Inf1-OP

Conditionals and Loops[1]

Perdita Stevens, adapting earlier version by Ewan Klein

School of Informatics

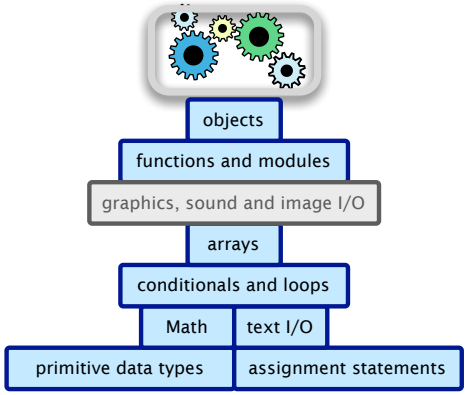
January 9, 2016

[1] Thanks to Sedgewick&Wayne for much of this content

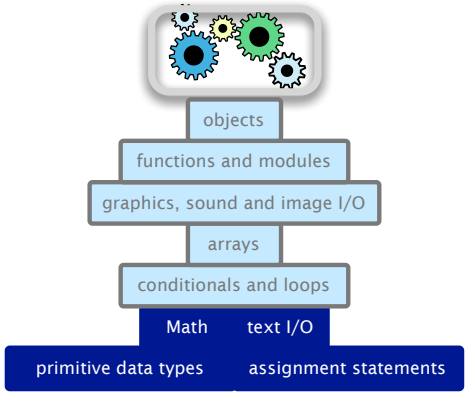
Learning Outcomes for this week

- Use `if` and `if-else` statements to execute a sequence of statements based on the truth value of Boolean expressions.
- Use nested `if-else` statements to compute results based on a number of mutually exclusive alternatives.
- Use `while`-loops to repeatedly execute a sequence of statements based on the truth value of Boolean expressions.
- Use `for`-loops to repeatedly execute a sequence of statements based on an initialization statement, a Boolean test, and an increment statement.
- Use `for`-loops to compute finite sums and finite products.

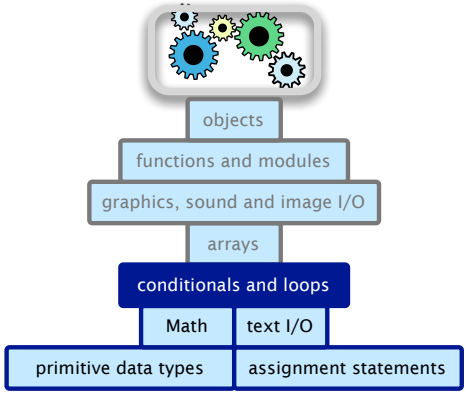
A Foundation for Programming



A Foundation for Programming



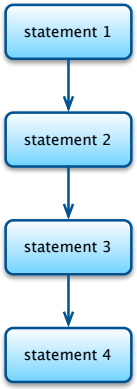
A Foundation for Programming

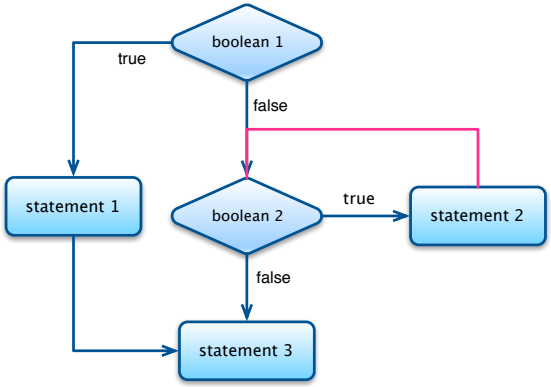


Control Flow

Control flow:

- A sequence of statements that are actually executed in a program
- Conditionals and loops enable us to choreograph control flow





If Statement

If / conditional statement:

