour, minute and second from a Date object
- Subclasses of Calendar can implement specific calendar systems such as Gregorian calendar, Lunar Calendar and Jewish calendar
- Currently, java.util.GregorianCalendar for the Gregorian calendar is supported in the Java API

Example 3: The Abstract Calendar Class and Its GregorianCalendar Subclass (3)

- The get Method in Calendar Class:
  - The get(int field) method defined in the Calendar class is useful to extract the date and time information from a Calendar object. The fields are defined as constants, as shown in the following:
  - See TestCalendar.java

Constant	*Description*
YEAR	The year of the calendar.
MONTH	The month of the calendar, with 0 for January.
DATE	The day of the calendar.
HOUR	The hour of the calendar (12-hour notation).
HOUR_OF_DAY	The hour of the calendar (24-hour notation).
MINUTE	The minute of the calendar.
SECOND	The second of the calendar.
DAY_OF_WEEK	The day number within the week, with 1 for Sunday.
DAY_OF_MONTH	Same as DATE.
DAY_OF_YEAR	The day number in the year, with 1 for the first day of the year.
WEEK_OF_MONTH	The week number within the month, with 1 for the first week.
WEEK_OF_YEAR	The week number within the year, with 1 for the first week.
AM_PM	Indicator for AM or PM (0 for AM and 1 for PM).