4. Java language basics (4)  

Function  

July 19, 2015  

Minhaeng Lee
Review : loop

• Program print from 0 to 10
  – While/for
  – Two guys write down on the board!

• Make a condition that make true only if a given int variable ‘int a;’ is larger than 10 and divided by 3

• Break / Continue?
Tip: Naming rule in java programming

• Usually kept
• Reserved keyword begins with lower case
  – new, for, while, import, package ...
• Class begins with upper case
  – MyClass a;

• If a word begins with upper case letter?
  – We knows it is class!
• Otherwise : purple colored words in eclipse
  – Reserved keywords
Tip: I don’t know what function I have to use!

• I know what class I have to use but I cannot remember a function for that!

• Use “.” after class, or instance

• *Math.*
Function / Method

• Same meaning different wording
• If a Function is in class then we call it as method
• Syntax
  – `<return type> <name>( [parameters]) { [statement] }`

• Return type can be any type in java
  – Return value from function
  – `void` : when there is nothing to return
Function example: two number add

```java
public int add(int a, int b) {
    return a + b;
}
```
Function and variable scope

• Usually only variables declared in a function are used in the function.

• Sometimes, it also possible to access outside of the function

• When?
  – Only if the variable is loaded to memory.

• When is that time?
  – static
Referencing to outside variables

```java
int outVariable = 10;
public int add(int a, int b){
    return a+b + outVariable;
}

static int staticOutVariable = 10;
public static int staticAdd(int a, int b){
    return a+b+staticOutVariable;
}
```
Variable creation and termination

```java
public int minus(int a, int b){
    int tmpvariable = 10;
    int result = a - b;
    return result;
}
```
Memory stack and function call

```java
public static void main(String[] args) {
    int a = 5;
    int b = 10;
    int sum2 = staticAdd(a, b);
    System.out.println(sum2);
}

static int staticOutVariable = 10;
public static int staticAdd(int a, int b) {
    int insideSum = a + b;
    int outsideSum = insideSum + staticOutVariable;
    return outsideSum;
}
```
public static void main(String[] args) {
    int a = 10;
    int b = 15;
    noReturnAdd(a, b);
    System.out.println(a);  //Guess the result!
}

public static void noReturnAdd(int a, int b) {
    a = a + b;
}
Key input

• Use Scanner library
• Import first (where should we put in our code?)
  – `import java.util.Scanner;`

• Example – get integer from keyboard
  – `Scanner keyboard = new Scanner(System.in);`
  – `int myInput = keyboard.nextInt();`
  – `String myString = keyboard.nextLine();`

• For other type? Check functions using “.” after keyboard
  – Type: “`keyboard.`”
  – Check the function list
Loop input processing for interactive programming

Type it and see how it works

```java
public static void main(String[] args) {
    Scanner keyboard = new Scanner(System.in);
    int myInput = -1;
    while (myInput != 0) {
        System.out.print("Give me input!:");
        myInput = keyboard.nextInt();
        System.out.println("Current input!:"+myInput);
    }
}
```
Loop input processing for interactive programming

Type it and see how it works

```java
public static void main(String[] args) {
    Scanner keyboard = new Scanner(System.in);

    String myInputString = "";

    do{
        System.out.print("Give me input!:");
        myInputString = keyboard.nextLine();
        System.out.println("Current input!:" + myInputString);
    } while(myInputString.length()>0);
    System.out.println("End!");
}
```
Exercise : interactive calculator

• Use **static** keyword to call function from main function

• Make simple interactive calculator

• Scenario
  – 1. choose menu : 1-add, 2-minus, 3-multiple, 4-divide, 5 quit
    • Number input
  – 2. enter two numbers : two times of receiving as integer
  – 3. show results
  – 4. go back to 1
    • Using loop – while or for
Array

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

- Don’t have to do de-initialization
  - Why?

- Example:
  - Single int: `int a;`
  - Multiple int: `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 0
  – examples
    • store 10 to the first one : a[0] = 10;
    • print the first one : System.out.println(a[0]);

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

• Input `string = “10 20 30”;`
• I need three integer value 10, 20, and 30.
• How?
  – 1. divide input string into three strings
    • `stringVal.split(delimiter);`
    • Ex : `string.split(“ “);`
  – 2. convert each string into integer
    • `Integer.parseInt(stringValue);`
Exercise : interactive calculator v2

• Use `static` keyword to call function from main function

• Make simple interactive calculator

• Scenario
  – 1. choose menu : 1-add, 2-minus, 3-multiple, 4-divide, 5 quit
    • Number input
  – 2. enter two numbers from one string input
    • Need to parse from string
  – 3. show results
  – 4. go back to 1
    • Using loop – while or for
Random Number

• Generate random number between 0 to 1.0
  – `Math.random();`

• Generate random number between 0 to N
  – `(int)(Math.random() * N);`

• **Exercise:** Try to print 10 times of randomly generated numbers using loop statement
Exercise: coin gambling

• Make function that return true/false
• Inside of the function, do `random` between 0 to 1, if the value is less than 0.5 then return `true`, otherwise return `false`

• Run 1000 times, then compute mean
Factorial

• Factorial
  \[-N! = N \times (N-1) \times (N-2) \times \ldots\]

• Define function
  \[-public\ static\ int\ factorial(int\ N)\{\ /*\ fill\ me!\ */\ \}\]

• **Exercise**: Using loop statement
Recursion

• Call function itself
• Need termination phrase
Recursion Example : Factorial

```java
public static int factorialIter(int N) {
    int var = 1;
    for (int i = N; i > 0; i--)
        var *= i;
    return var;
}

public static int factorialRecur(int N) {
    if (N == 1) return 1;
    else return N * factorialRecur(N - 1);
}
```
overloading

• To make functions *Same name* but *different body*
• Why?
  – Sometimes, we need make multiple functions have similar name for program simplicity
  – Example : add functions using integer input and String inputs. They should have similar name but different body to process different inputs
• For overloading, at least one of parameters must be different to the others
Overloading Example

```java
public int add(int a, int b){
    return a+b;
}

public int add(int a2, int c2){
    return a2+b2;
}

public double add(int a, int b){
    return a+b;
}

public double add(int a2, double b2){
    return a2+b2;
}
```
• Compile-time
• Run-time
• Memory loading timing
• Usually variable/functions will be loaded to memory when it reached
• *static* keywords infers
  – The variable/function will be loaded at the beginning of running
• Why does main have *static* keyword?
Useful library **Vector**

- Array has size limit (defined at the first time)
- But sometimes, we cannot expect its optimized size
- By using Vector, we don’t have to think about it
Vector Example

*Type it and see how it works*

```java
import java.util.Vector;

public static void main(String[] args) {
    Vector myList = new Vector();
    myList.add(10);
    myList.add(15);
    System.out.println(myList);
    System.out.println(myList.get(0));
    myList.set(0, 20);
    System.out.println(myList.get(0));
}
```
Vector Example
Type specified

Type it and see how it works

```java
public static void main(String[] args) {
    Vector<Integer> myList = new Vector<Integer>();
    myList.add(10);
    myList.add(15);
    System.out.println(myList);
    System.out.println(myList.get(0));
    myList.set(0, 20);
    System.out.println(myList.get(0));
}
```

Type cannot be primitive types. Use Integer instead of int, Double instead of double, Float instead of float
Homework 2 : Big number calculator

• How can we store a number over the memory size of variable types?
  – We knows the upper-bound of variables
• Store each digit of input number as individual letter
• Then compute add, minus, multi, divide as we’ve learned in elementary school
End of class