Java

Object-oriented fundamentals:

Introduction to classes, objects, methods

Lecture objectives

To be able to understand the following fundamental concepts of the Java programming language:

- Preliminary understanding of classes, objects and methods
- Creating objects
- The String class
- Libraries and packages
- The import declaration
- Class methods: Math
- Dialog boxes
- data conversions and Wrapper classes

Objects and Classes

A class
(the concept)

An object
(the state or realization)

Bank Account

John's Bank Account
Balance: $5,257

Bill's Bank Account
Balance: $1,245,069

Mary's Bank Account
Balance: $16,833

Muliple objects
from the same class

(the behaviours or methods)

deposit, withdraw, get balance

String class

- The `String` class is one of the most important prewritten classes that come with Java
- Every character string delimited by double quotation marks, represents a `String` object
- Note that class names always start with a capital letter

`String` objects

- Any object is defined by its state
- Each `String` object is defined by specific characters
  - eg “hello”
  - “abcdefg123456”

- Each `String` object can have methods that can do useful things to the string of characters
  - eg ?

Creating String Objects

- A variable either holds
  - a primitive type,

```
int totalValue;
```

  - or it holds an object

```
String title;
```

- A *class* name can be thought of as a *type* to declare an *object*

```
String title;
title =  "Java Software Solutions”;
```

- Or just like primitive variables, combine the 2 steps

```
String title = "Java Software Solutions";
```

Methods

- Methods are easy to recognise –
  - An identifier that is always followed by ( )
- Once an object has been created, we can do things to it by calling the methods in the object's class

- In general a method call takes this form:

```
object . method-name (parameters)
```

- Parameters (or arguments) are data that the method requires
- Methods may require
  - no parameter, one parameter, many parameters

Back to the String class

- The String class has many methods we can use to manipulating strings

```
length()
toUppercase()
toLowerCase()
```

- But we need a String object to apply the methods to

```
String title = “Java Software Solutions”;
```

The dot operator

- Because `title` is an object of the *String* class, it can use all the methods defined in that class

- Use the *dot operator* to invoke its methods eg

```
title.length()
title.toLowerCase()
```

- What is the result or output of these methods?
- But what happens to the result or output of the methods?

Methods that return results

- We need to define a variable to assign the returned value

```
String title = “Java Software Solutions”;
int lengthOfTitle = title.length();
```

- How do we know if a method returns anything?
- How do we know the type of the returned data?

- This is indicated in the method documentation

Method Documentation

- Read method documentation (description) to understand
  - What methods are available for each class
  - How each method works

- See the list of (for example) methods of the `String` class:
  - In your text chapters
  - The online Java API documentation

String Methods : *length()*

- The `length` method returns an `integer` of the number of characters in the `String` object

```
int length ( )
```

This is the description of a method i.e it tells us how to use the *length* method

- This method only requires the information contained in the *object* on which it is operating ie `title`.

```
int  lengthOfTitle = title.length();
```

This is the method actually being used in a program

```
//*********************************************************************
//  MethodTest1.java
// Author: J.Terry
// Date: 8/8/2
// Demonstrates the use of the length() method of the String class
//*********************************************************************

public class MethodTest1
{
     public static void main (String[ ] args)
     {
        int songLength;
        String songTitle = “Redundant";

        songLength = songTitle.length();

        System.out.println (“The song title ” + songTitle + “has“ + songLength + “ characters”);

    }
}
```

The song title Redundant has 9 characters

String Methods: toLowerCase, toUpperCase

- `toLowerCase()` and `toUpperCase()` convert any letters in a *string* to lowercase or uppercase

return type

```
String toUpperCase ( )
String toLowerCase ( )
```

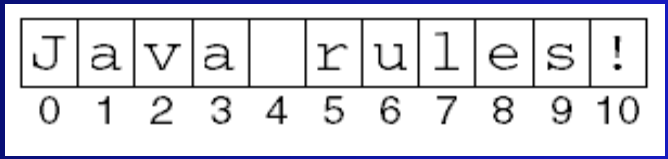
- Similar to `length()` but return type is a *String*
  - a new `String` containing the converted letters

```
//*********************************************************
//  MethodTest2.java
// Author: J.Terry
// Date: 8/8/2
// Demonstrates the use of the toUpperCase() method of the String class
//*********************************************************

public class MethodTest2
{
     public static void main (String[ ] args)
    {
        String bandName = “Blink 182";

        String bandUpper = bandName.toUpperCase();

        System.out.println (“The band name ” + bandName + “ in upper case is “ + bandUpper);

    }
}
```

The band name Blink 182 in upper case is BLINK 182

Visualizing a String

- A `String` object can be visualized as a series of characters,
- Each character is identified by its position
- The position of a character can be called its *index*
- The first character is located at position 0



Method with one parameter

- Say we want to know what the character is at a certain position in a string

- What information do we get from the *object* eg `title`
- What further information do we need?