- Evaluate a `boolean` expression E.
- If value of E is `true`, execute some statements.
- If value of E is `false`, execute some other statements — this is the *else* part of a conditional statement.

```
if (boolean expression) {
    statement T;
}
else {
    statement F;
}
```

can be any sequence of statements

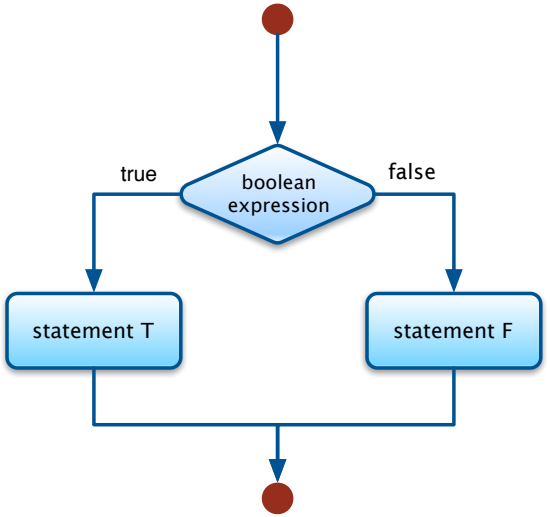
boolean expression

```
if (x > y) {
    int t = x;
    x = y;
    y = t;
}
```

sequence of statements

If Statement

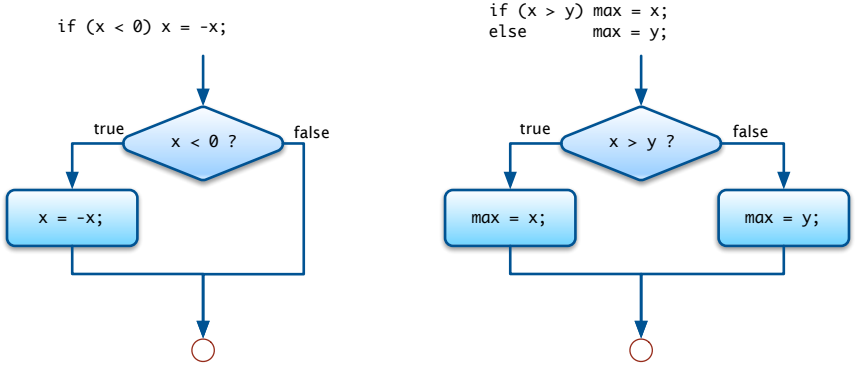
If / conditional statement — sometimes called branching structures:



If Statement

If / conditional statement:

- Evaluate a `boolean` expression.
- If `true`, execute some statements.
- If `false`, execute some other statements.



If Statement: Examples

absolute value	`if (x < 0) x = -x;`

put x and y into ascending order	`if (x > y) {`
	`  int temp = x;`
	`  x = y;`
	`  y = temp;`
	`}`

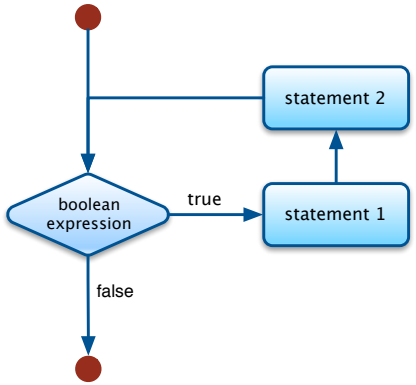
maximum of x and y	`if (x > y) max = x;` `else  max = y;`

error check for division operation	`if (den == 0) {`
	`  System.out.println("Division by zero");`
	`} else {`
	`  System.out.println("Quotient = " + num / den);`
	`}`

While Loop

The `while` loop is a structure for expressing repetition.

- Evaluate a `boolean` expression.
- If `true`, execute some statements.
- Repeat.



loop continuation condition

```
while (boolean expression) {
    statement 1;
    statement 2;
}
```

loop body

While Loop: Powers of Two

Print powers of 2 that are $\leq 2^n$ for some n. Set `n = 6`.

