

Lecture III:

Conditions + loops

Mark Huiskes

LIACS, Leiden University

Introduction to Programming, Media Technology MSc



Lecture Schedule

15 Sep	Lecture I	Variables, data types, operators + assignments
Today	Lecture II	Functions + mouse interaction
29 Sep		no class
Today	Lecture III	Conditions + loops
13 Oct	Lecture IV	Arrays
20 Oct	Lecture V	Classes + designing complex programs
27 Oct	Lecture VI	Recursion, data structures + sorting
3 Nov		no class
10 Nov	Lecture VII	Images and libraries
17 Nov	Lecture VIII	Max/MSP/Jitter (Edwin van der Heide)

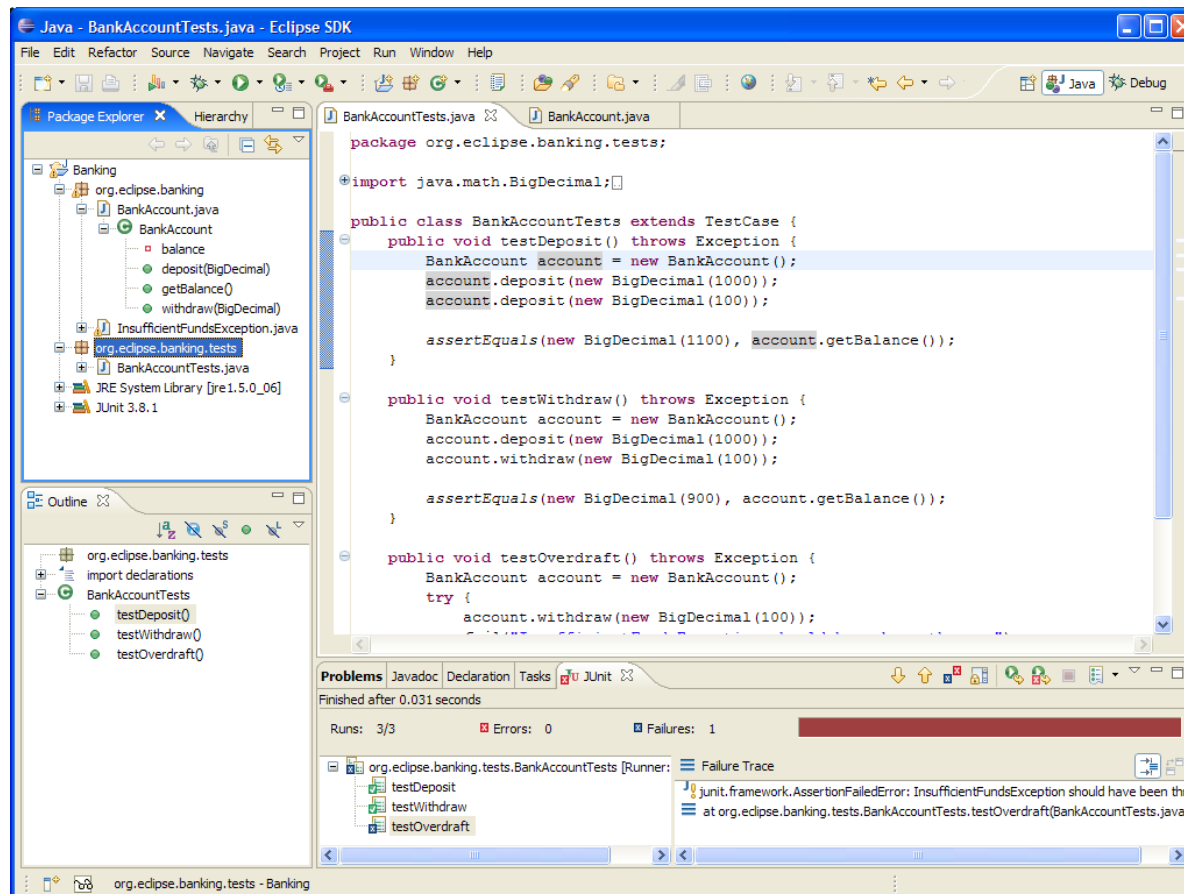
Today

1. Tips + tricks
2. Summary sketch
3. Assignments last week
4. Conditions: logical expressions and booleans
5. If-statements
6. Loops

Tips & Tricks

Using Processing from Eclipse:

<http://processing.org/learning/tutorials/eclipse/>



Summary Sketch

Assignments last week

Control Flow

Introduction

- The *control flow statements* of a language specify the order in which computations are performed

- Today we learn about conditional execution: **if-else**

```
if (something-is-true)
    do-something
else
    do-something-else
```

