Session 03: Operators and Simple Dialog Boxes

Dr Tomasz Kurowski
t.j.kurowski@cranfield.ac.uk

Summary

More Operators

The Message Box

Command Line Parameters

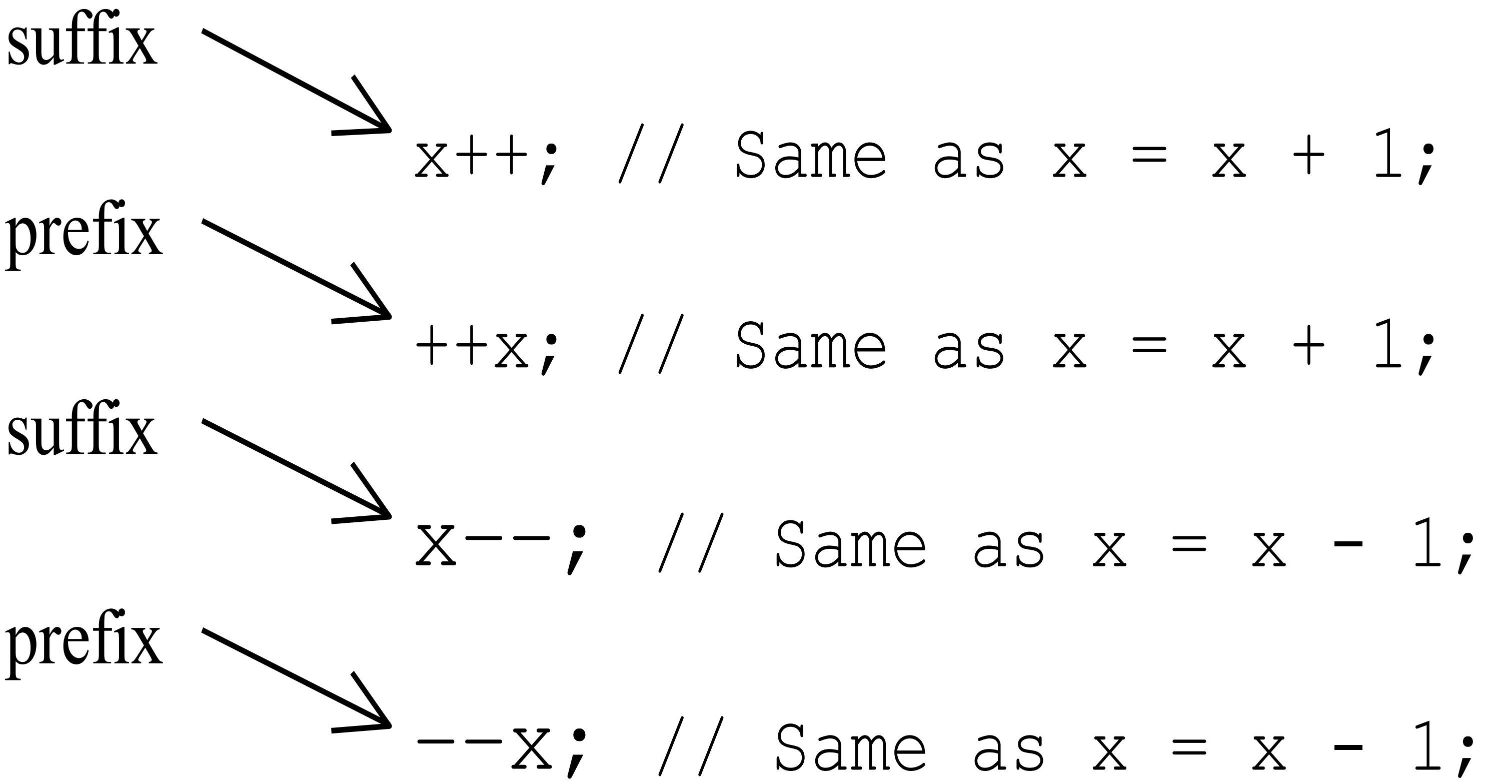
Boolean Data Type

Conditional Operators

Input dialog Box

Programming Style

Increment and Decrement Operators

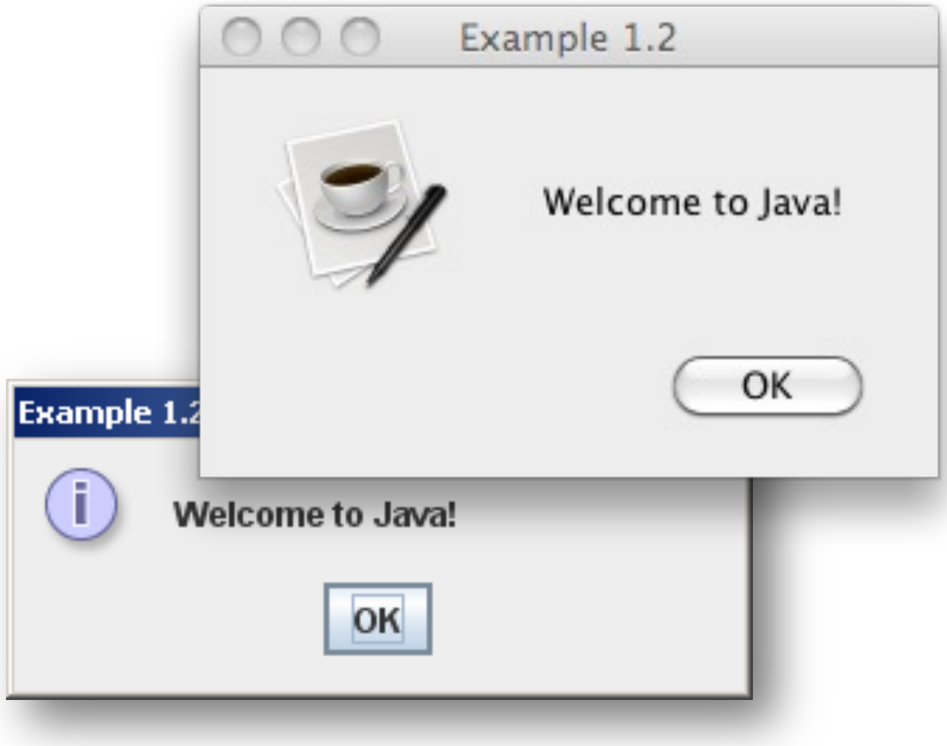


```
x++; // Same as x = x + 1;
++x; // Same as x = x + 1;
x--;  // Same as x = x - 1;
--x;  // Same as x = x - 1;
```

Shortcut Assignment Operators

Operator	*Example*	*Equivalent*
`+=`	`i+=8`	`i = i+8`
`-=`	`f-=8.0`	`f = f-8.0`
`*=`	`i*=8`	`i = i*8`
`/=`	`i/=8`	`i = i/8`
`%=`	`i%=8`	`i = i%8`

The showMessageDialog Method



```
JOptionPane.showMessageDialog(null, "Welcome to Java!",
  "Example 1.2", JOptionPane.INFORMATION_MESSAGE));
```

Displaying Text in a Message Dialog Box

As well as outputting to the system console device, you can use the showMessageDialog method to create a pop-up window (Dialog Box) to display your output.

