New Java Features

Advanced Programming Techniques
Java 1.5

- Generics
- For-Each loop
- Autoboxing
- Enums
- Varargs
- Static import
- Annotations

http://java.sun.com/j2se/1.5.0/docs/guide/language/index.html
Generics

Generics provides a way to communicate the type of a collection to the compiler, so that it can be checked. The compiler can check that you have used the collection consistently and can insert the correct casts on values being taken out of the collection.
Example

Java 1.4:

// Removes 4-letter words from c. Elements must be strings
static void expurgate(Collection c) {
    for (Iterator i = c.iterator(); i.hasNext(); )
        if (((String) i.next()).length() == 4)
            i.remove();
}

Java 1.5:

// Removes the 4-letter words from c
static void expurgate(Collection<String> c) {
    for (Iterator<String> i = c.iterator(); i.hasNext(); )
        if (i.next().length() == 4)
            i.remove();
}
Checked Collections

- Automatically generated casts may fail if you accidentally mix types
- Checked collection wrappers can be used to locate the problem:

```java
Set<String> s = Collections.checkedSet(new HashSet<String>(), String.class);
```
For-Each loop

Cleaner way to iterate over collections and arrays

Java 1.4

```java
int sum(int[] a) {
    int result = 0;
    for (int i=0; i<a.length; i++)
        result += i;
    return result;
}
```

Java 1.5

```java
int sum(int[] a) {
    int result = 0;
    for (int i : a)
        result += i;
    return result;
}
```
Autoboxing

- Collections can only store objects, not primitive data types (int, etc.)
- When you want to store an int in a collection, you box it using a wrapper class Integer
- When you get the stored element from the collection, you have to unbox it to get your int back
- Autoboxing does this for you
Example

**Java 1.4:**

```java
public static void main(String argv[]) {  
    Vector v = new Vector();
    v.add(new Integer(1));
    v.add(new Integer(2));
    for (Iterator i = v.iterator(); i.nextNext(); )
        System.out.println(((Integer)i.next()).intValue());
}
```

**Java 1.5:**

```java
public static void main(String argv[]) {  
    Vector<Integer> v = new Vector();
    v.add(1);
    v.add(2);
    for (int i : v)
        System.out.println(i);
    }
```
Enums

Java 1.4 and prior did not support enumerations

The standard way to represent an enumerated type:

```java
public static final int SEASON_WINTER = 0;
public static final int SEASON_SPRING = 1;
public static final int SEASON_SUMMER = 2;
public static final int SEASON_FALL = 3;
```

What are the problems with this?
Several problems without explicit enums

- Not typesafe
- No namespace
- Brittle
- Printed values are uninformative
enum Season { WINTER, SPRING, SUMMER, FALL }

- Full-fledged class
- Comparable
- Serializable
- Printed values not ints