- And about how to loop...: **for** and **while**, e.g.

```
while (something-is-true)
    do-something
```

- Both depend on **logical expressions**: expressions that evaluate to either `true` or `false`

Conditions: Logical Expressions and Booleans

Simple logical expressions

- Simple logical expressions: comparison of two expressions by means of a **relational operator**
- Later on we will see that we can make other logical expressions with **logical operators**

Relational Operators

These are all of them

- $>$ greater than
- $>=$ greater than or equal to
- $<$ less than
- $<=$ less than or equal to

- $==$ equal to
- $!=$ not equal to

Examples

- $1 < 2$ true
- $1 < 1$ false
- $1 \leq 1001$ true
- $1 \leq 1$ true
- $1 \leq -1$ false
- $1 == 1$ true
- $1 == 1.0$ true
- $1 != 2$ true
- $1 != 1$ false
- $1000 != 1000$ false
- $1 = 1$ error

Boolean variables

- Variables of type `boolean` can store the result of a logical expression
- A boolean variable can have only two values: `true` or `false`
- **Logical expressions** are also called **boolean expressions**
- (And sometimes they are also called relational or conditional expressions, depending on context)

Example:

```
int x = 100;
boolean flag;
flag = (x >= 90);

boolean f = true;
f = (x <= -10);
println(f);
```

Conditional execution/branching: If and If-Else

1. If

```
if (logical expression) {  
    statements  
}
```

Example:

```
int x = 150;  
if (x > 100) {  
    ellipse(50, 50, 36, 36);  
}  
line(20, 20, 80, 80);
```

2. If - else

```
if (logical expression) {  
    statements  
}  
else {  
    statements  
}
```

Example:

```
int x = 50;  
if (x > 100) {  
    ellipse(50, 50, 36, 36);  
}  
else {  
    rect(33, 33, 34, 34);  
}  
line(20, 20, 80, 80);
```

Example III.0

Example III.1

Logical operators

- `&&` logical AND
- `||` logical OR
- `!` logical NOT

More logical expressions

Examples:

1. `((x > 5) && (x < 10))`

is true only if both *value-of-x* > 5 AND *value-of-x* < 10, i.e. only if $5 < \text{value-of-}x < 10$.

2. `((x < 5) || (x > 10))`

is true if *value-of-x* < 5 OR if *value-of-x* > 10

$!(x < 5)$

is true only if the value of x is NOT less than 5,
so only if *value of $x \geq 5$*

value of x	value of y	value of x && y
false	false	false
false	true	false
true	false	false
true	true	true

value of x	value of y	value of x y
false	false	false
false	true	true
true	false	true
true	true	true

value of x	value of !x
false	true
true	false

A few examples

- `int x = 3;`
`boolean b;`
`b = (x > 2) && (x < 4);`
- `int y = 5;`
`b = (x > y) && (sth-very-difficult);`
- `b = (x - y >= 0) || (x + 2 == 5)`
- `(x != x) || !(2 * x > y + 0.5)`

Example III.2

Loops

Loops

- Repeating statements
- Two main constructs: `while` and `for`

While-loop

```
while (logical expression) {  
    statements  
}
```

Example:

```
int counter = 0;  
while (counter < 10) {  
    println("hello");  
    counter++;  
}
```

Printing sequence of numbers

Example III.3

For-loops

- Another very common way to do loops
- Clear notation: all loop-variable-admin is nicely packaged in the for-statement header
- (also prevents forgetting initialization and updating)

Original code

```
size(200, 200);  
line(20, 20, 20, 180);  
line(30, 20, 30, 180);  
line(40, 20, 40, 180);  
line(50, 20, 50, 180);  
line(60, 20, 60, 180);  
line(70, 20, 70, 180);  
line(80, 20, 80, 180);  
line(90, 20, 90, 180);  
line(100, 20, 100, 180);  
line(110, 20, 110, 180);  
line(120, 20, 120, 180);  
line(130, 20, 130, 180);  
line(140, 20, 140, 180);
```

Using for

```
size(200, 200);  
for (int i = 20; i < 150; i+=10) {  
    line(i, 20, i, 180);  
}
```

For-loops

- General form:

```
for (initialization; test, update) {  
    statements  
}
```
- Initialization typically initializes the variable that will change during the loop, e.g. `i = 0` (or: `int i = 0;` the variable goes out of scope at the end of the loop)
- The for-loop keeps running as long as the test (a logical expression) is true
- Update increases or decreases the loop-variable by a certain amount, e.g. `i++` or `i+=10`
- Typical use:

```
for (counter = 0; counter < 10; counter++) {  
    // do something with counter, e.g.  
    println(counter);  
}
```

Example III.4

Nesting for-loops

Example III.5

Example III.6