Overview on Java Programming

CSCI4180 tutorial 1

Qin Chuan

Outline

- Java program structure
- Java language basics
  - Primitive types
  - Operators
  - Object references
  - String
  - Control statements
  - File and I/O
  - Array
- OOP

Java Features

- High-level language
  - Simple syntax
- Purely object-oriented language
- Cross platform
- Varieties of libraries

Programming Environment

- Download JDK
  - http://www.oracle.com/technetwork/java/javase/downloads/index.html
- Edit, compile and run
  - IDE: eclipse http://www.eclipse.org/downloads/
  - Command line
    - javac FileName.java //get FileName.class
    - java FileName
    - -help for help

Project Structure: Single Source File

- Welcome.java

```
public class Welcome {

   /*
   * comments here
   */

     public static void main (String [ ] args) {
         System.out.println("Welcome to Java!");
     }
 }
```

Project Structure: Multiple Source Files

- PackageOne
  - ClassOne.java
  - ClassTwo.java
- PackageTwo
  - ClassThree.java
  - ClassFour.java
- Usually each file defines one class

Source File Structure

```
import <packagename>.<someclassname>;
import <packagename>.*;
```

```
class <ClassName>
{
    //field variables declaration

    <type>  <fieldName1>;
    <type>  <fieldName2> = <initial value>;


    <return_type>  <methodName1> ( arguments)
    { ...
    }
    <return_type>  <methodName2> (arguments)
    { ...
    }
}
```

Outline

- Java program structure
- Java language basics
  - Primitive types
  - Operators
  - Object references
  - String
  - Control statements
  - File and I/O
  - Exception
  - Array
- OOP

Integral Literals

- Integer literals are int by default
  - 4 bytes
  - int var = 100;
- byte
  - 1 byte
  - byte var = 100;
- short
  - 2 bytes
  - short var = 100;
- long
  - 8 bytes
  - long var = 100;
  - long var = 100L;
  - long var = 100l;

Floating-point Number

- Floating-point number literals are considered to be of type double by default
  - 8 bytes
  - double var = 3.14;
  - double var = 1e8;
- float
  - 4 bytes
  - float var = 3.14; //not ok
  - float var = (float) 3.14; //explicit type conversion
  - float var = 3.14F;
  - float var = 3.14f;

char

- chars are written in program with single quotes
  - char charVar = ‘a’;
- Each character is represented by using Unicode in Java
  - char charVar = (char) 65; // ‘A’
- char can be added, subtracted
  - Operation is based on Unicode
  - int diff = ‘A’ – ‘B’;

boolean

- A boolean value represents a true or false condition
  - boolean var = true;
- The reseved keywords true and false are the only valid values for a boolean type

Outline

- Java program structure
- Java language basics
  - Primitive types
  - Operators
  - Object references
  - String
  - Control statements
  - File and I/O
  - Array
- OOP

Operators

- Arithmetical operators
  - + - * / %
- Relational operators
  - < <= > >= == !=
- Logical operators
  - ! & | && || ^
- Conditional operator
  - bool_expression ? true_case : false_case
- Assignment operator
  - =
- Short-hand operators
  - ++ -- += -= *= etc.

Outline

- Java program structure
- Java language basics
  - Primitive types
  - Operators
  - Object references
  - String
  - Control statements
  - File and I/O
  - Array
- OOP

Creating Objects

- A variable holds either a primitive value or a reference to an object
- An object reference variable holds the address of an object
- Instantiation
  - Use the new operator
  - String text = new String(“abc”);
  - An object is an instance of a particular class

String

- Store and manipulate a sequence of characters
- Create String objects
  - String text = new String(“abc”);
  - String text = “abc”;
    - This short-hand is dedicated to class String only
- Empty String and null String object
  - String text = “”;
  - String text; // no assignment means null
- String comparison
  - str1 == str2 //object comparison
  - str1.equals(str2) //content comparison

Methods

- Common used methods in class String
  - char charAt(int index)
  - int length()
  - String[] split(String regex)
  - …
- Refer to
  - http://docs.oracle.com/javase/6/docs/api/java/lang/String.html

