204700
Data Structure and Programming Languages

Jakarin Chawachat

From: http://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-092-introduction-to-programming-in-java-january-iap-2010/index.htm

3.LOOPS AND ARRAYS

Assignment 2

Foo Corporation need a program to calculate how much to pay their employees.

1. Pay = hours worked x base pay
2. Hours over 40 get paid 1.5 the base pay
3. The base pay must be no less than $8.00
4. The number of hours must be no more than 60

Frequent Issues(I)

The signature of the main method cannot be modified.

```
public static void main(String[] arguments){
    ……..
}
```

Frequent Issues(II)

Return values: if you declare that the method is not void, then it has to return something!!

```
public static int pay(double basepay, int hours){
    if (basepay<8.0)     return -1;
    else if (hours >60)  return -1;
    else{
        int salary =0;
        …….
        return salary;
    }
}
```

Frequent Issues(III)

Don’t create duplicate variables with the same name

```
public static int pay(double basepay, int hours){
    int salary = 0;              //OK
    …
    int salary = 0;              //salary already defined!!
    …
    double salary = 0;       //salary already defined!!
}
```

```
class WeeklyPay {
    public static void pay(double basePay, int hours) {
        if (basePay < 8.0) {
            System.out.println("You must be paid at least $8.00/hour");
        } else if (hours > 60) {
            System.out.println("You can't work more than 60 hours a week");
        } else {
            int overtimeHours = 0;
            if (hours > 40) {
                overtimeHours = hours -40;
                hours = 40;
            }
            double pay = basePay * hours;
            pay += overtimeHours * basePay * 1.5;
            System.out.println("Pay this employee $" + pay);
        }
    }   public static void main(String[] arguments) {
        pay(7.5, 35);
        pay(8.2, 47);
        pay(10.0, 73);
    }
}
```

What we have learned so far

- Variables & types
- Operators
- Type conversions & casting
- Methods & parameters
- If statement

Outline

- Good programming style
- Loops
- Arrays

GOOD PROGRAMMING STYLE

Good Programming Style

The goal of good style is to make your code more readable

By you and by others.

Rule#1: use good (meaningful)names

```
String a1;
int a2;
double a3;                  //BAD!!

String firstName;           //GOOD
String lastName;            //GOOD
int temperature;            //GOOD
```

Rule#2: use indentation

```
public static void main (String[] arguments) {
    int x = 5;
    x = x * x;
    if (x > 20){
        System.out.println(x + “is greater than 20.”);

    }
    double y = 3.4;
}
```

Eclipse uses Ctrl-shift-F to auto-format the file

Rule#3: use whitespaces

Put whitespaces in complex expressions:

```
//BAD!!
double cel=fahr*42.0/(13.0-7.0);
//GOOD
double cel = fahr * 42.0 / (13.0 - 7.0);
```

Rule#3: use whitespaces

Put blank lines to improve readibility:

```
public static void main (String[] arguments) {
    int x = 5;
    x = x* x;

    if (x > 20) {
        System.out.println(x + “is > 20.”);
    }

    double y = 3.4;
}
```

Rule#4: do not duplicate tests

```
if (basePay < 8.0) {
    …
} else if (hours > 60) {
    …
} else if (basePay >= 8.0 && hours <= 60) {
    …
}
```

Rule#4: Do not duplicate tests

```
if (basePay < 8.0) {
    …
} else if (hours > 60) {
    …
} else {
    …
}
```

Good programming style (summary)

Use good names for variables and methods

Use indentation

Add whitespaces

Don’t duplicate tests

LOOPS

Loops

```
static void main (String[] arguments) {
    System.out.println(“Rule #1”);
    System.out.println(“Rule #2”);
    System.out.println(“Rule #3”);
}
```

What if you want to do it for 200 Rules?

Loops

Loop operators allow to loop through a block of code.

There are several loop operators in Java.

The while operator

```
while (condition) {
    statements
}
```

The while operator

```
int i = 0;
while (i < 3) {
    System.out.println(“Rule #” + i);
    i = i + 1;
}
```

Count carefully

Make sure that your loop has a chance to finish.

