

Boaz Kantor  
Introduction to Computer Science,  
Fall semester 2010-2011  
IDC Herzliya

## Flow Control

*Raistlin: This alters time.  
Astinus: This alters nothing...Time flows on, undisturbed.  
Raistlin: And carries me with it?  
Astinus: Unless you have the power to change the course of rivers by tossing in a pebble.  
Raistlin: Watch, Astinus. Watch for the pebble!*

---

---

---

---

---

---

---

---

## Key Notion

- ▣ No control flow = statements are executed by order of text, top to bottom
- ▣ Control flow = conditionally break the order:
  - Decision-making
  - Looping
  - Branching
- ▣ Conditionally = everything is based on boolean expressions

© Boaz Kantor, IDC

2

---

---

---

---

---

---

---

---

## Statement Categories

- |  |   |
|--|---|
| <ul style="list-style-type: none"> <li>▣ Decision-making:           <ul style="list-style-type: none"> <li>- <code>if</code>, <code>switch</code>, <code>ternary</code></li> </ul> </li> <li>▣ Looping:           <ul style="list-style-type: none"> <li>- <code>while</code></li> <li>- <code>do-while</code></li> <li>- <code>for</code></li> <li>- <code>foreach</code> (future)</li> </ul> </li> </ul> | <ul style="list-style-type: none"> <li>▣ Branching:           <ul style="list-style-type: none"> <li>- <code>break</code>;</li> <li>- <code>continue</code>;</li> <li>- <code>return</code>;</li> <li>(future)</li> </ul> </li> </ul> |
|--|---|



© Boaz Kantor, IDC

3

---

---

---

---

---

---

---

---

## Boolean Expressions

- ▣ Every expression returns either true or false.
- ▣ Compound expressions are evaluated according to precedence.
- ▣ Operators:
  - Relational: == != < <= > >=
  - Logical: ! && ||
- ▣ Note: you can change variable values within boolean expressions.
- ▣ Study all of them: <http://java.sun.com/docs/books/tutorial/java/nutsandbolts/operators.html>

© Boaz Kantor, IDC

4

---

---

---

---

---

---

---

---

## Decision Making: if, if..else

- ▣ Allows choosing one block over the other.
- ▣ Syntax:
 

```
if (condition) block1
if (condition) block1 else block2
```
- ▣ A block can include more flow-control ("nesting")

© Boaz Kantor, IDC

5

---

---

---

---

---

---

---

---

## Decision Making, Example #1

```
if (cd) {
  if (--li == 0) {
    System.out.println("Insert coin");
    if (noc >= 1) {
      li += 3;
      gameOn();
    } else {
      endGame();
    }
  }
  else {
    gameOn();
  }
  gameOn();
}
```

**Easy to understand?**  
**Do you now comprehend the importance of styling?**

© Boaz Kantor, IDC

6

---

---

---

---

---

---

---

---

## Decision Making, Example #1

```

if (characterDied) {
    if (--lives == 0) {
        System.out.println("Insert coin");
        if (numberOfCoins >= 1) {
            lives += 3;
            gameOn();
        } else {
            endGame();
        }
    } else {
        gameOn();
    }
}
gameOn();

```

Indentation.  
What a relief!

---

---

---

---

---

---

---

---

## Ternary operator

- ▣ The operator is a combination of an 'if..else' statement and an expression.
- ▣ It evaluates to a value according to the condition:

```

int years = 40;
System.out.println("You are " + (years < 30 ? "young" : "old"));
int daysInYear = (years % 4 == 0 && years % 100 != 0) ? 366 :
    years % 400 == 0 ? 366 : 365;

```

---

---

---

---

---

---

---

---

## Decision Making: switch

- ▣ Similar to if, more suitable in multi-conditional blocks.
- ▣ Syntax:

```

switch (byte/short/char/int) {
    case val1: block1 break;
    case val2: block2 break;
    .
    .
    default: default_block break;
}

```

---

---

---

---

---

---

---

---

## Decision Making, Example #2

```

Scanner s = new Scanner(System.in);
char choice = s.next().toUpperCase().charAt(0);
switch (choice) {
    case 'A': playAgain(); break;
    case 'P': pause(); break;
    case 'N': next(); break;
    case 'Q': quit(); break;
    default: System.out.println("Unsupported" +
        "\n command"); break;
}

```

© Boaz Kantor, IDC

10

---

---

---

---

---

---

---

---

## Decision Making, Example #2

```

if (choice == 'A') {
    playAgain();
} else if (choice == 'P') {
    pause();
} else if (choice == 'N') {
    next();
} else if (choice == 'Q') {
    quit();
} else {
    System.out.println("Unsupported command");
}

```

© Boaz Kantor, IDC

11

---

---

---

---

---

---

---

---

## Fall-Through

```

switch (month) {
    case 2: daysInMonth = 28; break;
    case 4:
    case 6: daysInMonth = 40;
    case 9:
    case 11: daysInMonth = 30; break;
    default: daysInMonth = 31; break;
}

```

© Boaz Kantor, IDC

12

---

---

---

---

---

---

---

---

## Looping: while, do-while

- As long as **condition** is true, run the block over and over again.
- do-while** runs at least once then checks
- while** first checks then runs (hence may never run)
- Syntax:

```
while (condition) block
do block while (condition);
```

---

---

---

---

---

---

---

---

## Looping, Example #1

```
boolean someoneWon = true;
String winner = "";
do {
    game.playRound();
    winner = game.getWinnerName();
    someoneWon = !winner.isEmpty();
} while (!someoneWon);
System.out.println("The winner is: " + winner);
do {game.playRound();} while (game.getWinnerName().isEmpty());
```

---

---

---

---

---

---

---

---

## Looping, Example #2

```
int x = 1024;
while (x > 2) {
    x /= 2;
}
```

Is this the same?  
**while ((x /= 2) > 2) {}**

How about this one?  
**do {} while((x /= 2) > 2);**

---

---

---

---

---

---

---

---

## Looping: for

Similar to other loops, more suitable for counting steps.

