Wrapper Classes for Primitive Types in Java

Primitive Data Types

Include...

byte, short, int, long, float, double

char

boolean

Q. Why aren’t these objects?

A. Efficiency (avoid “object overhead”)

Wrapper Classes

...but sometimes it would be useful to have objects hold primitive data.

Example

To include different primitive data types in a single `Object[]` array.

Wrapper Classes

– Classes for “wrapping” primitive data in objects.
– All override the Object methods toString, equals, and hashCode.
– All wrapper classes (except for Boolean) implement the Comparable interface (implement compareTo())

UML Class Diagram for Wrapper Classes

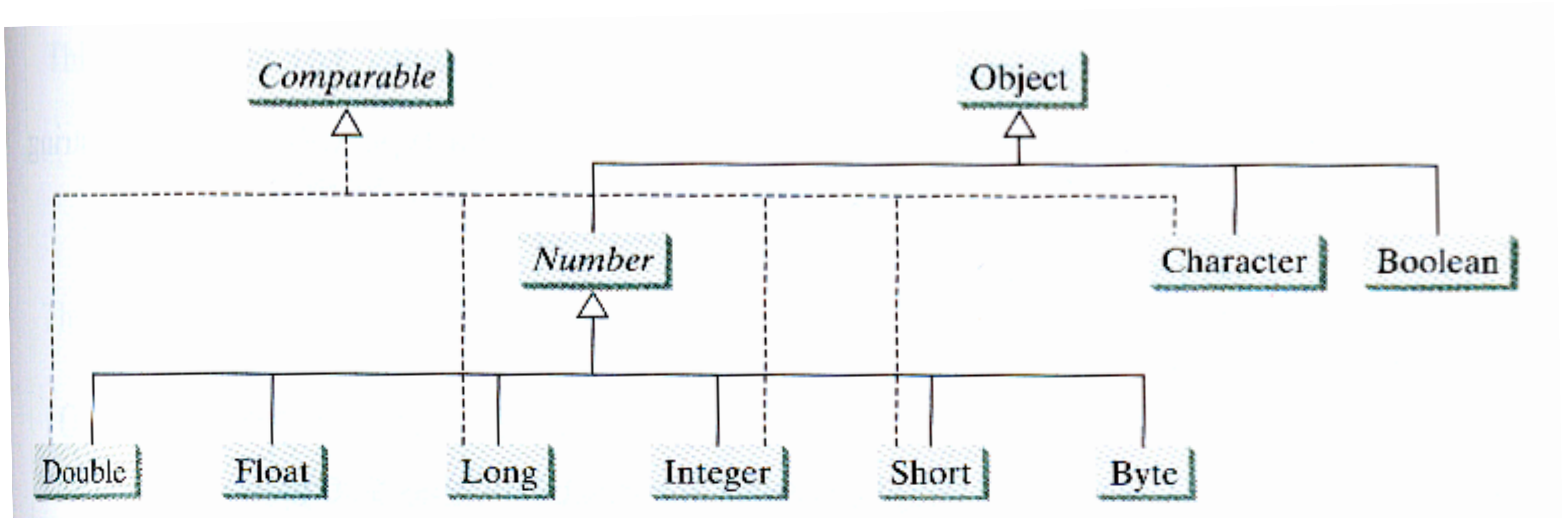
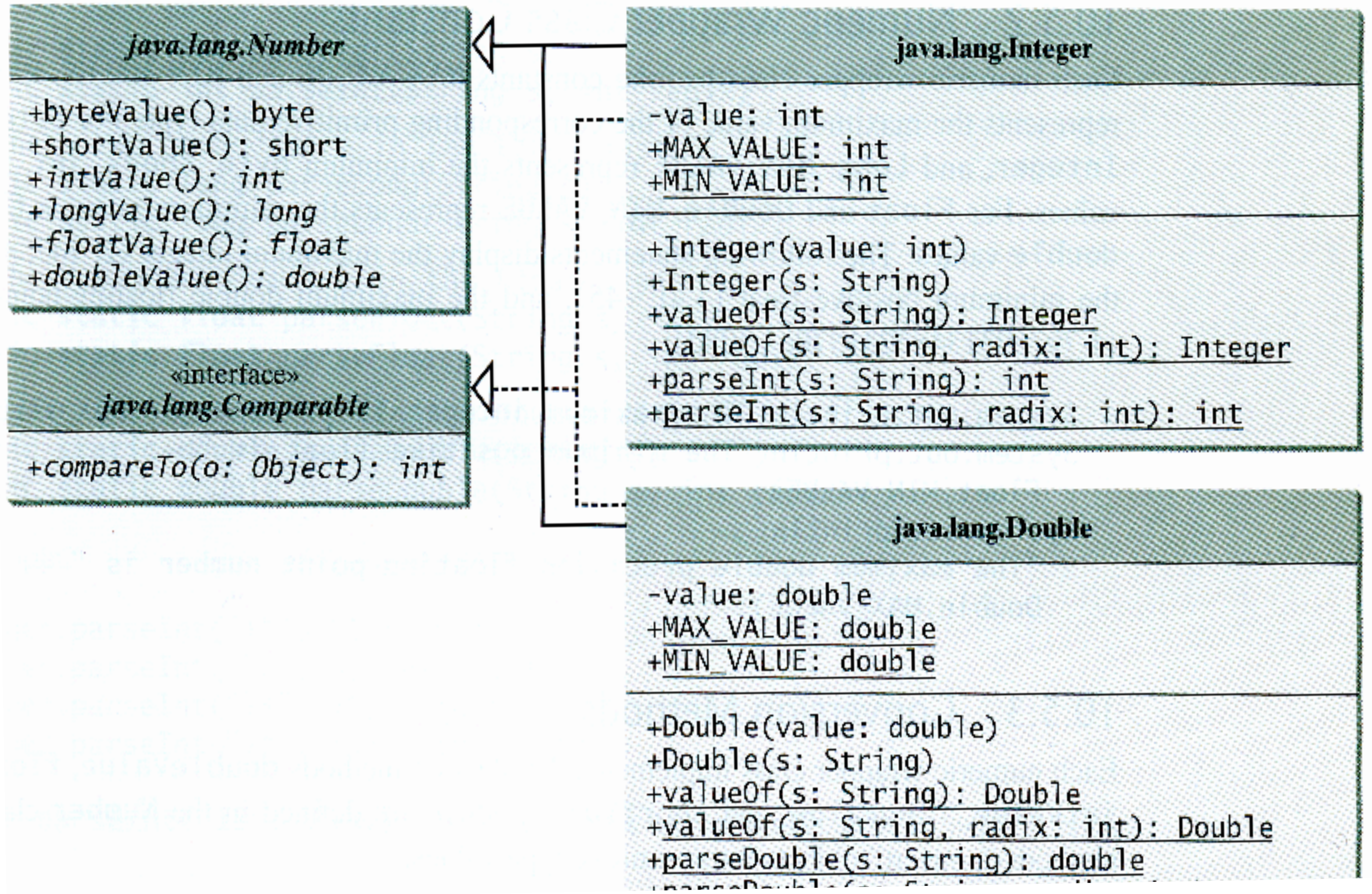


