2. Java language basics: 

*Condition, 1D array, and loop*

Minhaeng Lee
Contents

• Casting and conversion

• Conditional Statement (if)
• Loop
• Switch
• Practice (Multiplication Table!)
• Homework
Casting

• Data transfer between two different types
  – E.g. from integer to float / double
  – Use ‘(<type>)’ before source variable variable
    • E.g. \texttt{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
      – 3.141592 (double) \rightarrow 3 (int)
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
• int to boolean : (boolean) a  (not possible)
Useful Class Types

• String
  – Use double quotation “”
  – 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
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(stringValue);`

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

- int to String: `Integer.toString(value)`
- float to String: `Float.toString(value)`
- double to String: `Double.toString(value)`
- String to int: `Integer.parseInt(stringValue)`
- String to float: `Float.parseFloat(floatValue)`
- String to double: `Double.parseDouble(doubleValue)`

- Integer to int: `(int)intValue`
- Float to float: `(float)floatValue`
- Double to double: `(double)doubleValue`
Math Notation

• Operator
  – + : plus
  – - : minus
  – * : multiplication
  – / : division
  – % : remaining values  (4%3 : 1, 10%2 : 0)

• Math functions
  – Math.<something>();
  – Math.sqrt(var);
  – Math.round(var);
  – Math.ceil(var);
  – Math.floor(var);
  – Math.pow(var, var);
  – Etc.
Short notation

• Used in C, C++, Java

• `++`, `--`
  
  – `a++ : a = a + 1`
  – `a-- : a = a - 1`

• `+=`, `*=`, `/=`
  
  – `a+=1 : a = a + 1`
  – `A*=10 : A = A * 10`
  – `...`
public class practice1 {
  public static void main(String[] args) {

    //Instruction.
    //Print out root value of input string value
    //Convert stringFloat into double,
    //and then use math function to get square root.
    double rootValue = 0.0;
    String stringFloat = "3.125";

    //Fill out this part..

    System.out.println("Root of input string is "+rootValue);
  }
}

Boolean type

- Should be *true* or *false*
- 0 = false, otherwise = true but specific boolean result is suggested
- Can compare only two compatible variables
# Boolean Operators

<table>
<thead>
<tr>
<th>Operator</th>
<th>Usage</th>
<th>Returns true if ...</th>
</tr>
</thead>
<tbody>
<tr>
<td>&gt;</td>
<td>a &gt; b</td>
<td>a is greater than b</td>
</tr>
<tr>
<td>&gt;=</td>
<td>a &gt;= b</td>
<td>a is greater than or equal to b</td>
</tr>
<tr>
<td>&lt;</td>
<td>a &lt; b</td>
<td>a is less than b</td>
</tr>
<tr>
<td>&lt;=</td>
<td>a &lt;= b</td>
<td>a is less than or equal to b</td>
</tr>
<tr>
<td>==</td>
<td>a == b</td>
<td>a is equal to b</td>
</tr>
<tr>
<td>!=</td>
<td>a != b</td>
<td>a is not equal to b</td>
</tr>
<tr>
<td>&amp;&amp;</td>
<td>a &amp;&amp; b</td>
<td>a and b are both true, conditionally evaluates b (if a is false, b is not evaluated)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>!</td>
<td>!a</td>
<td>a is false</td>
</tr>
<tr>
<td>&amp;</td>
<td>a &amp; b</td>
<td>a and b are both true, always evaluates b</td>
</tr>
<tr>
<td></td>
<td></td>
<td>a</td>
</tr>
<tr>
<td>^</td>
<td>a ^ b</td>
<td>a and b are different</td>
</tr>
<tr>
<td>Expression</td>
<td>Value</td>
<td>Comment</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>----------</td>
<td>-------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>0 &lt; 200 &amp;&amp; 200 &lt; 100</code></td>
<td><code>false</code></td>
<td>Only the first condition is true.</td>
</tr>
<tr>
<td>`0 &lt; 200</td>
<td></td>
<td>200 &lt; 100`</td>
</tr>
<tr>
<td>`0 &lt; 200</td>
<td></td>
<td>100 &lt; 200`</td>
</tr>
<tr>
<td><code>0 &lt; 100 &lt; 200</code></td>
<td>Syntax error</td>
<td><strong>Error:</strong> The expression <code>0 &lt; 100</code> is true, which cannot be compared against 200.</td>
</tr>
<tr>
<td>`0 &lt; x</td>
<td></td>
<td>x &lt; 100`</td>
</tr>
<tr>
<td>`0 &lt; x &amp;&amp; x &lt; 100</td>
<td></td>
<td>x == -1`</td>
</tr>
<tr>
<td><code>!(0 &lt; 200)</code></td>
<td><code>false</code></td>
<td><code>0 &lt; 200</code> is true, therefore its negation is false.</td>
</tr>
<tr>
<td><code>frozen == true</code></td>
<td><code>frozen</code></td>
<td>There is no need to compare a Boolean variable with true.</td>
</tr>
<tr>
<td><code>frozen == false</code></td>
<td><code>!frozen</code></td>
<td>It is clearer to use <code>!</code> than to compare with <code>false</code>.</td>
</tr>
</tbody>
</table>
If, If else, and else

- **If**(<condition>) <statement>;
  - If given <condition> is true, then execute only one <statement>

- **if** (<condition>) { <statements> }
  - If given <condition> is true, then execute <statements>

- **if** (<condition>) { <statement1> } else { <statements2> }
  - If given <condition1> is true, then execute <statements1>, otherwise execute only <statements2>

- **if** (<condition1>) {<statements1>} else if(<conditions2>) {<statements2>} else{<statements3>}
  - If given <condition1> is true, then execute <statements1>, otherwise execute <statements2> if <condition2> is true, otherwise execute <statements3>
int age = 20;
if(age > 20){
    System.out.println("I can drink beer!!");
}else if(age > 5){
    System.out.println("I can drink soft drink!");
}else{
    System.out.println("I cannot drink anything but water");
}
If, if else, and else (practice!)

