CS 335
Java Programming – Controls

Fall 2007

Java Control Structures

- Selection:
  - If, If/Else, Switch
- Repetition (looping):
  - While, For, Do/While
- Assignment:
  - Expressions, increment/decrement

Java Reserved Words

Java Keywords

abstract	boolean	break	byte	case
catch	char	class	continue	default
do	double	else	extends	false
final	finally	float	for	if
implements	import	instanceof	int	interface
long	native	new	null	package
private	protected	public	return	short
static	super	switch	synchronized	this
throw	throws	transient	true	try
void	volatile	while		

Keywords that are reserved but not used by Java

const	goto

Fig. 4.2 Java keywords.

Example: Average


```
1  // Fig. 4.7: Average1.java
2  // Class average program with counter-controlled repetition
3  import javax.swing.JOptionPane;
4
5  public class Average1 {
6     public static void main( String args[] )
7     {
8        int total,           // sum of grades
9            gradeCounter,   // number of grades entered
10           gradeValue,     // grade value
11           average;        // average of all grades
12       String grade;       // grade typed by user
13
14       // Initialization Phase
15       total = 0;          // clear total
16       gradeCounter = 1;   // prepare to loop
17
```

```
18          // Processing Phase
19          while ( gradeCounter <= 10 ) {  // loop 10 times
20
21             // prompt for input and read grade from user
22             grade = JOptionPane.showInputDialog(
23                         "Enter integer grade: " );
24
25             // convert grade from a String to an integer
26             gradeValue = Integer.parseInt( grade );
27
28             // add gradeValue to total
29             total = total + gradeValue;
30
31             // add 1 to gradeCounter
32             gradeCounter = gradeCounter + 1;
33          }
34
35          // Termination Phase
36          average = total / 10;  // perform integer division
37
38          // display average of exam grades
39          JOptionPane.showMessageDialog(
40             null, "Class average is " + average, "Class Average",
41             JOptionPane.INFORMATION_MESSAGE );
42
43           System.exit( 0 );       // terminate the program
44       }
45    }
```

Fig. 4.7 Class-average program with counter-controlled repetition (part 1 of 2).

Assignments and Expressions

Assignment operator	Sample expression	Explanation	Assigns
Assume: `int c = 3, d = 5, e = 4, f = 6, g = 12;`			
`+=`	`c += 7`	`c = c + 7`	10 to `c`
`-=`	`d -= 4`	`d = d - 4`	1 to `d`
`*=`	`e *= 5`	`e = e * 5`	20 to `e`
`/=`	`f /= 3`	`f = f / 3`	2 to `f`
`%=`	`g %= 9`	`g = g % 9`	3 to `g`

Fig. 4.12 Arithmetic assignment operators.

Increment/Decrement

Operator	Called	Sample expression	Explanation
`++`	preincrement	`++a`	Increment `a` by 1, then use the new value of `a` in the expression in which `a` resides.
`++`	postincrement	`a++`	Use the current value of `a` in the expression in which `a` resides, then increment `a` by 1.
`--`	predecrement	`--b`	Decrement `b` by 1, then use the new value of `b` in the expression in which `b` resides.
`--`	postdecrement	`b--`	Use the current value of `b` in the expression in which `b` resides, then decrement `b` by 1.

Fig. 4.13 The increment and decrement operators.

Java Operators

Operators	Associativity	Type
`()`	left to right	parentheses
`++ --`	right to left	unary postfix
`++ -- + - (`*type*`)`	right to left	unary
`* / %`	left to right	multiplicative
`+ -`	left to right	additive
`< <= > >=`	left to right	relational
`== !=`	left to right	equality
`?:`	right to left	conditional
`= += -= *= /= %=`	right to left	assignment

Fig. 4.15 Precedence and associativity of the operators discussed so far.

Java Operators

- Order is important in post/pre increment operators
- Increment operators (and combined operators like *=) implicitly are assignments; they must always have an LVALUE as a target
- One can still confuse “=“ with “==“

Primitive Data Types

Type	Size in bits	Values	Standard
`boolean`	8	`true` or `false`	
`char`	16	`'\u0000'` to `'\uFFFF'`	(ISO Unicode character set)
`byte`	8	–128 to +127	
`short`	16	–32,768 to +32,767	
`int`	32	–2,147,483,648 to +2,147,483,647	
`long`	64	–9,223,372,036,854,775,808 to +9,223,372,036,854,775,807	
`float`	32	–3.40292347E+38 to +3.40292347E+38	(IEEE 754 floating point)
`double`	64	–1.79769313486231570E+308 to +1.79769313486231570E+308	(IEEE 754 floating point)

Fig. 4.16 The Java primitive data types.

Java Data Types

- Java is a *strongly-typed* language
- Primitive data types are meant to be portable across all platforms
- Note that **char** is 16 bits in Java, not 8!