Outline

- Java program structure
- Java language basics
  - Primitive types
  - Operators
  - Object references
  - String
  - Control statements
  - File and I/O
  - Array
- OOP

Branching

- if and if/else

```
if (boolean_expression)
{
    true_statements;
}
```

```
if (boolean_expression)
{
    true_statements;
}
else
{
    false_statements;
}
```

- Nested if-statements
  - An else-part attaches to the nearest available if, which has not already been matched

Branching

- switch statement

```
switch (var){
   case value1: statements;
         break;      //break is optional
   case value2: statements;
         break;
   default: statements;
}
```

- Switch knob (var) must be primitive data type
  - Usually be integers and characters
- Case labels must be constant
  - Cannot be ranges

Repetition

- for loop
  - starting value, ending condition and a loop counter

```
for (start; boolean_expression; update)
{
  body_statement(s);
}
```

- Executing order
  - 1. Execute the start part
  - 2. Check the ending condition
  - 3. Execute the loop body statements
  - 4. Counter update
  - 5. Go to 2
- Nested for loop

Repetition

- while loop

```
while (boolean_expression)
{   statement(s); }
```

  - Check the condition
    - true: execute the statements and check again
    - false: quit
- do-while loop

```
do
{
  statement(s);
} while (boolean_expression)
```

  - The statement is executed at least once

Repetition: break and continue

- continue: finish the current iteration and loop again
- break: finish the current loop

```
int i = 1;
while (i <= 8) {
    if (i == 4)
      break;
    System.out.print(i);
    i++;
}
```

Output: 123

```
int i = 1;
while (i <= 8) {
    if (i == 4)
      continue;
    System.out.print(i);
    i++;
}
```

Output: 1235678

Outline

- Java program structure
- Java language basics
  - Primitive types
  - Operators
  - Object references
  - String
  - Control statements
  - File and I/O
  - Array
- OOP

File Input/Output Operations

- Keep data non-volatile
  - Store/write data in a file in hard disk
  - Read data from a file
- Java provides methods for I/O operations
  - Class Scanner: input
  - Class PrintStream: output

Class Scanner

- The source for the Scanner object could be the keyboard, a file, a web source
  - Scanner input = new Scanner(System.in);
  - Scanner input = new Scanner(new File(“filename”));
  - Scanner input = new Scanner(newURL(“http://...”). openStream( ));

Scanner Object Usage

- Token-by-token
  - The methods hasNextInt(), hasNextDouble(), hasNext*Type*() … return us a boolean value that indicates if there is more data of the indicated type to read
  - The methods nextInt(), nextDouble(), … reads a piece of data (a token) from the source
- Line-by-line
  - The methods hasNextLine()returns a boolean value that indicates if there is one more line to read
  - The method nextLine() reads a line and returns a String
- Operations may fail, need to handle exceptions
- Refer to http://docs.oracle.com/javase/6/docs/api/index.html?java/util/Scanner.html

Class PrintStream

- Write data to a file
  - PrintStream output = new PrintStream(“out.txt”);
  - output.println(“hello”);
- System.out is a PrintStream object
  - Write data to console screen
  - System.out.println(“hello”);
- Refer to http://docs.oracle.com/javase/6/docs/api/index.html?java/io/PrintStream.html

Redirect I/O

- System.out is an object of PrintStream, and it is used to send data to console screen
  - Redirect the data to other places

```
import java.io.*;
class Redirect {
  public static void main(String[] args) throws IOException
  {
    PrintStream newPlace = new PrintStream(“out.txt");
    System.setOut(newPlace);

    System.out.println("Hello World");
    // System.out refers to the new PrintStream object!
  }
}
```

- System.setErr(…), System.setIn(…)
  - Refer to http://docs.oracle.com/javase/6/docs/api/index.html?java/lang/System.html

Outline

- Java program structure
- Java language basics
  - Primitive types
  - Operators
  - Object references
  - String
  - Control statements
  - File and I/O
  - Array (be talked in next week)
- OOP (be talked in next week)

References

- Java language specifications
  - http://java.sun.com/docs/books/jls/
- Java tutorial
  - http://docs.oracle.com/javase/tutorial/index.html
- Java API
  - http://docs.oracle.com/javase/6/docs/api/

Thanks!