String Methods: charAt

The `charAt()` method:

```
char charAt (int index)
```

- Requires an integer that represents the index of the character in the *String* you wish to find
- returns the *character* stored at the stated index position
- The required type `int` and variable `index` describes a *parameter variable* definition

```
//*********************************************************
//  MethodTest3.java
// Author: J.Terry
// Date: 8/8/2
// Demonstrates the use of the toUpperCase() method of the String class
//*********************************************************

public class MethodTest3
{
     public static void main (String[ ] args)
    {
         String slogan = “Java rules!";
         char ch;
         int index = 6;

        ch = slogan.charAt(index);

        System.out.println (“The character at index position “ + index + “ is ” + ch);

    }
}
```

The character at index position 6 is u

Class Methods

- Some methods can be invoked through the class name, instead of through an object of the class
- These methods are called *class* methods or *static* methods
- The `Math` class contains many static methods, providing various mathematical functions, such as
  - absolute value, trigonometry functions, square root, etc.
- These methods are invoked by the class name

```
Class . method-name (parameters)
```

The `sqrt` Method

`sqrt()` computes the square root of a `double`:

```
static double sqrt (double num)
```

```
Math.sqrt(2.0)          // 1.4142135623730951
Math.sqrt(4.0)          // 2.0 is returned
```

```
double testSqRoot = Math.sqrt(2.0);
```

The `abs` Method

- The `abs()` method computes the absolute value of a number:

```
static double abs (double num)
static int abs (int num)
```

```
Math.abs(2.0)          // 2.0 is returned
Math.abs(-2.0)         // 2.0 is returned
Math.abs(2)            // 2 is returned
Math.abs(-2)           // 2 is returned
```

```
double testAbs = Math.abs(-35.82);
```

Method with 2 parameters

- The `max` method finds the larger of two numbers (`min` is similar):

```
static double max (double num1, double num2)
static int max (int num1, int num2)
```

```
Math.max(3.0, 5.5)          // 5.5 is returned
Math.max(10.0, -2.0)        // 10.0 is returned
Math.max(-5, -2)            // –2 is returned
```

```
double testMax = Math.max(2.0, 3.0);
System.out.println("The larger of 2.0 and 3.0 = " + testMax);
```

The `pow` Method

- The `pow()` method raises a number to a power:

```
static double pow (double num, double power)
```

```
Math.pow(2.0, 3.0)        // 8.0 is returned
Math.pow(-2.0, 3.0)       // –8.0 is returned
Math.pow(2.0, -1.0)       // 0.5 is returned
```

```
double testCube= Math.pow(2.0, 3.0);
```

The `random` Method

- The `random` method of the `Math` class returns a random `double` greater than or equal to 0.0 and less than 1.0:

  `static double random()`

Eg

```
double randomNumber = Math.random();
```

- How would you get a random number that is either 0 or 1?

```
int randomBinary = (int)(Math.random() * 2);
```

- What about a random number that is 1 to 6 eg for tossing a die

```
int randomDie = (int)(Math.random() * 6) + 1;
```

Using the Result of a Method Call

- The value returned by a method can be saved in a variable:

```
double y = Math.abs(x);
```

- Or use the result returned by a method directly, without first saving it in a variable.

```
double z = Math.sqrt(Math.abs(x));

System.out.println(Math.sqrt(2.0));
```

Packages/Classes

- Classes are in packages eg.