- Increment loop counter `i` by `1`, from `0` to `n`.
- Double `val` each time.

```
int i = 0;
int val = 1;
while (i <= n) {
  System.out.println(i + " " + val);
  i = i + 1;
  val = 2 * val;
}
```

i	val	i ≤ n	Output	
0	1	true	0	1
1	2	true	1	2
2	4	true	2	4
3	8	true	3	8
4	16	true	4	16
5	32	true	5	32
6	64	true	6	64
7	128	false		

Start Again

Powers of Two

```
public class PowersOfTwo {
   public static void main(String[] args) {
      int n = Integer.parseInt(args[0]);
      int i = 0;
      int val = 1;
      while (i <= n) {
         System.out.println(i + " " + val);
         i = i + 1;
         val = 2 * val;
      }
   }
}
```

```
% java PowersOfTwo 3
0 1
1 2
2 4
3 8
```

While Loop Challenge

Q: Is anything wrong with the following version of `PowersOfTwo`?

```
int i = 0;
int val = 1;
while (i <= n)
   System.out.println(i + " " + val);
   i = i + 1;
   val = 2 * val;
```

A: Need curly braces around statements in `while` loop. Otherwise, only the first of the statements is executed before returning to `while` condition; enters an infinite loop, printing `0 1` for ever.

(How to stop an infinite loop? At the Linux command-line, hit Control-c.)

The Increment Operator

```
int i = 0;
int val = 1;
while (i <= n) {
   System.out.println(i + " " + val);
   i = i + 1;
   val = 2 * val;
}
```

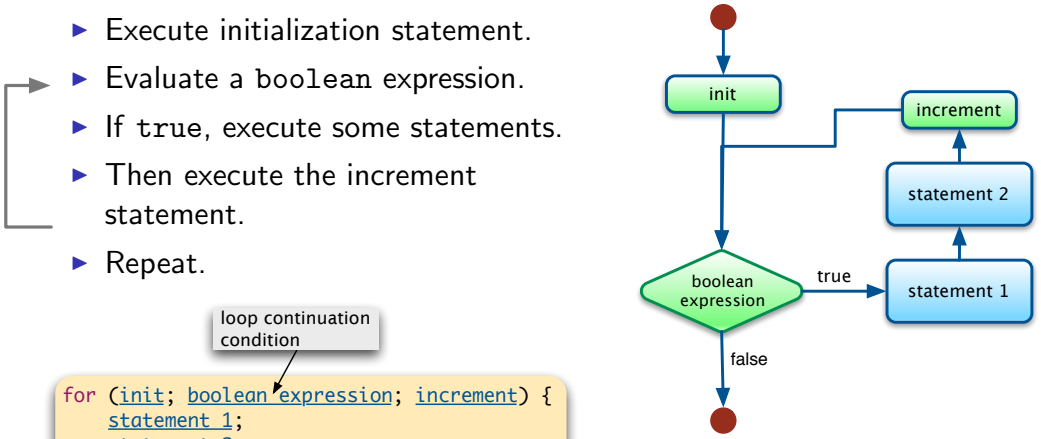
- standard assignment: `i = i + 1;`
- semantically equivalent shorthand: `i++;`

```
int i = 0;
int val = 1;
while (i <= n) {
   System.out.println(i + " " + val);
   i++;
   val = 2 * val;
}
```

For Loop

The `for` loop is another common structure for repeating things.

- Execute initialization statement.
- Evaluate a `boolean` expression.
- If `true`, execute some statements.
- Then execute the increment statement.
- Repeat.

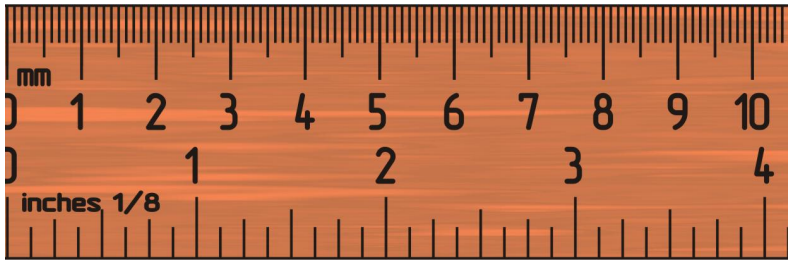


loop continuation condition

```
for (init; boolean expression; increment) {
    statement 1;
    statement 2;
}
```

loop body

Subdivisions of a Ruler



Output

```
% java Ruler
1 2 1 3 1 2 1 4 1 2 1 3 1 2 1
```

Subdivisions of a Ruler — the hard way

```
public class Ruler {
    public static void main(String[] args) {
        String ruler1 = "1";
        String ruler2 = ruler1 + " 2 " + ruler1;
        String ruler3 = ruler2 + " 3 " + ruler2;
        String ruler4 = ruler3 + " 4 " + ruler3;
        System.out.println(ruler1);
        System.out.println(ruler2);
        System.out.println(ruler3);
        System.out.println(ruler4);
    }
}
```