Java Methods

- Methods are functions/procedures attached to data via classes
- Methods are always invoked on an object, either *implicitly* or *explicitly*
- As with procedures, methods do not need to return a value (void is an option)
- As with functions, methods can return a value

The `Math` Class

- The Math class is an object library, and one must use a Math object to invoke methods:

```
System.out.println (Math.sqrt (16.0));
```

Method	Description	Example
`abs( x )`	absolute value of *x* (this method also has versions for `float`, `int` and `long` values)	if `x > 0` then `abs( x )` is `x` if `x = 0` then `abs( x )` is `0` if `x < 0` then `abs( x )` is `-x`
`ceil( x )`	rounds *x* to the smallest integer not less than *x*	`ceil( 9.2 )` is `10.0` `ceil( -9.8 )` is `-9.0`
`cos( x )`	trigonometric cosine of *x* (*x* in radians)	`cos( 0.0 )` is `1.0`
`exp( x )`	exponential method e^x	`exp( 1.0 )` is `2.71828` `exp( 2.0 )` is `7.38906`
`floor( x )`	rounds *x* to the largest integer not greater than *x*	`floor( 9.2 )` is `9.0` `floor( -9.8 )` is `-10.0`
`log( x )`	natural logarithm of *x* (base *e*)	`log( 2.718282 )` is `1.0` `log( 7.389056 )` is `2.0`
`max( x, y )`	larger value of *x* and *y* (this method also has versions for `float`, `int` and `long` values)	`max( 2.3, 12.7 )` is `12.7` `max( -2.3, -12.7 )` is `-2.3`
`min( x, y )`	smaller value of *x* and *y* (this method also has versions for `float`, `int` and `long` values)	`min( 2.3, 12.7 )` is `2.3` `min( -2.3, -12.7 )` is `-12.7`
`pow( x, y )`	*x* raised to power *y* (x^y)	`pow( 2.0, 7.0 )` is `128.0` `pow( 9.0, .5 )` is `3.0`
`sin( x )`	trigonometric sine of *x* (*x* in radians)	`sin( 0.0 )` is `0.0`
`sqrt( x )`	square root of *x*	`sqrt( 900.0 )` is `30.0` `sqrt( 9.0 )` is `3.0`
`tan( x )`	trigonometric tangent of *x* (*x* in radians)	`tan( 0.0 )` is `0.0`

Fig. 6.2 Commonly used `Math` class methods.

Example

```
class SquareRoot {

    public static void main(String[] args) {

        String outputString;
        double result;

        result = Math.sqrt(64.0);
        // result = mysquare(result);

        outputString = Double.toString(result);

        // outputString = "The result is " + outputString;

        System.out.println(outputString);

    }
```

Example (Continued)

```
public static double mysquare (double s) {
        return s*s;
    }
}
```

Method Definition

- Note user-defined method mysquare()
- Parameter passing: reference and value
- Type definitions and “coercion of arguments”
- Explicit cast necessary when there are no promotion rules

Defining Methods

- Note scope and purpose of method:
  - constructor
  - accessor
  - mutator
  - …
- Parameter passing vs. object-based data access
- Public/private method access and data hiding

Java's Allowed Promotions

Type	Allowed promotions
`double`	None
`float`	`double`
`long`	`float` or `double`
`int`	`long`, `float` or `double`
`char`	`int`, `long`, `float` or `double`
`short`	`int`, `long`, `float` or `double`
`byte`	`short`, `int`, `long`, `float` or `double`
`boolean`	None (`boolean` values are not considered to be numbers in Java)

Fig. 6.5 Allowed promotions for primitive data types.

The Java API Packages

- Java.awt: abstract windowing toolkit
- javax.swing: “Swing” GUI components
- java.util: date, time, random numbers, etc.
- java.sql: database connectivity
- java.io: streams, etc
- java.rmi: remote method invocation

Random Numbers

- Random number generation is important in most gaming/simulation situations
- java.util contains the random number package
- Typical call (returns value between 0 and 1)

```
Math.random();

value = 1+(int)(Math.random()*12)
```

Basic GUI Elements

- Textfield
- TextLabel
- Button

Categories of Variables

- Automatic:
  - variables come and go with context
  - values do not persist
- Static:
  - Variable is created with first instantiation
  - values persist until explicit programmer deconstruction

Variables

- Instance variables:
  - set up when object is instantiated
  - automatic duration unless tagged as “static”
- Local variables (block vars, method vars, formal parameters):
  - automatic duration related to block/context execution

Scoping

- Class scope: seen throughout a class/object
- Block scope: seen only within a block (and from nested blocks)

A. Instance variables and class methods: class scope

B. Local vars, block vars, formal parameters: block scope

Parameter Passing

- Primitive data types are ALWAYS passed by **value**
- Objects are ALWAYS passed by **reference**

Value: copy is made
Reference: Pointer to original location is passed

Method Signatures/Overloading

- Signature is data type of method parameter list
- Method names can be same: this is overloading
- Signature which matches invocation is method that is selected at run time for execution