2. Java language basics (1)

Minhaeng Lee
Contents

• Features
• Create your first java program
• Packages
• Type
  – Primitive Types
  – Useful Class Types
  – Casting
  – Conversion (Integer/Float/Double to String, vise versa)
Feature of Java

• Java Language
  – Its own syntax rule, structures
  – **Object-Oriented** Programming concepts (OOP)
  – Much like C++ language
  – Code block modularized into method (= function)
  – Delimited by braces { }
  – Statement ends with semi-colon ;
Feature of Java

• The Java Compiler
  – Source code .java
  – Compile ($javac test.java)
  – Byte code .class targeting for JVM
  – Running ($java test)
  – $ means command prompt (e.g. c:\Users\Name> )

• JVM
  – Heart of java for "write-once, run-anywhere”
  – We can run java code anywhere if JVM is available
Java Virtual Machine

1. Java Code (.java)
2. JAVAC compiler
3. Byte Code (.class)
4. JVM
   - Windows
   - Linux
   - Mac

Using a single source

Working on different platform

Features of Java

• **Garbage Collector**
  – Memory management unit
  – Automatically prevent **memory leak**
  – Release memory automatically
  – *c.f. C/C++* language, user have to explicitly release it

• **Java Development Kit (JDK)**
  – Including compiler and additional libraries

• **Java Runtime Environment (JRE)**
  – Including JVM and libraries for running
JDK, JRE, and JVM

**JDK**
- javac, jar, debugging tools, javap

**JRE**
- java, javaw, libraries, rt.jar

**JVM**
- Just In Time Compiler (JIT)

http://www.javabeat.net/what-is-the-difference-between-jrejvm-and-jdk/
Create your first java program using console/terminal

• Most basic way of running java
  – Sometimes, we cannot use editing tools like Eclipse (e.g. pure linux environment)

• Check whether you can run java on your terminal/console
  – Windows : start – execution – cmd
  – OSX : spotlight search – terminal
  – Type “java” and enter to check whether it is installed or not
Create your first java program

Check whether you have java or not

dhcp-v220-167:~ mac$ java
Usage: java [-options] class [args...]
(to execute a class)
or java [-options] -jar jarfile [args...]
(to execute a jar file)
where options include:
  -d32    use a 32-bit data model if available
  -d64    use a 64-bit data model if available
  -server to select the "server" VM
  The default VM is server,
because you are running on a server-class machine.
  
  -cp <class search path of directories and zip/jar files>
  -classpath <class search path of directories and zip/jar files>
  A: separated list of directories, JAR archives,
  and ZIP archives to search for class files.
  -D<name>=<value>
  set a system property
  verbose:[class|gc|jni]

If you can see this kinds of output,
then we can use java in console environment!
Type! test.java file

class test{
    public static void main(String[] args){
        System.out.println("Hello world");
    }
}

- Display folder contents
  - Mac OSX: ls
  - Windows: dir
- Directory change (common)
  - cd <target folder>
- Go parent directory (common)
  - cd ..

- Editing tools
  - Mac OSX
    - $vi
    - “i” for insert
    - Save: “Esc” - “:wq <filename to save>”
  - Windows
    - notepad <filename>
Run!

dhcp-v220-167:~ mac$ javac test.java
dhcp-v220-167:~ mac$ java test
Hello world

C:\Users\mhlee>javac test.java
C:\Users\mhlee>java test
hello world

- Filename should be similar to class name
- One class per one file
- Main function has same header
- Reference: http://introcs.cs.princeton.edu/java/11hello/
Trouble Shooting

• Javac is unrecognized (Windows)?
  – Need to add path to use command in console
  – Find javac path (something like “C:\Program Files\Java\jdk<YOUR_JAVA_VERSION>\bin”)
  – Start – computer – (right click) – properties – Advanced system setting – Environment Variables
  – Edit System Variable “Path”
  – Delimiter is “;”
Programming using Eclipse

- Workspace directory setting
- Cannot open more than one Eclipses for a single path

- Beginning screen
- Just click “workbench”
Programming using Eclipse
Create project

- Rightclick in Project Explorer space
- New - Project
Programming using Eclipse
Select project type

- We are going to use Java Project
Programming using Eclipse

- Set project Name as you wish
- Just using default JRE
- Sometimes, syntax differs based on JRE version
Programming using Eclipse

- Set Output Folder
- Folder for Class file (binary)
- We just use default location
Programming using Eclipse
create a new class file

- Right click on "src"
- New - Class
Programming using Eclipse

• Set name of class as you wish
Programming using Eclipse

```java
public class HelloWorld {
}
```
Programming using Eclipse
Type one more time!