Syntax:

```
for (initialization; condition; increment) block
```



---

---

---

---

---

---

---

---

## Looping, Example #

```
for (int x = 1; x <= 20; ++x) {
    System.out.print(x + " ");
    for (int y = 1; y <= x; ++y) {
        if (x % y == 0)
            System.out.print(" " + y);
    }
    System.out.println();
}
```

```
1: 1
2: 1 2
3: 1 3
4: 1 2 4
5: 1 5
6: 1 2 3 6
7: 1 7
8: 1 2 4 8
9: 1 3 9
10: 1 2 5 10
11: 1 11
12: 1 2 3 4 6 12
13: 1 13
14: 1 2 7 14
15: 1 3 5 15
.
Etc..
```

---

---

---

---

---

---

---

---

## Branching: break, continue, return

**break** stops the execution of the current **switch, for, while, do-while**

**continue** stops the execution of the current iteration of the loop

**return** stops the execution of the current method (future material)

---

---

---

---

---

---

---

---

## Flow Control, Overall Example

```

private static final int M = 100;
public static void main(String[] args) {
    boolean hd = false; int t = 0;
    for (int n = 3; n <= M; n++) {
        if (0 == n % 2) continue;
        hd = false;
        for (int d = 3; d <= Math.ceil(Math.sqrt(n)); ++d) {
            if (n % d == 0) {
                hd = true;
                break;
            }
        }
        if (!hd) t++;
    }
    System.out.println(t);
}

```

---

---

---

---

---

---

---

---

---

---

<http://java.sun.com/docs/books/tutorial/java/nutsandbolts/operators.html>  
<http://java.sun.com/docs/books/tutorial/java/nutsandbolts/flow.html>

## TUTORIALS

- Following slides:
- Styling conventions
- Best practices

---

---

---

---

---

---

---

---

---

---

## Styling conventions 'if', 'while', 'for'

### 'if', general format:

```

if (condition) {
    block1
}
-----
if (condition) {
    block1
} else {
    block2
}
-----
if (condition1) {
    block1
} else if (condition2) {
    block2
}

```

### 'while', general format:

```

while (condition);
-----
while (condition) {
    block1
}
-----
do {
    block1
} while (condition);

```

---

---

---

---

---

---

---

---

---

---

## Styling conventions

### 'if', 'while', 'for'

Always use braces for controlled blocks, even if they include a single row.

**'for', general format:**

```
for (init; condition; step;) {
  block1
}
-----
for (init; condition; step);
for (init1, init2; condition; step1, step2;) {
  block1
}
```

**Bad:**

```
while (i % 2 == 0) i++;
if (i % 2 == 0)
  i++;
```

**Good:**

```
if (i % 2 == 0) {
  i++;
}
```

© Boaz Kantor, IDC 22

---

---

---

---

---

---

---

---

## Styling conventions

### 'if', 'while'

If the boolean condition is too long, prefer breaking after logical operators, then indent twice to differentiate from block code:

```
if (a > b + 1000 &&
    c > d.someLongMethodName() ||
    e.someMethod().someBooleanMethod()) {
  i++;
}
```

© Boaz Kantor, IDC 23

---

---

---

---

---

---

---

---

## Styling conventions

### all boolean conditions

In a boolean expression, use nothing or ! Instead of == true or == false:

– Good:

```
if (!a.someMethod()) {
  if (a > b) {
```

– Bad:

```
if (a.someMethod() == false) {
if (a > b == true) {
```

© Boaz Kantor, IDC 24

---

---

---

---

---

---

---

---

## Styling conventions

### 'for'

- If the condition is too long, prefer breaking after logical operators, and indent twice to differentiate from block code:

```
for (init; a > b &&
    c > d.someLongMethod() ||
    e < f + 50000 / g.otherMethod;
    step;) {
    block1
}
```

---

---

---

---

---

---

---

---

## Styling conventions

### 'switch'

'switch', general format:

```
switch (value) {
case value1: block1; break;
case value2: block1; break;
.
.
default: block1; break;
}
```

Prefer writing cases in a single line, where possible

```
switch (x) {
case 1: x++; break;
case 2: x++; y = x; break;
case 3: {
    x++;
    y = x;
    z = someStaticMethod;
    break;
}
default: block1; break;
}
```

---

---

---

---

---

---

---

---

## Styling conventions

### ternary operator ('??')

'??', general format:

```
condition ? block1 : block2;
```

If nested, prefer differentiating with parentheses

```
condition1 ? (condition2 ? block1 : block2) : block3;
```

---

---

---

---

---

---

---

---

## Best practices

- ▣ Prefer using 'else if' over nesting.
- ▣ Use 'break' even in the default case.
- ▣ Prefer 'break' over boolean flags
- ▣ Prefer writing blocks shorter than 10 lines.  
If longer, consider writing a separate method.

© Boaz Kantor, IDC

28

---

---

---

---

---

---

---

---

## Best practices

- ▣ Prefer writing the shorter blocks first:

```

if (condition) {
    line1;
} else {
    line1;
    line2;
    line3;
    line4;
}

```

© Boaz Kantor, IDC

29

---

---

---

---

---

---

---

---


## Best practices

- ▣ If an 'if' block exits the method, omit the 'else' block:

```

if (condition) {
    line1;
    return true;
} else {
    line2;
    return false;
}

```



```

if (condition) {
    line1;
    return true;
}
line2;
return false;

```

© Boaz Kantor, IDC

30

---

---

---

---

---

---

---

---