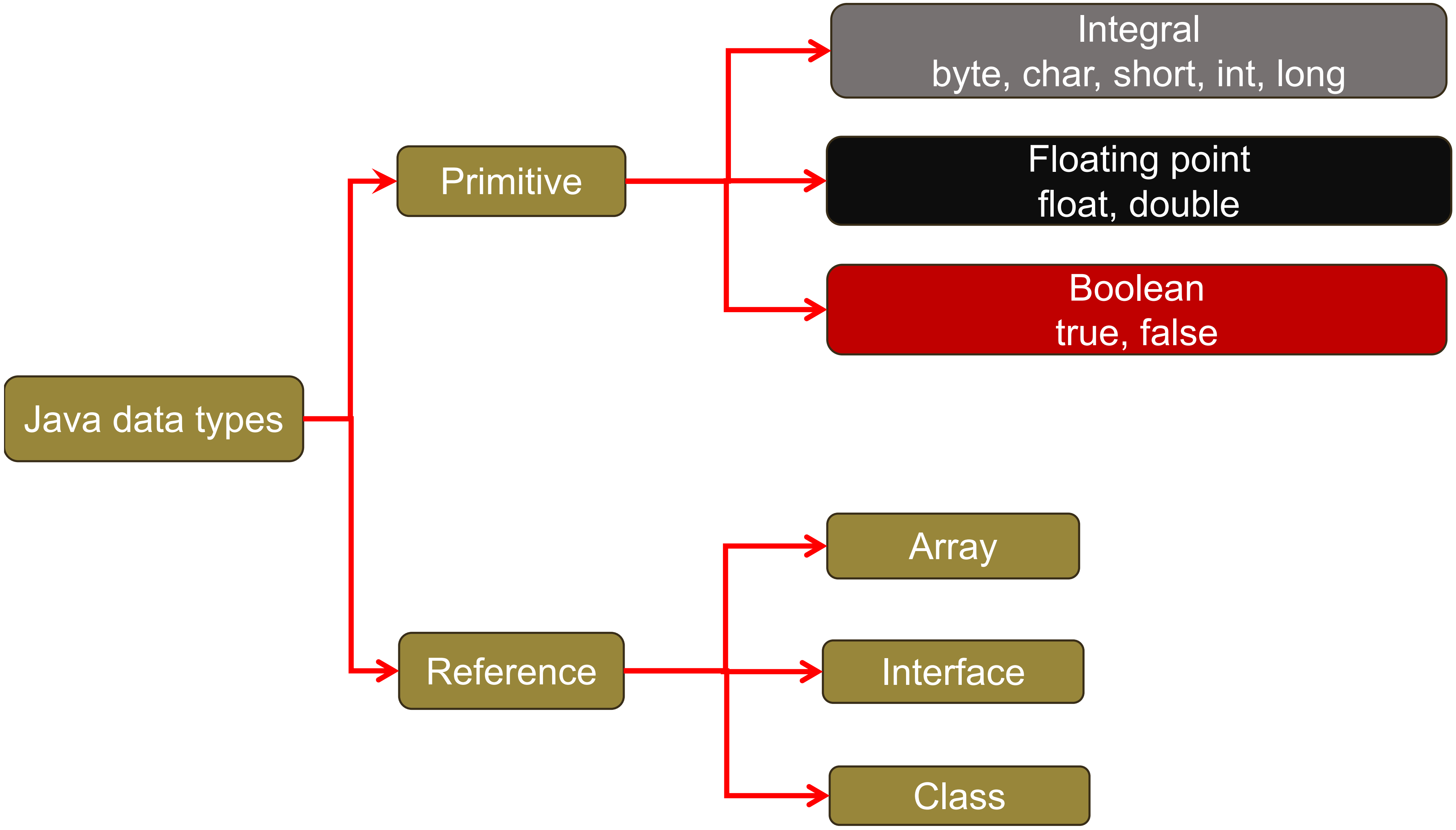
showMessageDialog is a member of the JOptionPane class. Part of the Swing GUI package.

JOptionPane is one of the many predefined classes in the Java system, which can be reused rather than “reinventing the wheel.”

Source

Run

Java data types



The `boolean` Type and Operators

- Boolean types can only be true or false
- Comparisons return a result of type boolean

```
boolean lightsOn = true;
boolean lightsOn = false;
boolean b = (1 > 2);
```

- && (and) (1 < x) && (x < 100)
- `||` **(or)** (lightsOn) || (isDayTime)
- ! (not) !(isStopped)

Comparison Operators

Operator	**Name**
<	less than
<=	less than or equal to
>	greater than
>=	greater than or equal to
==	equal to
!=	not equal to

Comparison Operators - Examples

x=6, y=3; if(x == y)	FALSE
x=6, y=6; if(x == y)	TRUE
x=5, y=2; if(x != y)	TRUE
x=6, y=5; if(x < y)	FALSE
x=7, y=6; if(x > y)	TRUE
x=6, y=6; if(x >= y)	TRUE
x=6, y=6; if(x = y)	ERROR

Boolean Operators - Examples

For example, a=2, b=4, x=6, y=6;

if (a < b) && (x == y)	TRUE
if (a == b) && (x == y)	FALSE
if (a == b) || (x == y)	TRUE
if (a == b) || (x > y)	FALSE
if !(a == b)	TRUE
if !(a<= b)	FALSE

Using Conditional Operators

```
double radius

...

if (radius > 0) {
   area = radius*radius*PI;
  System.out.println("The area for the “
      + “circle of radius " + radius +
      " is " + area);
}
else {
   System.out.println(“Invalid radius”);
}
```

Command-Line Parameters

You can supply command line arguments when you invoke a Java program :

```
C:>java TestMain arg0 arg1 arg2 ... argn
```

These arguments can be picked up within the application as a set of strings :

```
class TestMain {
  public static void main(String[] args) {
   ...
   }
}
```

This is a simple way to enter data into a program

Processing Command-Line Parameters

In the main method, get the arguments from `args[0], args[1], ..., args[n]`, which corresponds to `arg0, arg1, ..., argn` in the command line.