Package	Purpose
`java.lang`	General support
`java.applet`	Creating applets for the web
`java.awt`	Graphics and graphical user interfaces
`javax.swing`	Additional graphics capabilities and components
`java.net`	Network communication
`java.util`	Utilities

- Import the classes you need at the start of the program

```
import javax.swing.JOptionPane;        or
import javax.swing.*;
```

The import Declaration

- The `java.lang` package contains classes such as `System`, `String` and `Math`

- It is the only package that does not need to be imported,
  - as though the folloing was written in every program

```
import java.lang.* ;
```

API Specification

- The Application Programming Interface (API) specification is the online documentation of the Java language
- Follow the link from CSP1150 on *ecourse* to
  - The Sun java homepage at `http://java.sun.com/`
  - Click link to API Specification
  - Click the J2SE API Specification

- We can only skim the surface of classes and their methods etc in lectures

Java 2 Platform SE v1.3.1 - Microsoft Internet Explorer

File Edit View Favorites Tools Help

Back Forward Stop Refresh Home Search Favorites Media History Mail Print Edit Discuss

Address C:\jdk1.3.1_02\j2sdk-1_3_1-doc\docs\api\index.html Go Links

Java™ 2 Platform Std. Ed. v1.3.1

All Classes

Packages

java.applet
java.awt

All Classes

AbstractAction
AbstractBorder
AbstractButton
AbstractCellEditor
AbstractCollection
AbstractColorChooserPan
AbstractDocument
AbstractDocument.Attrib
AbstractDocument.Conte
AbstractDocument.Elemen
AbstractLayoutCache
AbstractLayoutCache.No
AbstractList
AbstractListModel
AbstractMap
AbstractMethodError
AbstractSequentialList
AbstractSet
AbstractTableModel

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

PREV NEXT FRAMES NO FRAMES

Java™ 2 Platform Std. Ed. v1.3..

Java™ 2 Platform, Standard Edition, v 1.3.1 API Specification

This document is the API specification for the Java 2 Platform, Standard Edition, version 1.3.1.

See:

Description

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.

My Computer

Java 2 Platform SE v1.3.1 - Microsoft Internet Explorer

File Edit View Favorites Tools Help

Back Forward Stop Refresh Home Search Favorites Media History Mail Print Edit Discuss

Address C:\jdk1.3.1_02\j2sdk-1_3_1-doc\docs\api\index.html Go Links

Java™ 2 Platform Std. Ed. v1.3.1

All Classes

Packages

java.applet
java.awt

MarshalledObject
Math
MatteBorder
MediaTracker
Member
MemoryImageSource
Menu
MenuBar
MenuBarUI
MenuComponent
MenuContainer
MenuDragMouseEvent
MenuDragMouseListener
MenuElement
MenuEvent
MenuItem
MenuItemUI
MenuKeyEvent
MenuKeyListener
MenuListener
MenuSelectionManager

Field Summary

static double	**E** The double value that is closer than any other to e, the base of the natural logarithms.
static double	**PI** The double value that is closer than any other to *pi*, the ratio of the circumference of a circle to its diameter.

Method Summary

static double	**abs**(double a) Returns the absolute value of a double value.
static float	**abs**(float a) Returns the absolute value of a float value.
static int	**abs**(int a) Returns the absolute value of an int value.
static long	**abs**(long a) Returns the absolute value of a long value.
static double	**acos**(double a) Returns the arc cosine of an angle, in the range of 0.0 through *pi*.
static double	**asin**(double a) Returns the arc sine of an angle, in the range of -*pi*/2 through *pi*/2.
static double	**atan**(double a) Returns the arc tangent of an angle, in the range of -*pi*/2 through *pi*/2.
static double	**atan2**(double a, double b) Converts rectangular coordinates (b, a) to polar (r, *theta*).

My Computer

Dialog Boxes

- A *dialog box* is a graphical window that pops up on top of any currently active window for the user
- Import the `JOptionPane` class from the *swing* package

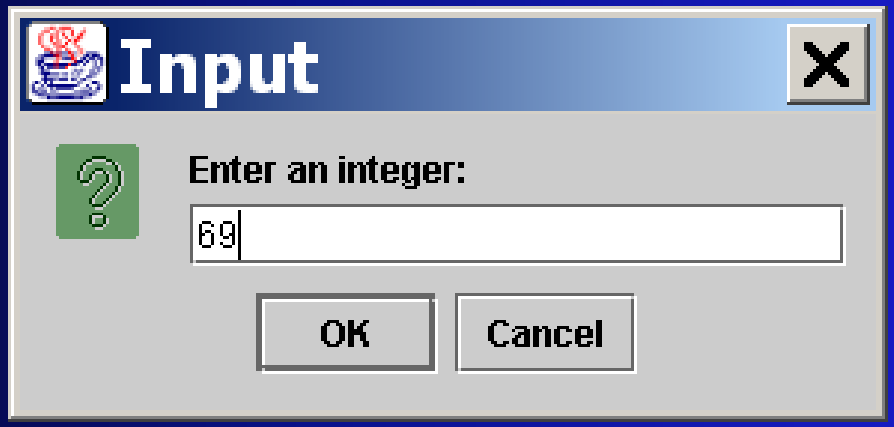
```
import javax.swing.JOptionPane;    or
import javax.swing.*;
```

Getting Input

To display a dialog box with the specified message and an input text field. The contents of the text field are returned.

