CS202 Java Object Oriented Programming
Review of Language Basics

Chengyu Sun
University of California, Santa Barbara

Overview

- Programming environments
- Basic program structure
- Variables and types
- Operators
- Methods and recursion
- Arrays

Netbeans

- www.netbeans.org
- File system and directories
- Welcome.java
- Options
- Projects

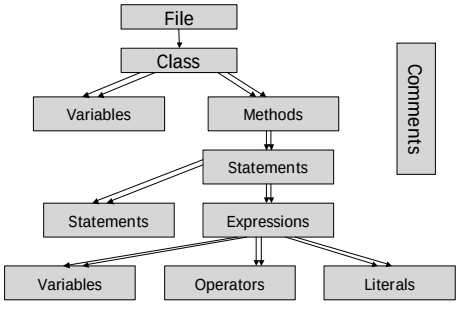
JDK

- java.sun.com
- `javac` Welcome.java
- `java` Welcome

Example: Grades.java

- Input
  - A set of grades
- Output
  - Highest grade
  - Lowest grade
  - Average grade

Basic Program Structure



Code Conventions

http://java.sun.com/docs/codeconv/html/CodeConvTOC.doc.html

- Required
  - Naming conventions (2.1, 9)
  - Comments (5)
    - Information not readily available in code itself
  - Indentation of if-else (7.4)
- Recommended
  - Line length (4.1, 4.2)
  - Programming practice (10)
  - Statements (7)

Comments

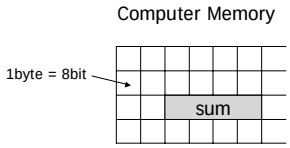
- Description of certain program functions
- Ignored by Java compiler
- Can appear anywhere of the program

```
/* a comment */          // another comment

/* a                     /*
  multiple-line           * a better looking
  comment                 * multiple-line comment
*/                        */
```

Variables

- Name
- Type
- Value




```
// declaration
int sum;

// assignment
sum = 0
```

// declaration and assignment
int sum=0;

Types

- Class types
- Primitive types
  - boolean – true or false
  - char – ‘a’, ‘b’, ‘c’, ..., ‘A’, ‘B’, ‘C’, ...
  - short – integers between -2^{15} and $2^{15}-1$
  - int – integers between -2^{31} and $2^{31}-1$
  - float – single precision real number
  - double – double precision real number

Coercion

- Implicit type conversion
- Also called type promotion
- No loss of precision
  - `char` → `int`
  - `int` → `double`
  - ...
  - Full list on p228, [D&D 5e]

Cast

- Explicit type conversion
- *Possible* loss of precision
- Syntax: （*Type*） *Expression*

```
double    number = 3.6;

int    integer_part = (int) number; // cast
double    fraction_part = number – integer_part;
```

Values

- boolean: true, false
- char: 'a', 'b', ... , 'A', 'B', ..., '1', '2', ...
- float, double: -0.1, 99.99, 1.1e13
- short, int: -10, 203, 0x11, 011 ...

Number Systems

- Base-2 (Binary)
  - 0, 1
- Base-8 (Octal)
  - 0, 1, ... , 7
- Base-10 (Decimal)
  - 0, 1, ... , 9
- Base-16 (Hexadecimal)
  - 0, ..., 9, A, B, C, D, E

1	1	0	1
2^3	2^2	2^1	2^0

Bin: $1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$

1	1	0	1
16^3	16^2	16^1	16^0

Hex: $1 \times 16^3 + 1 \times 16^2 + 0 \times 16^1 + 1 \times 16^0$

Operators

- Arithmetic
  - +
  - -
  - *
  - /
  - %
- Assignment
  - =
  - +=, -=, *=, /=, %=
- Increment and decrement
  - ++
  - --

More Operators

- Relational
  - ==, !=
  - >, <
  - >=, <=
- Conditional
  - ?:
- Logical
  - Negation: !
  - AND: &&
  - OR: ||

Precedence

- Determines the evaluation order of different types of operators
- Or, *parenthesis* to the rescue
- Exercise: check out the operator precedence table in the textbook (Appendix A)

Increment/decrement
Arithmetic
Logical
Assignment

```
a + b * c – d
a + b * c – d++
a + b * c - ++d
```

```
!a && b || c && d && a > d
!a && (b || c) && d && (a > d)
```

Associativity

- Determine the evaluation order of the operators with the same precedence
- Left-associative
  - Most operators are left-associate
  - E.g. *a + b + c*
- Right-associative
  - E.g. ??

Control Statements

- Branch
  - if
  - if ... else
- Switch
  - switch
- Loop
  - while
  - do ... while
  - for
- Break and continue
  - break
  - continue

Method

- Header
  - Access modifier
  - Return type
  - Name
  - Parameter list
- Body

Recursion

- A method calls itself

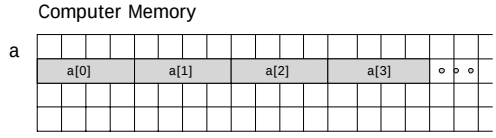
```
void print( int n )
{
   if( n <= 0 ) System.out.println();
   else
   {
      System.out.print("a");
      print(n-1);
   }
}
```

Ending Condition

- When the recursion should stop
- To avoid infinite recursion, make sure the ending condition
  - Exists
  - Reachable
  - Comes before the recursive call

Arrays

- Name
- Type
- Length (or Size) – number of elements in the array
- Values



Access Array Elements

- arrayname.length
- Index is from 0 to (arrayname.length-1)

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

a[5] = 3;    // assign 3 to the 6th element

// prints out all elements
for( int i=0 ; i < a.length ; ++i )
  System.out.println( a[i] );
```

index

Array as Parameter

- Write a method sumArray() which returns the sum of the elements in a given array

```
int sumArray( ?? )
{
    int sum = 0;

    ??

    return sum;
}
```

Array as Return Type

- Write a method createArray() which returns an integer array of given size n.

```
?? createArray( int n )
{
    return ??;
}
```

Working with More Than One Classes

- ConsoleReader.java
- ConsoleReaderTest.java