*Int V/s Integer*

***int vs Integer***

*Newbies are often confused by the difference between****int****and Integer,****boolean****vs Boolean,****char****vsCharacter,****short****vs Short,****long****vs Long,****float****vs Float or****double****vs Double.*

*To properly understand the difference, you should read an introductory textbook on Java. I will make a small stab at answering here.*

*Everything I say here applies analogously to****char****and Character,****short****and Short,****long****and Long,****float****and Float,****double****and Double.*

***Definitions***

*An****int****is a*[*primitive*](http://mindprod.com/jgloss/primitive.html)*. It is****not****an Object. An****int****is a high performance, streamlined beast for calculating numbers in the range -2,147,483,648 [-231] aka Integer.MIN\_VALUE to +2,147,483,647 [2 31-1] aka Integer.MAX\_VALUE. An****int****is a bare bones 32-bit chunk of information.****int****variables are*[*mutable*](http://mindprod.com/jgloss/mutable.html)*. Unless you mark them****final****, you can change their value at any time.*

*An Integer, is a Object that contains a single****int****field. An Integer is much bulkier than an****int****. It is sort like a Fedex box to contain the****int****. Integers are*[*immutable*](http://mindprod.com/jgloss/immutable.html)*. If you want to affect the value of aInteger variable, the only way is to create a****new****Integer object and discard the old one.*

***Which is better?***

*That is like asking which is better a volley ball or a volleyball in a Fedex box. It depends what you want to do with it.*

|  |  |  |
| --- | --- | --- |
| ***int vs Integer*** |  |  |
| ***ability*** | ***unboxed int*** | ***boxed Integer*** |
| *Calculate with it,****+ - \* / % ^****etc.* |  |  |
| *Pass it as a parameter* |  |  |
| *Return it as a value* |  |  |
| *Use methods on it from java.lang.Integer* |  |  |
| *Store it in a Vector or other Collection* |  |  |
| *Use it as a HashMap key* |  |  |
| *serialize it* |  |  |
| *send it by itself over*[***RMI***](http://mindprod.com/jgloss/rmi.html)*(****R****emote****M****ethod****I****nvocation)* |  |  |
| *send it as part of another Object over RMI* |  |  |
| *Pass it as a generic object to a TableCellRenderer. You may pass a variety of different types via the same paramter.* |  |  |
| *Allow for a****null****value that means there is no value.* |  |  |

***Converting***

*Fortunately it is easy to convert back and forth between****int****and Integer.*

*// to int i from Integer ii*

***int i =*** *ii****.****intValue****();***

*// to Integer ii from int i*

*Integer* ***ii = new*** *Integer****(*** *i* ***);***

*See this*[*Amanuensis*](http://mindprod.com/applet/converter.html)*for other conversions.*

*Starting with Java 1.5, with****boxing****and****unboxing****, the compiler will sometimes automatically do the conversions to and from****int****and Integer for you so you can treat****int****and Integer as if they were almost the same thing.*

***Why Both?***

*Why are there both****int****and Integer? For speed.****int****s, without any Object packaging are compact and fast. Would it not have been easier if there were only one of sort of creature that could do everything and have the compiler automatically figure out when the packaging was needed and when not? The Eiffel language designers thought so and the Java designers are gradually coming around to the same conclusion.*

**Introduction**

*Java is an object-oriented language and can view everything as an object. A simple file can be treated as an object (with****java.io.File****), an address of a system can be seen as an object (with****java.util.URL****), an image can be treated as an object (with****java.awt.Image****) and a simple data type can be converted into an object (with****wrapper classes****). This tutorial discusses wrapper classes.****Wrapper classes****are used to convert any data type into an object.*

*The primitive data types are not objects; they do not belong to any class; they are defined in the language itself. Sometimes, it is required to convert data types into objects in Java language. For example, upto JDK1.4, the data structures accept only objects to store. A data type is to be converted into an object and then added to a Stack or Vector etc. For this conversion, the designers introduced****wrapper classes****.*

***What are Wrapper classes?***

*As the name says, a wrapper class wraps (encloses) around a data type and gives it an object appearance. Wherever, the data type is required as an object, this object can be used. Wrapper classes include methods to unwrap the object and give back the data type. It can be compared with a chocolate. The manufacturer wraps the chocolate with some foil or paper to prevent from pollution. The user takes the chocolate, removes and throws the wrapper and eats it.*