The for operator

```
for (initialization;condition;update) {
    statements
}
```

The for operator

```
for (int i = 0; i < 3; i = i + 1) {
    System.out.println("Rule #" +i);
}
```

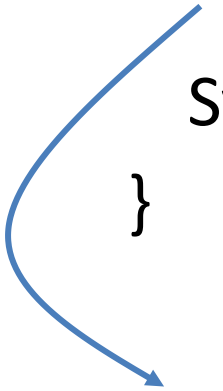
```
Rule #0
Rule #1
Rule #2
```

Note: i = i+1 may be replaced by i++

Branching Statements

break terminates a for loop or while loop

```
for (int i = 0; i < 100; i++) {
   if (i==50)
     break;
   System.out.println("Rule #" + i);
}
```



```
……
Rule #43
Rule #44
Rule #45
Rule #46
Rule #47
Rule #48
Rule #49
```

Branching Statements

continue skips the current iteration of a loop and proceeds directly to the next iteration

```
for (int i = 0; i < 100; i++) {
    if (i==50)
      continue;
    System.out.println("Rule #" + i);
}
```

```
…
Rule #46
Rule #47
Rule #48
Rule #49
Rule #51
Rule #52
Rule #53
…
```

Embedded loops

```
for (int i = 0; i < 3; i++) {
    for (int j = 2; j < 4; j++) {
        System.out.println(i + " " + j);
    }
}
```

Scope of the variable defined in the initialization: respective for block

```
0 2
0 3
1 2
1 3
2 2
2 3
```

ARRAYS

Arrays

An array is an indexed list of values.

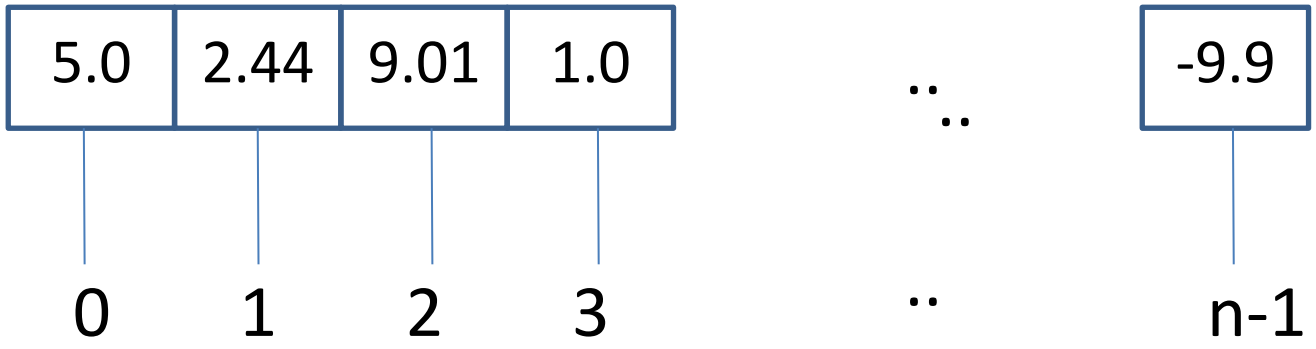
You can make an array of any type

int, double, String, etc..

All elements of an array must have the same type.

Arrays

Example: double[]



Arrays

The index starts at zero and ends at length-1.

Example:

```
int[] values = new int [5];
values[0] = 12;          //CORRECT
values[4] = 12;          //CORRECT
values[5] = 12;          //WRONG!! compiles but
                         // throws an Exception
                         // at run-time
```

Arrays

An array is defined using TYPE[].

Array are just another type.

```
int[] values;          //array of int
int[][] values;        //int[] is a type
```

Arrays

To create an array of a given size, use the operator new:

```
int[] values = new int[5];
```

or you may use a variable to specify the size:

```
int size = 12;
int[] values = new int[size];
```

Array Initialization

Curly braces can be used to initialize an array. It can ONLY be used when you declare the variable.

```
int[] values = {12, 24, -23, 47};
```

Quiz time!

Is there an error in this code?

```
int[] values = {1, 2.5, 3, 3.5, 4};
```

Accessing Arrays

To access the elements of an array, use the [ ] operator:

```
value[index]
```

Example:

```
int[] values = {12, 24, -23, 47};
values[3] = 18;                 //{12, 24, -23, 18}
int x = value[1] + 3;           //24+3
```

The length variable

Each array has a length variable built-in that contains the length of the array.

```
int[] values =      new int [12];
int size =          values.length;    //12

int[] values2 =     {11,12,13,14,15};
int size2 =         values2.length    //5
```

String arrays

```
public static void main (String[] arguments) {
   System.out.println(arguments.length);
   System.out.println(arguments[0]);
   System.out.println(arguments[1]);
}
```

Combining Loops and Arrays

Example 1:

```
int[] values = new int[5];

for (int i = 0;i<values.length;i++) {
    values[i] = i;
    int y = values[i] * values[i];
    System.out.println(y);
}
```

Combining Loops and Arrays

Example 2:

```
int[] values = new int[5];
int i = 0;
while ( i<values.length) {
    values[i] = i;
    int y = values[i] * values[i];
    System.out.println(y);
    i++;
}
```

Summary for today

1. Programming Style
2. Loops
3. Arrays

Assignment 3

A group of friends participate in the Boston Marathon.

Find the best performance.

Find the second-best performance.