Java

Enumeration, Type Wrappers and Autoboxing

Enumeration

- An enumeration is a list of named constants
- Java enumerations is similar to enumerations in other languages with some differences
- In Java, an enumeration defines a class type
- In Java, an enumeration can have constructors, methods, and instance variables
- ***Example****: EnumDemo.java*

Enumeration

- All enumerations automatically contain two predefined methods:

  public static enum-type [ ] values( )

  – Returns an array that contains a list of the enumeration constants

  public static enum-type valueOf(String s)

  – Returns the enumeration constant whose value corresponds to the string passed in s

- ***Example****: EnumDemo2.java*

Enumeration

- Java enumeration is a class type
  - Although you can't instantiate an enum using new
- Enumeration can have constructors, instance variables and methods
  - Each enumeration constant is an object of its enumeration type
  - The constructor is called when each enumeration constant is created
  - Each enumeration constant has its own copy of any instance variables defined by the enumeration
- ***Example**: EnumDemo3.java*

Type Wrappers

- Despite the performance benefit offered by the primitive types, there are times when you will need an object representation
  - you can’t pass a primitive type by reference to a method
  - many of the standard data structures implemented by Java operate on objects, which means that you can’t use these data structures to store primitive types

Type Wrappers

- Java provides ***type wrappers***
  - classes that encapsulate a primitive type within an object
- The type wrappers are:
  - Character
  - Boolean
  - Double, Float, Long, Integer, Short, Byte

Type Wrappers

```
public class WrapDemo {
    public static void main(String args[]) {
        Integer iOb = new Integer(100);
        int i = iOb.intValue();
        System.out.println(i + " " + iOb);
    }
}
```

- The process of encapsulating a value within an object is called ***boxing***

  Integer iOb = new Integer(100);

- The process of extracting a value from a type wrapper is called ***unboxing***

  int i = iOb.intValue();

Auto (boxing/unboxing)

- ***Autoboxing***
  - the process by which a primitive type is automatically encapsulated into its equivalent type wrapper whenever an object of that type is needed
  - There is no need to explicitly construct an object
- ***Auto-unboxing***
  - the process by which the value of a boxed object is automatically extracted from a type wrapper when its value is needed
  - There is no need to call a method such as intValue() or doubleValue()

Autoboxing

- With autoboxing, it is no longer necessary to manually construct an object in order to wrap a primitive type
- You need only assign that value to a type-wrapper reference
- Java automatically constructs the object for you

  Integer iOb = 100; // autobox an int ***100***
- Notice that the object is not explicitly created through the use of new. Java handles this for you, automatically

Auto-unboxing

- To unbox an object, simply assign that object reference to a primitive-type variable

  int i = iOb; // auto-unbox

- Java handles the details for you
- ***Example****: AutoBoxingUnboxingDemo.java*