9 THE WRAPPER CLASSES

Terry Marris 18 April 2001

9.1 OBJECTIVES

By the end of this lesson the student should be able to

- distinguish between *int* and *Integer*, *double* and *Double*, *boolean* and *Boolean*, *char* and *Character*
- browse the standard Java library documentation
- use wrapper class methods

9.2 PRE-REQUISITES

The student should be comfortable with using method signatures to create test routines, and with using a web browser e.g. Netscape.

9.3 PREVIEW

Each of the primitive data types, *int*, *double*, *char* and *boolean* for example, has their own wrapper class. Perhaps the most useful feature of wrapper classes is that they provide methods for conversion between the primitive data types and strings.

We look at how to view the standard Java class documentation.

9.4 THE INTEGER CLASS

The *Integer* class wraps a value of the primitive type *int* in an object. An object of type *Integer* contains a single field whose type is *int*.

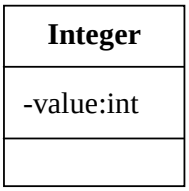


Figure 9.1 *An Integer object wraps an int value*

The class provides methods for converting an *int* to a string, and a string to an *int*, as well as other constants and useful methods.

9.4.1 STATIC FIELDS AND METHODS

The *Integer* class contains two static fields.

Fields		
static int	MAX_VALUE	the largest possible *int* value, 2 147 483 647
static int	MIN_VALUE	the least possible *int* value, -2 147 483 648

Static fields belong to the class and not to objects of the class. So to use them we attach them to the class name using the dot selector operator like this:

```
int largestInt = Integer.MAX_VALUE;
```

The *Integer* class contains several static methods. One of them converts a string to an *int*.

static int parseInt(String s) converts the given string to an *int*.

To use static methods we attach them to the class name using the dot selector operator like this:

```
int i = Integer.parseInt("5");
```

9.4.2 CONSTRUCTORS

The *Integer* class has two constructors.

Constructors	
Integer(int value)	an *Integer* object that represents the given *int* value.
Integer(String s)	an *Integer* object that represents the *int* value represented by the given string, *s*. s must represent a valid *int* (otherwise a *NumberFormatException* error is generated).

One constructor has an *int* as its parameter, the other has a string. We might use them to initialise two *Integer* objects with value five like this:

```
Integer integer1 = new Integer(5);
Integer integer2 = new Integer("5");
```

It would be an error to supply a string argument that could not possibly be converted to an *int*. For example

```
Integer anInteger = new Integer("XX");
```

would generate a run-time error.

9.4.3 METHODS

Some useful methods are shown in the following table.

Methods		
int	compareTo(Integer anInteger)	returns a value less than zero if this Integer's value is less than the given anInteger's value, zero if the two Integer values are the same, returns a value more than zero if this Integer's value is more than the given anInteger's value.
double	doubleValue()	returns the value of this *Integer* as a *double*.
boolean	equals(Object obj)	returns *true* if the given object is an *Integer* object with the same *int* value as this Integer's *int* value.
int	intValue()	returns the value of this *Integer* as an *int*.
static int	parseInt(String s)	converts the given string to an *int*. *s* must represent a valid *int* otherwise a *NumberFormatException* error is generated.
String	toString()	returns a string object representing this Integer's value.
static String	toString(int i)	returns a new string object representing the given *int*.
static Integer	valueOf(String s)	returns a new *Integer* object initialised to the *int* value represented by *s*. *s* must represent a valid *int* otherwise a *NumberFormatException* error is generated.

9.4.4 TEST PROGRAM

A simple test program that uses some of the *Integer* class fields and methods is shown below.

```
/* TestInteger.java
   Terry Marris  19 April 2001
*/

public class TestInteger {
  public static void main(String[] s)
  {
    System.out.println("The largest int value is ... " +
                        Integer.MAX_VALUE);

    System.out.println("Using the two constructors to " +
              "create two Integer objects with value 5 ... ");
    Integer integer1 = new Integer(5);
    Integer integer2 = new Integer("5");

    System.out.println("Both Integer objects are " +
                 "the same ... " + integer1.equals(integer2));

    System.out.println("Converting Strings to ints.");
    int i = Integer.parseInt("5") + Integer.parseInt("2");
    System.out.println("The result should be 7. " +
                       "It is ... " + i);

    System.out.println("Performing arithmetic with " +
                       "Integer objects.");
    i = integer1.intValue() + integer2.intValue();
    System.out.println("The result should be 10. " +
                       "It is .. " + i);
  }
}
```

Program run:

```
The largest int value is ... 2147483647
Using the two constructors to create two Integer objects with
value 5 ...
Both Integer objects are the same ... true
Converting Strings to ints.
The result should be 7.  It is ... 7
Performing arithmetic with Integer objects.
The result should be 10.  It is .. 10
```

