

Advanced Java Programming

Ohad Barzilay,
Tel-Aviv University
Spring '06

Java Syntax & Structures

(full version in the course web site)

Identifiers, Keywords, Literals
Separators and Operators

Control Structures and statements

Flow Control Statements

- `if` , `if - else`
- `while` , `do - while`
- `for`
- `switch`
- `break` , `continue`
- **Labeled `break`** , **Labeled `continue`**

Labeled continue and break

- In nested loops we might want to break/continue with respect to the outer loop
- As there is no goto in java – labels serves the missing feature

Labeled break

outer:

```
do {  
    statement1;  
    do {  
        statement2;  
        if ( condition ) {  
            break outer;  
        }  
        statement3;  
    } while ( test_expr );  
  
    statement4;  
  
} while ( test_expr );
```

Labeled continue

```
test:
    do {
        statement1;
        do {
            statement2;
            if ( condition ) {
                continue test;
            }
            statement3;
        } while ( test_expr );

        statement4;

    } while ( test_expr );
```

Packages and Import statements

Java API (Packages)

- ❑ Java comes with 3,000+ pre-designed components.
- ❑ The Java API is the library of classes supplied by Java.
- ❑ The classes in the Java API is separated into packages. Each package contains a set of classes that are related in some way.

The Java API Packages

`java.applet`

`java.awt`

`java.beans`

`java.io`

`java.lang`

`java.math`

`...`

`java.net`

`java.rmi`

`java.security`

`java.sql`

`java.text`

`java.util`

Documentation:

<http://java.sun.com/j2se/1.5.0/docs/api/>

The screenshot shows the Java™ 2 Platform, Standard Edition, v 1.4.0 API Specification page. The browser window title is "All Classes - Microsoft Internet Explorer". The address bar shows the URL <http://java.sun.com/j2se/1.4/docs/api/>. The page content includes a navigation menu with "Overview", "Package", "Class", "Use", "Tree", "Deprecated", "Index", and "Help". The main heading is "Java™ 2 Platform, Standard Edition, v 1.4.0 API Specification". Below the heading, it states "This document is the API specification for the Java 2 Platform, Standard Edition, version 1.4." and "See: [Description](#)".

Three callouts highlight specific features:

- List of Packages:** A callout pointing to the left sidebar, which lists various Java packages such as `java.awt.image.renderab`, `java.awt.print`, `java.beans.beancontext`, `java.io`, `java.lang`, `java.lang.ref`, `java.lang.reflect`, and `java.math`.
- List of Classes:** A callout pointing to the "All Classes" section in the left sidebar, which lists numerous classes including `AbstractAction`, `AbstractBorder`, `AbstractButton`, `AbstractCellEditor`, `AbstractCollection`, `AbstractColorChooserPa`, `AbstractDocument`, `AbstractDocument.Attrib`, `AbstractDocument.Cont`, `AbstractDocument.Elem`, `AbstractInterruptibleChar`, `AbstractLayoutCache`, `AbstractLayoutCache.No`, `AbstractList`, `AbstractListModel`, `AbstractMap`, `AbstractMethodError`, `AbstractPreferences`, `AbstractSelectableChan`, and `AbstractSelectionKey`.
- Details of Packages:** A callout pointing to the "Java 2 Platform Packages" table, which provides descriptions for several packages:

Java 2 Platform Packages	
java.applet	Provides the classes necessary to create an applet and the classes an applet uses to communicate with its applet context.
java.awt	Contains all of the classes for creating user interfaces and for painting graphics and images.
java.awt.color	Provides classes for color spaces.
java.awt.datatransfer	Provides interfaces and classes for transferring data between and within applications.
java.awt.dnd	Drag and Drop is a direct manipulation gesture found in many Graphical User Interface systems that provides a mechanism to transfer information between two entities logically associated with presentation elements in the GUI.
java.awt.event	Provides interfaces and classes for dealing with different types of events fired by AWT components.

java.lang

Choose
java.lang
from list of
Packages

List of
Classes
defined in
Package

Overview **Package** Class Use Tree Deprecated Index Help

PREV PACKAGE NEXT PACKAGE FRAMES NO FRAMES

Package java.lang

Provides classes that are fundamental to the design of the Java programming language.

See:

- [Description](#)

Interface Summary

CharSequence	A CharSequence is a readable sequence of characters.
Cloneable	A class implements the Cloneable interface to indicate to the <code>Object.clone()</code> method that it is legal for that method to make a field-for-field copy of instances of that class.
Comparable	This interface imposes a total ordering on the objects of each class that implements it.
Runnable	The Runnable interface should be implemented by any class whose instances are intended to be executed by a thread.

Class Summary

Boolean	The Boolean class wraps a value of the primitive type <code>boolean</code> in an object.
Byte	The Byte class wraps a value of primitive type <code>byte</code> in an object.
Character	The Character class wraps a value of the primitive type <code>char</code> in an object.
Character.Subset	Instances of this class represent particular subsets of the Unicode character set.