• In Eclipse, save involves compile that generate binary code (.class file)
• * near Editor tab means “mofieid”
Programming using Eclipse
Run it!
Programming using Eclipse
other ways of running

- Rightclick on java file
- Run As – Java Application

- Bottom arrow right side of run icon
- Run As – Java Application
More tips: show line numbers

- Windows – preferences – General – Editors – Text Editors
- Check show line number
- Easy to follow up
- Java error messages involves line number
Example error
Packages

• Located at the top of each source file
• Sort of source code group
• Make it well organized (e.g. like folders)
• Don’t force you to use packages

package orgType.orgName.appName.compName;

• orgType is the organization type such as com, org, or net.
• orgName is the name of the organization's domain, such as makotogroup, sun, or ibm.
• appName is the name of the application, abbreviated.
• compName is the name of the component.
Comments

• // : comment a single line
  – Eclipse short cut – Ctrl + /
• /* : comment begin
• */ : comment ending

• // this line commented
• /* All of this par is
  commented.. */
Binary Number

- Internal data storage is binary number
- Important to understand data overflow and memory management
Binary Number (practice)

• Convert from Binary to Decimal
  – 101
  – 1101
  – 1000001
  – 1111

• Convert from Decimal to Binary
  – 10
  – 100
  – 128
  – 1024
Note for newbies

• Each java file has at least one class
  – For now, consider class as a template

• Everything should be declared before we use
  – Some of them are pre-declared by java
    • Primitive types
    • Internal/External Libraries
  – Other must be declared by ourselves.
Variable Declaration

• Every variables **MUST BE** declared before use them
  • Anywhere before we use

• Syntax
  – `<TYPE> <NAME>;`
  – `<TYPE> <NAME> = <INITIAL VALUE>;`
  – You can use any `<NAME>` for the variable.
  – `<NAME>` Must begin with characters (by syntax)
  – Camel Case (Suggested for readability)
    • Lower case at the beginning, Upper case at the beginning of each word
    • E.g. `hiMyNameIsCamelCase`

• Example
  – `int myIntVar;`
    • Integer type variable called myIntVar
  – `double myDoubleVar;`
    • Double type variable called myDoubleVar
  – `short myVar = 10;`
    • Also possible declaration with initialization
Variable Value Assignment

- Right to Left
- Using “=“ operator called assignment operator

```c
int myVar1;
double myDouble;

//Usually used
myVar1 = 10;
myDouble = myVar1 + 10;
//Not very good because confusing
myDouble = myVar1 = 10;
```
Primitive Types

• Most basic type defined in Java
• Begins with lower case
• Quite common name comparing to other languages

• `int / short`
  – Contain integer (4byte, 2 byte)

• `float / double`
  – Contains real number (4, 8 byte respectively)
  – float myFloat = 0.3f;
  – double myDouble = 0.3;

• `boolean` (1 byte)
  – Contains true or false.
  – E.g. boolean isNice = true;

• `char` (2 byte)
  – One character
  – Using single quotation “"
Casting

• Why?
  – Each data has its best fit datatype
  – For example,
    • PI should be saved as.. (integer/double) ?
    • Student numbers should be saved as (integer/double) ?
  – Sometimes need to be changed
  – Just call a variable as a different type with same data

• Data transfer between different types
  – E.g. from integer to float / double
  – Use “(<type>)” before source variable
    • E.g. double target = (double) iAmInteger;
  – Be careful when bigger one to smaller one
    • Data loss expected
      – Float (3.4) to Integer (3)
    • E.g. casting from double to short
Casting (practice)

- int a; float b; double c; boolean d;

- int to double: `(double) a`
- double to float: `(float) c`
- float to int: `(int) b`
  - It may loss data. Why?
- int to boolean: `(boolean) a` (not possible)
Useful Class Types

• **String**
  – Use double quotation “” - **Please memorize**
  – Can store sequence of characters
  – Concatenation using +
    • LONG = “small” + “small”; (LONG = “smallsmall”)

• **From other types to String**
  – Use toString
  – Every class have
  – Every class can be printed as String type
Operators

• Operators may have different roles for different data types.
• We can define operators for our own type
• + : PLUS - Add value for number type value
  – Concatenation for <String> - Please Memorize
• - : MINUS - Subtract value for number type value
• * : STAR - Multiply for number type values
• / : SLASH - Divide for number type values
• % : PERCENT – Get Remaining Value (e.g. 10%3 = 1)
Conversion

• Inbuilt explicit method in Java
  – Integer to String
  – String “1000” to Integer 1000

• From String to number
  – `<Integer/Float/Double>.parse<Int/Float/Double>();`
  – e.g. `double d = Double.parseDouble();`

• From number to String
  – Use `toString()` that every class have
Conversion (practice)

- int to String
- float to String
- double to String
- String to int
- String to float
- String to double
- Integer to int
- Float to float
- Double to double
Math Functions

• Math functions
  – `Math.<something>()`;

• Examples
  – `Math.sqrt(var);`
  – `Math.round(var);`
  – `Math.ceil(var);`
  – `Math.pow(var, var);`
  – Etc.
System.out.println()

- Please Memorize how to use it.
- Standard output of Java
- I’ve used that more than 10k times.
- new line automatically involved
- `System.out.println("Anything you want!");`
- `int age = 10;`
- `System.out.println("I am " + age + " years old");`
  - What is the meaning of “+” in here?
String[] args

- **Argument** string array ([]) (name can be changed)
- Parameter (or argument) set in main method
- Useful to change parameters without compiling

- **`$Java test a b c`**
  - args[0] : “a”
Practice! Let’s make root equation!

\[ ax^2 + bx + c = 0. \]

\[ x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} \]

- We have three parameters come from “args”

```java
public static void main(String[] args) {
    double root1 = 0; //known
    double root2 = 0; //known

    String a = args[0];
    String b = args[1];
    String c = args[2];

    //... fill out

    System.out.println("First root is "+root1);
    System.out.println("Second root is "+root2);
}
```
References

• IBM