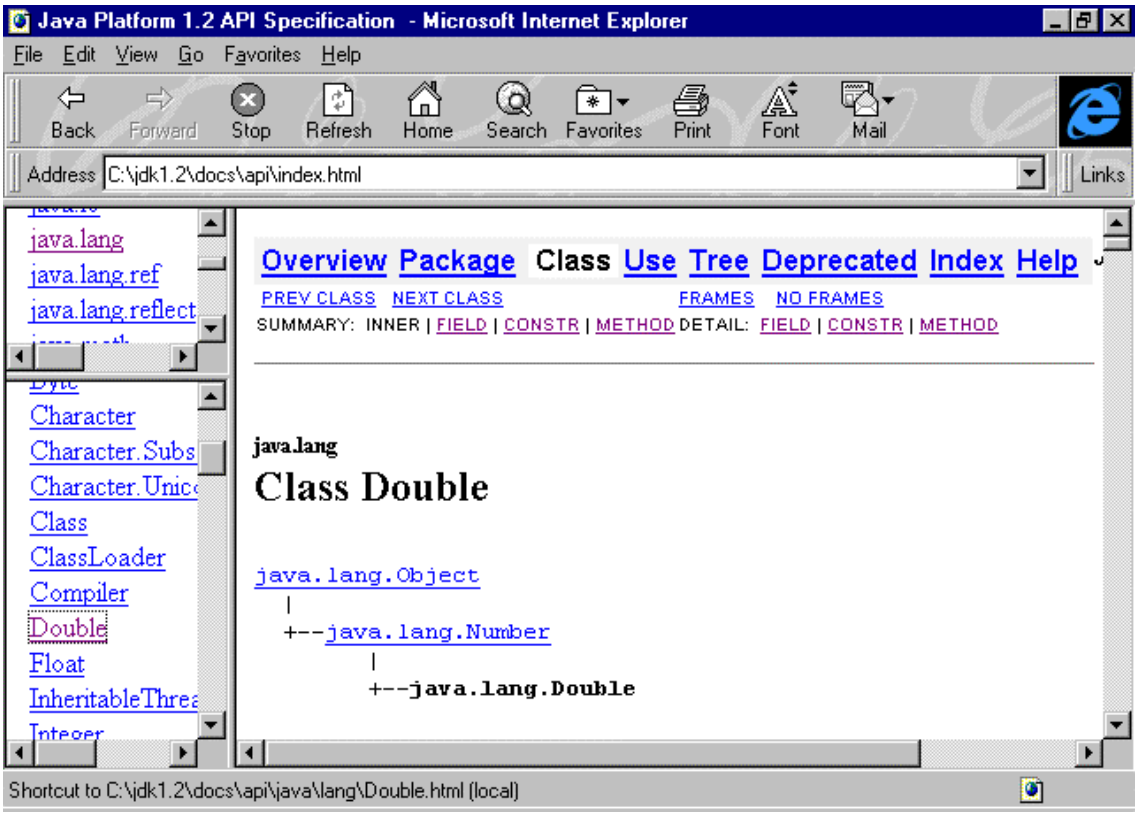
Exercise: identify which line (or lines) of coding produced each line of output.

9.5 THE JAVA DOCUMENTATION

The standard Java documentation is in Hyper Text Markup Language (HTML) format and you should become accustomed to referring to it.

Remember that it was written for professional programmers - so do not be put off if there are parts of it you do not understand. You will understand most of it eventually.

Point your web browser at *c:\jdk1.2\docs\api\index.html*. You will see three panes. First choose the package e.g. *java.lang* from the top left hand pane. Then choose the class e.g. *Double* from the bottom left hand pane. The documentation describing the class and its methods is in the large right hand pane. You will find it useful to print up the class documentation for ease of reference.



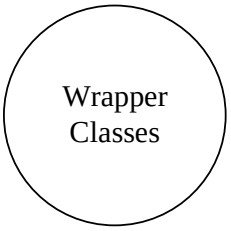
9.6 DOUBLE, BOOLEAN AND CHARACTER

The *Double*, *Boolean* and *Character* wrapper classes have similar functionality to the *Integer* wrapper class. The student should look the documentation for these classes and, for each class, list the method signatures (*returnType methodName(parameterTypes)*) and descriptions that they think are useful here.

9.7 FURTHER READING

HORSTMANN & CORNELL Core Java 2 Volume 1 pp 144, 184, 195

9.8 REVIEW



wrap a primitive (non-object) type

primitive types: int, double, char, boolean

classes

Integer, Double, Character, Boolean

have useful constants e.g. MAX_VALUE

have useful methods

from string to int:

```
int i = Integer.parseInt("7");
```

from int to string

```
String s = Integer.toString(7);
```

documented in *c:\jdk1.2\docs\api\java\lang*

9.9 EXERCISES

1 Look up and print the documentation for the standard Java *Integer*, *Double*, *Character* and *Boolean* wrapper classes.

2 Identify the methods that could be used to perform the following conversions in each direction:

(a) *int* ↔ *String*
(b) *double* ↔ *String*
(c) *char* ↔ *String*
(d) *boolean* ↔ *String*