Software Construction
Basics of Java Part III

Dhammika Elkaduwe

Department of Computer Engineering
Faculty of Engineering

University of Peradeniya

ILOs

- Conditional executions
- Conditions/compound conditions
- Shorthand if
- Lazy evaluation in compound conditions
- loops: for, while, do-while
- continue and break
- when to use what loop

Conditional executions

```
int year = 1996;
if(year < 1900)
  System.out.println("May be born in the 20th Century");
else
  System.out.println("Not born in the 20th Century");

if(year > 2017) System.out.println("You do not exists");
```

see Conditional.java

- If the given *condition* is true (i.e. evaluates to true) then execute the statement(s) in the *if* block
- If the condition is not true, then execute the statement(s) in the *else* block.
- *else* part can be dropped.

Compound conditions

Logical operators:

Sign	meaning
==	equals
!=	Not equals
>	greater than
>=	greater than or equal
<	less than
<=	less than or equal

Logic expressions can be joined using:

Sign	meaning
||	Logical OR
&&	Logical AND

```
if(year > 1900 && year < 2000)
  System.out.println("Born in the 20th Century");
```

see Conditional.java

Conditions

Following rules apply for conditions:

- Conditions can be compounded using logical operators as shown above
- If there are more than one statement to execute, they have to go within $\{\}$.
- If there is only one statement to execute, then brackets can be dropped.
- Instead of the expression, we can call a function that returns a *boolean* value.
- **the rules are the same for loops**

Compound conditions

Compounding is *Lazy*: You do not evaluate subsequent statements (function calls etc.) unless it is essential. Example:

```
public static boolean callMe() {
  System.out.println("callMe was called!!!");
  return false;
}
...
flag = true;
if(flag || callMe()) // callMe() will not be called
  System.out.println("flag or callMe true");
else
 System.out.println("Neighther flag or callMe is true");
```

```
\$ java Compound
flag or callMe true
..
```

Shorthand *if*

```
String result = year > 1900 && year < 2000 ?
    "Born in the 20th Century" : "Unknown";

System.out.println(result);
```

see Conditional.java

```
condition ?  <what to do if true> :  <what to do if false>;
```

Loops: the *for* loop

```
for(int i=0; i < MAX; i++)
   System.out.println("Value of i = " + i);
```

see Loops.java

Things to note:

1. Similar to a C-for loop
2. Do the initialization (i.e. $int\ i = 0$)
3. Check of the *condition* (i.e. $i < MAX$)
4. If true, do the *body* (i.e. $System.out.println(....)$)
5. Else exit the *for* loop
6. Once the body is done execute the post condition (i.e. $i++$)
7. repeat from 3

Notes: for loop

- Any part inside *for* definition can be dropped

```
for(; i < MAX; i++) // no initialization
  System.out.println("Value of i = " + i);

for(i=0; ; i++) if(i == 5) break;
for(; i < MAX;) System.out.println((i++));
for(; ; ) // infinite loop
```

- More than one operation can be done separating them with ","

```
for(i = 1, sum = 0; i < MAX; i++) // sum of 1 to 10
  sum += i;
System.out.println("Sum = " + sum);

for(i = 1, sum = 0; i < MAX; sum +=i, i++);
System.out.println("Sum = " + sum);
```

(see DifferentFor.java)

Loops: the *for* loop

```
for(int i=0; i < MAX; i++)
   System.out.println("Value of i = " + i);
```

see Loops.java

Things to note (Similar to a C-for loop):

1. Do the initialization (i.e. $int\ i = 0$)
2. Check the *condition* (i.e. $i < MAX$)
3. If true, do the *body* (i.e. *System.out.println(....)*)
4. Else exit the *for* loop
5. Once the body is done execute the post condition (i.e. $i{+}{+}$)
6. repeat from 2

Loops: the *while* loop

```
int j = 0;
while(++j < MAX) // How many times will you print?
    System.out.printf("j = %d\n", j);
```

see Loops.java

Things to note (Similar to C-while loop):

1. Check the *condition* (i.e. $++j < MAX$)
2. If true, do the *body*
3. Else exit the *while* loop
4. repeat from 1

Note:

- Suitable for unstructured loops.
- Initialization of variables, if required has to be done before the loop

Loops: the *do-while* loop

```
boolean flag = false;
do
    System.out.println("We will always execute once!");
while(flag);
```

see Loops.java

Things to note (Similar to C-do-while loop):

1. execute the body (i.e. *System.out.println("We....");*)
2. check the *condition* (i.e. *flag == true*)
3. If true, repeat the *body*
4. Else exit the loop

Note:

- Suitable for unstructured loops.
- Initialization of variables, if required has to be done before the loop
- Condition is checked after executing the body $\Longrightarrow$ body will be executed at-least once

Body of the loop: Re-iteration

```
int j = 0;
while(++j < MAX) // only one line in the body
   System.out.printf("j = %d\n", j);

int j = 0;
while(j < MAX) { // more than one line. Need brackets
   System.out.printf("j = %d\n", j);
   j++;
}
```

- If there are more than one statement in the body then {} are a **MUST**.
- If you have only one statement, brackets *can be* dropped. (you can have them as well).

Continue

```
for(i=0; i < MAX; i++) {
  if(i % 2 != 0) continue;
    System.out.printf("Value of i = %d\n", i);
}

i = 0;
while(i > 0) {
  if(i % 2 != 0 ) continue;
  System.out.printf("Value of i = %d\n", i);
  i++;
}
```

see Continue.java

- In a *for* loop the *continue* cause the control to jump to the update statement immediately.
- In a *while* loop the *continue* cause the control to jump to the Boolean expression.

Break

```
for(i=0; i < MAX; i++) {
  if(i % 2 != 0) break;
  System.out.printf("Value of i = %d\n", i);
}

i = 0;
while(i > 0) {
  if(i % 2 != 0 ) break;
  System.out.printf("Value of i = %d\n", i);
  i++;
}
```

see Continue.java

- The *break* statement cause the control to jump out of the loop.

Exercises

You have an array (of integers) called *data*. You need to:

- Find the maximum value in the array and display
- Apply the *bubble sort* algorithm:
  - Starting from the 0^{th} element to $N-2$ compare adjacent elements and swap if the one on the left side is smaller.
  - Repeat the previous step if there were any swaps
- Read all the contents of a file and display on screen. Assume that read function will return *null* when there is nothing to read.

You task is to identify suitable loops for the above three situations.

ILOs: Revisited

- Conditional executions
- Conditions/compound conditions
- Shorthand if
- Lazy evaluation in compound conditions
- loops: for, while, do-while
- continue and break
- when to use what loop