*Observe the following conversion.*

*int k = 100;  
Integer it1 = new Integer(k);*

*The int data type k is converted into an object, it1 using Integer class. The it1 object can be used in Java programming wherever k is required an object.*

*The following code can be used to unwrap (getting back int from Integer object) the object it1.*

*int m = it1.intValue();  
System.out.println(m\*m); // prints 10000*

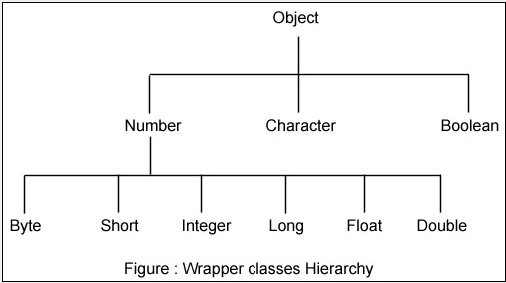
*intValue() is a method of Integer class that returns an int data type.*

*List of Wrapper classes*

*In the above code, Integer class is known as a wrapper class (because it wraps around int data type to give it an impression of object). To wrap (or to convert) each primitive data type, there comes a wrapper class. Eight wrapper classes exist in java.lang package that represent 8 data types. Following list gives.*

|  |  |
| --- | --- |
| PRIMITIVE DATA TYPE | WRAPPER CLASS |
| byte | Byte |
| short | Short |
| int | Integer |
| long | Long |
| float | Float |
| double | Double |
| char | Character |
| boolean | Boolean |

*Following is the hierarchy of the above classes.*

[](http://way2java.com/wp-content/uploads/2011/01/ss52.bmp)

*All the 8 wrapper classes are placed injava.lang package so that they are implicitly imported and made available to the programmer. As you can observe in the above hierarchy, the super class of all numeric wrapper classes is Number and the super class for Character and Boolean is Object. All the wrapper classes are defined as final and thus designers prevented them from inheritance.*

*Importance of Wrapper classes*

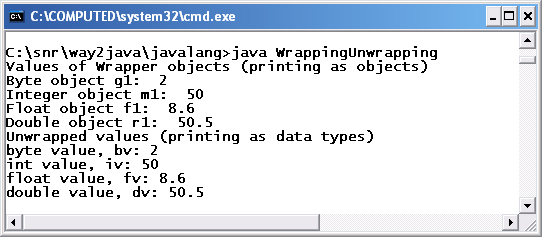
*There are mainly two uses with wrapper classes.*

*To convert simple data types into objects, that is, to give object form to a data type; here constructors are used.*

*To convert strings into data types (known as parsing operations), here methods of type parseXXX() are used.*

*The following program expresses the style of converting data type into an object and at the same time retrieving the data type from the object.*

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32 | public class WrappingUnwrapping  {    public static void main(String args[])    {                   //  data types      byte grade = 2;      int marks = 50;      float price = 8.6f;                 // observe a suffix of <strong>f</strong> for float      double rate = 50.5;                                    // data types to objects      Byte g1 = new Byte(grade);                    // wrapping      Integer m1 = new Integer(marks);      Float f1 = new Float(price);      Double r1 = new Double(rate);                                                                      // let us print the values from objects      System.out.println("Values of Wrapper objects (printing as objects)");      System.out.println("Byte object g1:  " + g1);      System.out.println("Integer object m1:  " + m1);      System.out.println("Float object f1:  " + f1);      System.out.println("Double object r1:  " + r1);          // objects to data types (retrieving data types from objects)      byte bv = g1.byteValue();                 // unwrapping      int iv = m1.intValue();      float fv = f1.floatValue();      double dv = r1.doubleValue();                                                                      // let us print the values from data types      System.out.println("Unwrapped values (printing as data types)");      System.out.println("byte value, bv: " + bv);      System.out.println("int value, iv: " + iv);      System.out.println("float value, fv: " + fv);      System.out.println("double value, dv: " + dv);    }  } |

[](http://way2java.com/wp-content/uploads/2011/01/ss16.bmp)  
*As you can observe from the screenshot, constructors of wrapper classes are used to convert data types into objects and the methods of the form XXXValue() are used to retrieve back the data type from the object.*