Details of
Classes

String Class

The screenshot shows the Java documentation for the `String` class. At the top, there are navigation links: [Overview](#), [Package](#), **Class**, [Use Tree](#), [Deprecated](#), [Index](#), and [Help](#). On the right, it says "Java™ 2 Platform Std. Ed. v1.4.0". Below the navigation, there are links for [PREV CLASS](#), [NEXT CLASS](#), [FRAMES](#), and [NO FRAMES](#). A summary line includes [FIELD](#), [CONSTR](#), and [METHOD](#). The main content area shows the package `java.lang` and the class `String` extending `Object`. It lists implemented interfaces: `CharSequence`, `Comparable`, and `Serializable`. Below this, the class signature is shown: `public final class String` extending `Object` and implementing `Serializable`, `Comparable`, and `CharSequence`. A paragraph explains that `String` represents character strings and that string literals are instances of this class. Another paragraph states that strings are constant and immutable, and can be shared.

Class Hierarchy

`java.lang`
Class String
[java.lang.Object](#)
|
+--`java.lang.String`

All Implemented Interfaces:
[CharSequence](#), [Comparable](#), [Serializable](#)

`public final class String`
extends [Object](#)
implements [Serializable](#), [Comparable](#), [CharSequence](#)

The `String` class represents character strings. All string literals in Java programs, such as "abc", are implemented as instances of this class.

Strings are constant; their values cannot be changed after they are created. String buffers support mutable strings. Because `String` objects are immutable they can be shared. For example:

String Methods

Method Summary

char	charAt (int index) Returns the character at the specified index.
int	compareTo (Object o) Compares this String to another Object.
int	compareTo (String anotherString) Compares two strings lexicographically.
int	compareToIgnoreCase (String str) Compares two strings lexicographically, ignoring case considerations.
String	concat (String str) Concatenates the specified string to the end of this string.
boolean	contentEquals (StringBuffer sb) Returns true if and only if this String represents the same sequence of characters as the specified StringBuffer.
static String	copyValueOf (char[] data) Returns a String that represents the character sequence in the array specified.
static String	copyValueOf (char[] data, int offset, int count) Returns a String that represents the character sequence in the array specified.
boolean	equalsIgnoreCase (String str)

Methods List

Importing Packages

- Using a class from the Java API can be accomplished by using its fully qualified name:

```
java.util.Random random =  
    new java.util.Random();
```

- Or the class can be imported once with the import statement at the top of the file:

```
import java.util.Random;  
.  
.  
.  
Random random = new Random();
```

Importing Packages

- You can also import all the classes in a given package with a single import statement:

```
import java.util.*;
```

- The `java.lang` package is automatically imported into every Java program.

Arrays

Arrays

- ❑ An array is an object that can be used to store a list of values.
- ❑ All array elements are of the same type (primitive or objects).
- ❑ Arrays have fixed sizes, set when the array is created.

Array Elements

- A particular value in an array is referenced using the array name followed by the index in brackets.
- As in C, a java Array of size n is indexed from 0 to $n-1$.
- The Java interpreter will throw an exception if an array index is out of bounds.

Creating Arrays

- Array elements are initialized with their default values

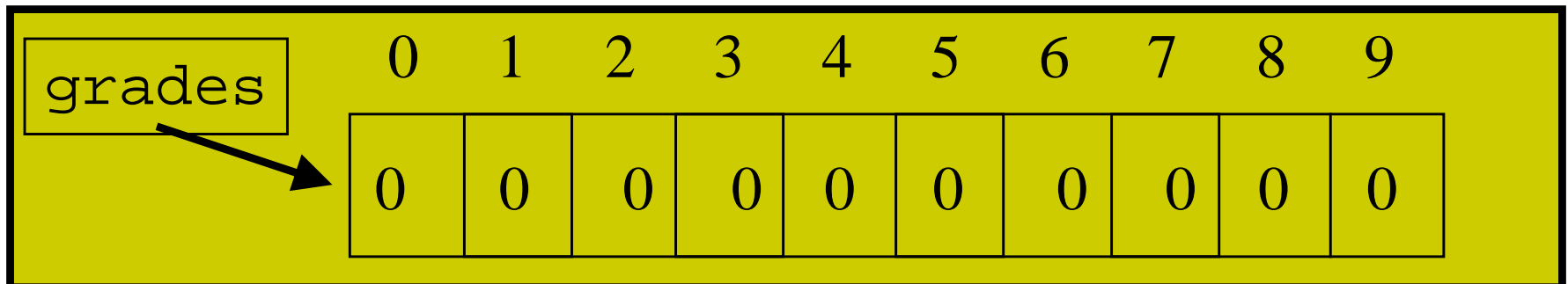
```
int[] grades = new int[10];
```

```
grades[3] = 70;
```

```
grades[7] = 87;
```

```
int i = 5;
```

```
grades[i/2] = 39;
```



Initialization list

- You can declare, construct, and initialize the array all in one statement:

```
int[] primes = {2, 3, 5, 7, 11, 13, 17, 19};
```

- This declares an array of type `int`, constructs an array of 8 slots, and assigns the designated values into the array.

```
int[][] values = {{1, 2, 5}, {4, 3, 2, 1}, {11}};
```