Practice 1

• Print grade from “A” ... to “F”
• From one single integer value (0 – 100)
• Use if, if else, and else.
• 90 – 100 : A
• 80 – 90 : B
• 70 – 80 : C
• 60 – 70 : D
• 0 – 60 : F

Practice 2

• People can drink beer when they are over 20
• People can drive when they are over 14
• Write a code that print what people can do.
• E.g. 15 -> “can drive but cannot drinking”, 21 ->”can drive and drinking”, 10 -> “cannot neither drive nor drinking”
• Use nested if
Array

• Store multiple data in a single variable
• [] indicates 1D-array
• `<Type>[] <variable name> = new <Type>[][<size>];`
• Need initialization
  – Use `new` keyword to indicate memory allocation
  – c.f. single variable doesn’t need to do it

• Example:
  – Single int : `int a;`
  – Multiple ints : `int[] a = new int[10];`
Array - access

• int array : int[] a = new int[10];
• String array : String[] strArray = new String[10];

• Access the first element
  – Index begins with 0
  – Examples
    • Store 10 to the first one : a[0] = 10;
    • Get second number and store : b = a[1];
    • Print the first one : System.out.println(a[0]);

• Useful property : length
  – Returns the length of array
  – Example : a.length
Loop

• **While**
  – While (<condition>) {<statement>}
  – Execute <statement> while <condition> is true
  – Sequence: <condition> - <statement> - <condition> - <statement> ...

• **Do-While**
  – Do { <statement> } While( <condition> )
  – Similar to While but execute <statement> at least one time.
  – Sequence: <statement> - <condition - <statement> - <condition> ...

• **For**
  – For ([init] ; <condition> ; [incremental]) {<statement>}
  – Sequence: [init] - <condition> - <statement> - [incremental] -
    <condition> - <statement> - [incremental] - ....

• Break needed
//Initialize
int a = 0;
while ( a < 10 /* Condition */){
    //Statement for execution
    System.out.println("hello, a :"+a);
    //condition changed
    a++; //Equal to a = a + 1;
}
int c = 10;
do {
    System.out.println("hello c:" + c);
} while (c < 10);
Loop: for

```java
for ( int b = 0; b < 10; b++){
    System.out.println("hello, b :"+b);
}
for ( int b = 0 /*init*/; 
    b < 10 /*condition*/ ; 
    b++ /*condition change*/){
    System.out.println("hello, b :"+b);
}
```
Let’s practice: Times table

• Times table for one number (ex. 2)
• Using one loop
• Expected output:
  – 2 x 1 = 2
  – 2 x 2 = 4
  – 2 x 3 = 6
  – ...

• Times table for multiple number (ex 2 – 5)
• Using nested loop
• Expected output:
  – 2 x 1 = 2, 3 x 1 = 3, 4 x 1 = 4, 5 x 1 = 5
  – 2 x 2 = 4, 3 x 2 = 6, 4 x 2 = 8, 5 x 2 = 10
  – ....
break and continue

• **Break**
  – Used in loop
  – Stop current loop

• **Continue**
  – Used in loop
  – Skip rest of statement and go next iteration
Example

```java
for (int a = 0; a < 10; a++){
    if (a < 3) continue;
    System.out.println("Hello! I am "+a);
    if (a > 6) break;
}
```

Hello! I am 3
Hello! I am 4
Hello! I am 5
Hello! I am 6
Hello! I am 7
Switch

• Similar to multiple if else statement
• Only possible when we use int variable and conditional value check whether it is equal
• Execute from certain case to the end

• Switch <variable> {
  – Case <case1>:
    • <statement1>;
    • break;
  – Case <case2>:
    • <statement2>;
    • ...
    • break;
  – Default:
    • <default statement>;
  }

If <variable> is equal to <case1> then execute <statement1>
Else
If <variable> is equal to <case2> then execute <statement2>
...
Else
Execute <default statement>
Switch

```java
switch (age) {
    case 25:
        System.out.println("you can drink beer and drive!");
    case 15:
        System.out.println("you can only drive!");
    default:
        System.out.println("I don't know..");
}
```

Exactly Same?

```java
if (age == 25){
    System.out.println("you can drink beer and drive!");
} else if (age == 15){
    System.out.println("you can only drive!");
} else{
    System.out.println("I don't know..");
}
```
Homework 1.

- Multi-line times table (2 – 9)
- Using triple nested loops
  - 2 to 5 for top row
  - 6 to 9 for lower row
- Omit similar number multiplication
- Using “if” and “else” on top of previous one
  - E.g. 1x1, 2x2, 3x3 ... so on.