1. Wrapper Classes and Autoboxing

The data types **int**, **double**, **char**, **boolean**, ... are referred to as **primitive data types** to distinguish them from class data types, i.e., classes.

In some situations it would be convenient to have a class that represents, for example, an integer. For these situations, the Java Class Library provides **wrapper classes** which encapsulate values of the primitive data types. These are some of the wrapper classes:

```
java.lang.Boolean
java.lang.Byte
java.lang.Char
java.lang.Double
java.lang.Integer
```

1. Wrapper Classes and Autoboxing :: *Integer* Wrapper Class

Let's look at *java.lang.Integer* which encapsulates an **int** and provides these methods:

***Integer*(int *value*)**

Constructs a new *Integer* object which encapsulates the **int** value.

```
// I encapsulates int 10.
Integer I = new Integer(10);
```

***Integer*(*String strValue*)**

strValue is a string containing an integer, e.g., something like "123" or "-4567" which is converted to an **int** value and is encapsulated in the object.

```
// J encapsulates int -4567
Integer J = new Integer("-4567");
```

int *compareTo*(*Integer anotherInteger*)

Returns 0 if this *Integer* is equal to *anotherInteger*, -1 if this *Integer* is less than *anotherInteger*, or 1 if this *Integer* is greater than *anotherInteger*.

```
// K encapsulates int 200. L encapsulates int 10.
Integer K = new Integer(200);
Integer L = new Integer(10);
int b1 = I.compareTo(J);
int b2 = I.compareTo(K);
int b3 = J.compareTo(K);
int b4 = L.compareTo(I);
```

1. Wrapper Classes and Autoboxing :: *Integer* Wrapper Class

int *intValue*()

Returns the **int** encapsulated within this *Integer*.

```
// y is assigned 10.
int y = I.intValue();
```

static int *parseInt*(*String strValue*)

strValue is a string containing an integer, e.g., something like "123" or "-4567" which is converted to an **int** value and returned.

```
// x is assigned -4567.
String s = "-4567";
int x = Integer.parseInt(s);
```

***String toString*()**

Returns a string representation of this *Integer*.

```
// Prints "10".
System.out.println(I.toString());
```

```
// Prints "10". toString() does not need to be called on objects when an object is
// passed as an argument to System.out.print() or System.out.println() because it
// gets called automatically.
System.out.println(I);
```

1. Wrapper Classes and Autoboxing :: Autoboxing

It can become tedious using wrapper classes and the primitive data types together:

```
int x = a + b;
Integer intX = new Integer(x);
...
int y = intX.intValue() + b;
intX = new Integer(x);
```

To make working with the wrapper classes more convenient, Java provides a feature called **autoboxing**. Examples,

```
int x = a + b;
Integer intX = x;
...
x = intX;
int y = x + b;
intX = x;
```