FIGURE 10.7 The Number class is an abstract superclass for Double, Float, Long, Integer, Short, and Byte.

NOTE: all wrapper classes capitalize the name of the associated primitive type, except for Integer and Character.


java.lang.Number
+byteValue(): byte
+shortValue(): short
+intValue(): int
+longValue(): long
+floatValue(): float
+doubleValue(): double
«interface»
java.lang.Comparable
+compareTo(o: Object): int
java.lang.Integer
-value: int
+MAX_VALUE: int
+MIN_VALUE: int
+Integer(value: int)
+Integer(s: String)
+valueOf(s: String): Integer
+valueOf(s: String, radix: int): Integer
+parseInt(s: String): int
+parseInt(s: String, radix: int): int
java.lang.Double
-value: double
+MAX_VALUE: double
+MIN_VALUE: double
+Double(value: double)
+Double(s: String)
+valueOf(s: String): Double
+valueOf(s: String, radix: int): Double
+parseDouble(s: String): double

Example: Constructing Wrapped Numbers

```
Double doubleObject = new Double(5.0);
Double doubleObject = new Double(“5.0”);
Double doubleObject = Double.valueOf(“12.4”)

Integer intObject = new Integer(5);
Integer intObject = new Integer(“5”);
Integer intObject = Integer.valueOf(“12”);
```

NOTE: *valueOf* is a static method defined for all numeric wrapper classes.

Converting Between Strings and Primitive Numeric Types

Converting to String

```
Double doubleObject = new Double(5.0);
String s = doubleObject.toString();
```

Converting from String

```
double d = Double.parseDouble(“5.0”);
int i = Integer.parseInt(“5”);
// Using ‘parse’ method with a radix (base):
int j = Integer.parseInt(“11”, 2);  // j=3 (in base 10!)
```

Example: A Polymorphic (“Generic”) Sorting Method

Text Example, GenericSort.java

(implementation of Selection Sort: iteratively finds largest element, places it at the end of the array)

- Using the `Comparable` interface (`compareTo()`), different object types are sorted using the same sorting method; each class defines how objects of the class should be ordered.

- NOTE: Java defines a static sort in the Arrays class, for any array of objects implementing *Comparable*
  - e.g. `Arrays.sort(intArray);`

Automatic Conversion Between Primitive and Wrapper Class Types

‘Boxing’

Converting primitive → wrapper

e.g. Integer[ ] intArray = {1, 2, 3};

e.g. Integer intObject = 2;   // both legal, ‘autoboxing’ occurs

‘Unboxing’

Converting wrapper → primitive

e.g. System.out.println(intArray[0] + intArray[1] + intArray[2]);

    // int values are summed before output.

e.g. int i = new Integer(3);   // legal, ‘autounboxing occurs’

Automatic Conversions

– Compiler will box for contexts requiring an object
– Compiler will unbox for contexts requiring a primitive