- For example

```
String numStr = JOptionPane.showInputDialog ("Enter an integer:");
```



Displaying an Output Message

- To display a dialog box containing the specified message. (*null* centres the box on the screen)

- For example

```
JOptionPane.showMessageDialog (null, result);
```

Data Conversions

- In Java, each data value is associated with a type
  - short, int, double, char etc.
- Sometimes we need to convert a data value to another type
- Data conversions can occur in three ways:
  - assignment conversion
  - arithmetic promotion
  - casting

Assignment conversion - Coercion

- *Assignment conversion* or *coercion* occurs when a value of one type is assigned to a variable of another
- Only use widening conversions are permitted
  - (eg. an `int` to a `double`)

```
int exampleInteger = 99;

double  exampleDouble;
exampleDouble = exampleInteger;
```

The integer will be converted to a double and will be stored as one: (`exampleDouble = 99.0`)

- Narrowing conversions eg `double` to `int` cause a compiler error

Arithmetic promotion

- *Arithmetic promotion* happens automatically when operators in expressions are different data types
- Java converts operands to a type that will safely accommodate both values
- For expressions using integer and floating-point values:

```
anInteger = aDouble * anInteger;
```

  - The integer value is temporarily converted to a double type
  - The operation is performed
  - The result is a double
  - It is stored according to the receiving variable type

Another example

```
int sum = 50;
int count = 100;
double average;

average = sum/count;
```

- What is the value of average?
- It is 0.0
- Because the expression is not of mixed types, integer division occurs with the result 0

Casting

- Is done explicitly by the programmer – so is clear
- Both widening and narrowing conversions can be accomplished by explicitly casting a value
- To cast, the type is put in parentheses in front of the value being converted

```
int sum = 50;
int count = 100;
double average;

average = (double)sum/count;
```

- Floating-point division is enforced

Wrapper Classes

- There is a wrapper class in the `java.lang` package for each primitive type:

Primitive Type	Wrapper Class
`byte`	`Byte`
`short`	`Short`
`int`	`Integer`
`long`	`Long`
`float`	`Float`
`double`	`Double`
`char`	`Character`
`boolean`	`Boolean`

Wrapper Classes

- Contain static methods that convert Strings to the associated type eg.

```
static int parseInt (String str)

static double parseDouble (String str)
```

- eg `Integer` and `Double` classes methods convert an integer or double stored in a `String` :

```
String aString = “123”;
int num1 = Integer.parseInt (aString);

String anotherString = “123.999”;
double num2 = Double.parseDouble(anotherString);
```

```
//  Demonstrates  Dialog boxes and Wrapper class (if-else covered next week)

import javax.swing.JOptionPane;
public class EvenOdd
{
   public static void main (String[] args)
   {
         String numStr, result;
         int num;

         numStr = JOptionPane.showInputDialog ("Enter an integer: ");
         num = Integer.parseInt(numStr);

         if (num%2 == 0)
               result = "That number is even";
         else
               result = "That number is odd";

         JOptionPane.showMessageDialog (null, result);
          System.exit(0);   // use this with dialog boxes to ensure program ends
   }
}
```

Workshop 3

Workshop Assessments

- as per the assessment schedule, the assessment programs from weeks 2 and 3 will be handed in and marked next week
- you must use the Workshop Assessment Submission Template to submit your work. It is in the *workshop assessment* document on ecourse.
- It must be handed to your tutor at the commencement of the workshop
- you must be present and be ready to answer questions about the programs

Lecture Outcomes

Today we have covered:

- Preliminary understanding of classes, objects and methods
- Creating objects
- The String class
- Libraries and packages
- The import declaration
- Class methods: Math
- Dialog boxes
- Data conversions and Wrapper classes

- Questions?