You can check if arguments have been passed in by :

```
    if (args.length > 0) {.. my code ..}
```

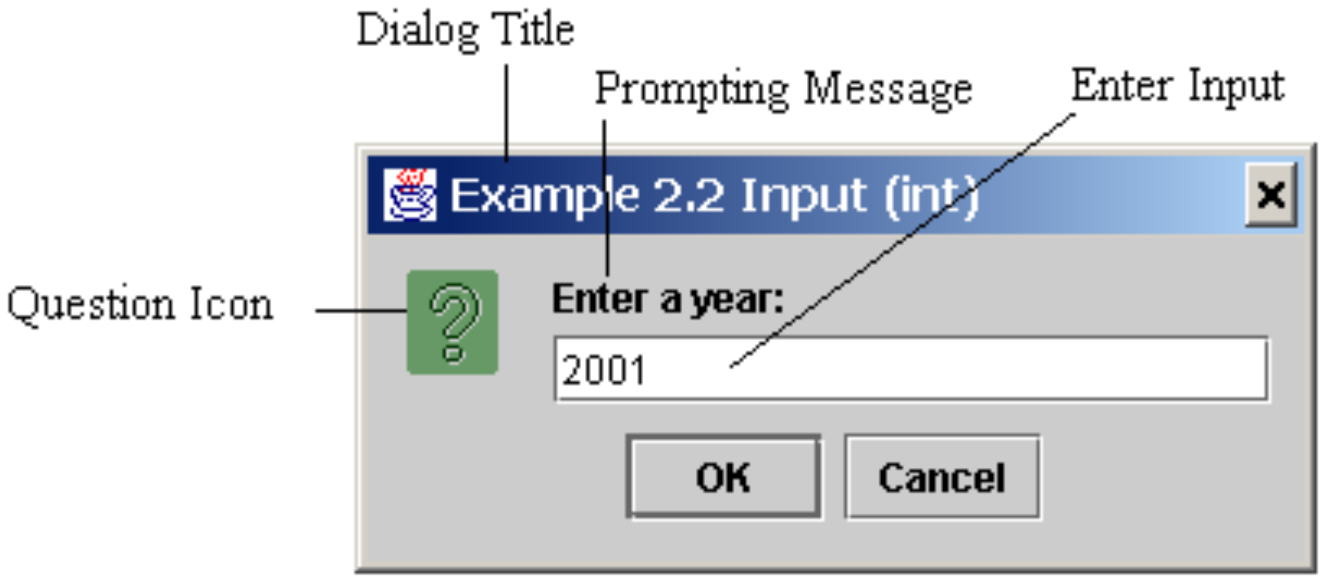
Remember – these arguments are strings, you will need additional code to convert to other types such as integers.

```
    int val = Integer.parseInt(args[1]);
```

Getting Input from Input Dialog Boxes

```
String s = JOptionPane.showInputDialog(
     null, "Prompt Message",
     "Dialog Title",
     JOptionPane.QUESTION_MESSAGE
);
```

where x is a string for the prompting message and y is a string for the title of the input dialog box.



Converting Strings to Integers

The input returned from the input dialog box is a string. If you enter a numeric value such as 123, it returns “123”. To obtain the input as a number, you have to convert a string into a number.

To convert a string into an int value, you can use the static parseInt method in the Integer class as follows:

```
int intValue = Integer.parseInt(integerString);
```

where integerString is a numeric string such as “123”.

Converting Strings to Doubles

To convert a string into a double value, you can use the static parseDouble method in the Double class as follows:

double doubleValue =Double.parseDouble(doubleString);

where doubleString is a numeric string such as “123.45”.

