Data Types, Operators, Statements and Data Input

2 – 4 pm Thursday 6/26/2008 @JD2211

1

Agenda

- Primitive data
- Operators
- Statements
- Read data from users

2

Java Numeric Primitive Types

Type	Storage	Min Value	Max Value
`byte`	8 bits	-128	127
`short`	16 bits	-32,768	32767
`int`	32 bits	-2,127,483,648	2,147,483,647
`long`	64 bits	-9,223,372,036,854,775,808	9,223,372,036,854,775,807
`float`	32 bits	1.40129846432481707e-45	3.40282346638528860e+38
`double`	64 bits	4.94065645841246544e-324	1.79769313486231570e+308

3

Character

- Expressed with single quotes
- ASCII (American Standard Code for Information Interchange) charter set
  - 7 bits/character → 128 characters
- Extended ASCII
  - 8 bits/character → 256 characters
- Unicode character set
  - 16 bits/character → 65,535 unique characters

4

Escape Sequences

Escape Sequence	Meaning
\b	Backspace
\t	Tab
\n	Newline
\r	Carriage return
\”	Double quote
\’	Single quote
\\	backslash

An escape sequence can be used to represent a character that would otherwise cause compilation errors.

5

Booleans

- Defined in Java using the reserved word `boolean`
- Two values
  - `true`
  - `false`

6

String

- (Character) String
  - Class
    - http://java.sun.com/j2se/1.3/docs/api/java/lang/String.html
  - Delimited by double quotations
  - String Concatenation
    - String + String
    - String + numbers

7

Default Values

Data Type	Default Value
byte	0
short	0
int	0
long	0L
float	0.0f
double	0.0d
char	'\u0000'
String (or any object)	null
boolean	false

8

Data Conversion

- Widening conversion is O.K.
- Narrowing conversion is NOT O.K.

From	To
byte	short, int, long, float, or double
short	int, long, float, or double
char	int, long, float, or double
int	long, float, or double
long	float or double
float	double

9

Conversion Techniques

- Assignment
- Promotion
- Casting

10

Operators

- Arithmetic operators
  - Addition
  - Subtraction
  - Multiplication
  - Division
  - Remainder

11

Increment and Decrement Operators

- Increment operator ++
- Decrement operator --
- count ++ (postfix)
- ++count (prefix)
- count = count + 1;

12

Assigned Operators

- Total += 5;
- Total = total + 5;
- Total += (sum – 12) / count;
- -=, *=, /=, %=

13

Selected Java Operator Precedence

Precedence level	Operator	Operation	Associates
1	+	unary plus	R to L
	-	unary minus	
2	*	multiplication	L to R
	/	division	
	%	remainder	
3	+	addition	L to R
	-	subtraction	
	+	string concatenation	
4	=	assignment	R to L

Precedence can be forced in an expression by using parentheses.

14

Complete Java Operator Precedence Table

- http://www.cs.uwf.edu/~eelsheik/cop2253/resources/op_precedence.html

15

Flow of Control

- Program consists of a set of statement
- Unless otherwise specified, the execution of a program proceeds in a linear fashion
- Invoking a method alters the flow of control

16

Flow of Control

- Within a given method, we can alter the flow of control through the code by using certain types of program statements
- Conditional (selection) statement
- Loop (repetition) statement

17

Boolean Expressions

- All conditional and loop statements are based on boolean expressions
- Boolean expressions
  - Equality operators
  - Relational operators
  - Logical operators

18

Equality and Relational Operators

Operator	Meaning
==	Equal to
!=	Not equal to
<	Less than
<=	Less than or equal to
>	Greater than
>=	Greater than or equal to

19

Logical Operators

Operator	Description	Example	Result
!	Logical NOT	! a	True if a is false and false if a is true
&&	Logical AND	a && b	True if a and b are both true and false otherwise
||	Logical OR	a || b	True if a or b or both are true and false otherwise

20

Truth Table

a	b	a && b	a || b
false	false	false	false
false	true	false	true
true	false	false	true
true	true	true	true

21

if Statement

- if
- if – else
- Using block statements

22

Conditional Operator

- ? :
- Similar to an if-else statement.
- It is a ternary operator

```
PassFail = (score >= 80)?  " -- PASS" :  " -- FAIL";
```

23

Nested `if`

- The statement executed as the result of an if statement could be another if statement

```
if (college_code == 'E')
if (dept_code == 'C')
System.out.println ("Computer Science");
else
System.out.println ("Non Computer Science");
```

```
 if (college_code == 'E')
        if (dept_code == 'C')
                System.out.println ("Computer Science");
        else
                System.out.println ("Non Computer Science");
```

24

The `switch` Statement

- Causes the executing program to follow one of several paths based on a single value.

```
switch (idChar)
{
       case 'A':
              aCount += 1;
              break;
       case 'B':
              bCount += 1;
              break;
       case 'C':
              cCount +=1;
              break;
       default:
              System.out.println("Error in
Identification Character.");
}
```

25

The `while` Statement

- a loop statement allows us to execute statements multiple times
- a white statement evaluates a boolean condition and executes the body of the loop if condition is true.

26

The `do` Statement

- The `do` statement is similar to the while statement except that its termination condition is at the end of the loop body

```
int count = 0;
do
{
      count++;
      System.out.println (count);
}  while (count < 5);
```

27

The `for` Statement

- The while and do statements are good to use when you don't initially know how many times you want to execute the loop body
- The for statement is well suited for executing the body of a loop a specific number of times that can be determined before the loop is executed

```
for (int count = 1; count <= 5; count ++)
     System.out.println (count);
```

28

Loop Controls

- `break`
- `continue`

29

Read Input from the User

```
import java.io.*;

public class ReadInputScore
{
      public static void main (String args[]) throws
IOException
      {
            int score;
            String ScoreInput;

            BufferedReader ReadScore = new
BufferedReader (new InputStreamReader(System.in));
            ScoreInput = ReadScore.readLine();
            score = Integer.parseInt(ScoreInput);
      }
}
```

30

Lab

- Write a program to determine a grade given a score that comes from the user. If the score is between 100 and 0, the program prints the grade accordingly. Otherwise, it prints "invalid score". Below is the criteria of the grade. The program runs until the input score is -1.

Score	Grade
100	A+
>= 90 && < 100	A
>= 80 && < 90	B
>= 70 && < 80	C
>=60 && < 70	D
< 60	F

31