**Overriding Object.equals()**

The == comparison operator is used for comparing two primitive data type values or for determining whether two objects have the same references.

The equals method is intended to test whether two objects have the same contents, provided that the method is modified in the defining class of the objects.

The == operator is stronger than the equals method, in that the == operator checks whether the two reference variables refer to the same object.

The equals() method compares the  
contents of two objects. The default implementation of the equals method in the Object class is as follows:

public boolean equals(Object obj) {

return this == obj;

}

For example, the equals method is overridden in the Circle class.

public boolean equals(Object o) {

if (o instanceof Circle) {

return radius == ((Circle)o).radius;

}

else

return false;

}

The instanceof operator test whether an object is an instance of a class.

**What is Enum in Java**

Enum in Java is a keyword, a feature which is used to represent fixed number of well-known values in Java, For example, Number of days in Week, Number of planets in Solar system etc.

Enumeration (Enum) in Java was introduced in JDK 1.5 and it is one of my favorite features of J2SE 5 among Autoboxing and unboxing , Generics, varargs and static import. One of the common use of Enum which emerged in recent years is [Using Enum to write Singleton in Java](http://javarevisited.blogspot.gr/2012/07/why-enum-singleton-are-better-in-java.html), which is by far easiest way to implement Singleton and handles several issues related to thread-safety and Serialization automatically.

Java Enum as a type is more suitable to represent well known fixed set of things and state,  for example representing the state of Order as NEW, PARTIAL FILL, FILL or CLOSED.  
  
Enumeration(Enum) was not originally available in Java though it was available in another language like C and C++, but eventually, Java realized and introduced Enum on JDK 5 (Tiger) by keyword Enum.   
  
In past programs, you've probably created sets of constants to make your program easier to code and more readable. For example, maybe you've created a set of constants for the numbers of the days of the week:

ublic class EnumsExample {

    public static final int SUNDAY = 0;

    public static final int MONDAY = 1;

    public static final int TUESDAY = 2;

    public static final int WEDNESDAY = 3;

    public static final int THURSDAY = 4;

    public static final int FRIDAY = 5;

    public static final int SATURDAY = 6;

    public static void main(String[] args) {

        int dayOfWeek = 3;

        switch(dayOfWeek) {

            case SUNDAY:

                System.out.println("Sunday");

                break;

            case MONDAY:

                System.out.println("Monday");

                break;

            case TUESDAY:

                System.out.println("Tuesday");

                break;

            case WEDNESDAY:

                System.out.println("Wednesday");

                break;

            case THURSDAY:

                System.out.println("Thursday");

                break;

            case FRIDAY:

                System.out.println("Friday");

                break;

            case SATURDAY:

                System.out.println("Saturday");

                break;

            default:

                System.out.println("Invalid day of week number.");

        }

    }

}

This program works fine, however there are a couple of problems with it:

1. The variable dayOfWeek can be assigned any integer value. For example, the statement  
   int dayOfWeek = 99;  
   would not cause an error. Instead, the number would simply be considered invalid and will be caught by the default: clause of the switch statement.
2. Printing a constant's value is meaningless: printing the value of FRIDAY simply prints the integer 5. It would be better if it printed something meaningful like "Friday".
3. The constants are not reusable. You'd have to add them to some kind of utility class in a library so you could use them in other programs.

Creating an enumeration for the days of the week solves these issues and also adds a number of other conveniences that will make your code easier to use and read, less work, and more efficient.