Entering Input from Dialog Boxes

This program first prompts the user to enter a year as an int value and checks if it is a leap year, it then prompts you to enter a double value and checks if it is positive.

A year is a leap year if it is divisible by 4 but not by 100, or if it is divisible by 400.

InputDialogDemo

Run

The exit Method

Use Exit to terminate the program and stop all threads.

NOTE: When your program starts, a thread is spawned to run the program.

When the showMessageDialog is invoked, a separate thread is spawned to run this method.

This thread is not terminated even when you close the dialog box.

To terminate the thread, you have to invoke the exit method.

Programming Style and Documentation

- Appropriate Comments
- Naming Conventions
- Proper Indentation and Spacing Lines
- Block Styles

Appropriate Comments

Include a summary at the beginning of the program to explain what the program does, its key features, its supporting data structures, and any unique techniques it uses.

Include your name, class section, instruction, date, and a brief description at the beginning of the program.

Naming Conventions

- Choose meaningful and descriptive names.
- Variables and method names:
  - Use lowercase. If the name consists of several words, concatenate all in one, use lowercase for the first word, and capitalize the first letter of each subsequent word in the name. For example, the variables `radius` and `area`, and the method `computeArea`.

Naming Conventions, cont.

- Class names:
  - Capitalize the first letter of each word in the name. For example, the class name `ComputeArea`.
- Constants:
  - Capitalize all letters in constants. For example, the constant `PI`.

Proper Indentation and Spacing

- Java source code can be free format – as much white space, as many newlines as required.
- Java is case sensitive : ix, IX not the same.
- Identifiers / keyword separated by white space
- Be careful with String literals.
- Program layout helps readability & maintenance
- Recommendation - Indentation
  - Indent two spaces.
- Recommendation - Spacing
  - Use blank line to separate segments of the code.

Block Styles

Use end-of-line style for braces.

Next-line style

```
public class Test
{
  public static void main(String[] args)
  {
    System.out.println("Block Styles");
  }
}
```

End-of-line style

```
public class Test {
  public static void main(String[] args) {
    System.out.println("Block Styles");
  }
}
```

Unreadable Format

```
import javax.swing.JOptionPane; public class TestSum
 { /** Main method */ public static void
 main(String[] args) { /* Initialize sum */ float sum
 = 0; /* Keep adding 0.01 to sum*/ for (float
 i=0.01f; i <= 1.0f; i = i+0.01f) sum += i; /*
 Display result */
 JOptionPane.showMessageDialog(null,
    "The summation is " + sum, "Example 3.3 Output",
 JOptionPane.INFORMATION_MESSAGE); System.exit(0); }
 }
```

Readable Format

```
import javax.swing.JOptionPane;
public class TestSum {
   public static void main(String[] args) {  /* Main method */
        float sum = 0;                       /* Initialize sum */
      // Keep adding 0.01 to sum
      for (float i=0.01f; i <= 1.0f; i = i+0.01f)
         sum += i;
      // Display result
     JOptionPane.showMessageDialog (null,
              "The summation is    " + sum, "Example 3.3 Output",
               JOptionPane.INFORMATION_MESSAGE);
       System.exit(0);
   }
}
```

Programming Errors

✗

Syntax Errors

Detected by the compiler

⚠

Runtime Errors

Causes the program to abort

✓

Logic Errors

Produces incorrect result

Compilation Errors

```
public class ShowSyntaxErrors {
  public static void main(String[] args) {
    i = 30;
    System.out.println (i);
  }
}
```

Runtime Errors

```
public class ShowRuntimeErrors {
  public static void main(String[] args)
 {
    int i = 1 / 0;
   }
}
```

Logic Errors

```
public class ShowLogicErrors {
   // Determine if a number is between 1 and 100 inclusively
   public static void main(String[] args) {
      // Prompt the user to enter a number
      String input = JOptionPane.showInputDialog(null, "Enter an integer
 between 1 and 100:",
            "ShowLogicErrors", JOptionPane.QUESTION_MESSAGE);

   // Convert from string to integer
      int number = Integer.parseInt (input);

      // Display the result .. but what about boundary conditions ???
 boolean res = (1 < number) || (number < 100);
      System.out.println ("Is between 1 and 100 inclusively ?, " + res);
      System.exit(0);
   }
}
```