Output

```
% java Ruler
1
1 2 1
1 2 1 3 1 2 1
1 2 1 3 1 2 1 4 1 2 1 3 1 2 1
```

Subdivisions of a Ruler — with `for` loop

- Initialize `ruler` to `" "` (empty string).
- For each value `i` from `1` to `n`:
  sandwich two copies of `ruler` on either side of `i`.

```
public class RulerN {
   public static void main(String[] args) {
      int n = Integer.parseInt(args[0]);
      String ruler = " ";
      for (int i = 1; i <= n; i++) {
         ruler = ruler + i + ruler;
      }
      System.out.println(ruler);
   }
}
```

i	ruler
	" "
1	" 1 "
2	" 1 2 1 "
3	" 1 2 1 3 1 2 1

Subdivisions of a Ruler — with `for` loop

Output

```
% java Ruler 1
 1

% java Ruler 2
 1 2 1

% java Ruler 3
 1 2 1 3 1 2 1

% java Ruler 4
 1 2 1 3 1 2 1 4 1 2 1 3 1 2 1

% java Ruler 100
Exception in thread "main" java.lang.OutOfMemoryError
```

Loop Examples 1

Print largest power of two that is $\leq n$

```
int val = 1;
while (val <= n / 2) {
   val = 2 * val;
}
System.out.println(val);
```

Loop Examples 2

Print the result of computing the finite sum
$(1 + 2 + \ldots + n)$

```
int sum = 0;
for (int i = 1; i <= n; i++) {
   sum += i;
}
```

Loop Examples 3

Print the result of computing the finite product
$(n! = 1 \times 2 \times \ldots \times n)$

```
int product = 1;
for (int i = 1; i <= n; i++) {
   product *= i;
}
```

Nested If Statements

How to classify Scottish weather:

degrees C	verdict
< -5	wear a sweater
-5 to 0	nippy
1 to 10	normal
> 10	roastin'

4 mutually exclusive alternatives

```
String verdict;
if (temp < -5) verdict = "wear a sweater";
else {
    if (temp < 1) verdict = "nippy";
    else {
        if (temp < 11) verdict = "normal";
        else verdict = "roastin'";
    }
}
```

Nested If Statements

We don't necessarily need all those braces.

```
public class ScottishWeather {
    public static void main(String[] args) {
        String verdict;
        int temp = Integer.parseInt(args[0]);
        if      (temp < -5) verdict = "wear a sweater";
        else if (temp < 1)  verdict = "nippy";
        else if (temp < 11) verdict = "normal";
        else                verdict = "roastin'";
        System.out.println("Verdict: " + verdict);
    }
}
```

Output

```
% java ScottishWeather -1
Verdict: nippy

% java ScottishWeather 1
Verdict: normal
```

Nested If Statements

Is there anything wrong with the logic of the following code?

degrees C	verdict
< -5	wear a sweater
-5 to 0	nippy
1 to 10	normal
> 10	roastin'

4 mutually exclusive alternatives

```
String verdict;
int temp = Integer.parseInt(args[0]);
if (temp < -5) verdict = "wear a sweater";
if (temp < 1)  verdict = "nippy";
if (temp < 11) verdict = "normal";
if (temp >= 11) verdict = "roastin'";
```

Summary

Control flow:

- Sequence of statements that are actually executed in a program run.
- Conditionals and loops: enable us to choreograph the control flow.

Control Flow	Description	Examples
straight-line programs	all statements are executed in the order given	
conditionals	certain statements are executed depending on the values of certain variables	`if`, `if-else`
loops	certain statements are executed repeatedly until certain conditions are met	`while`, `for`

Tutorials

Start this week — please let the ITO know if you need to switch tutorial groups.
Labs continue this week and every week (except ILW).

This Week's Reading

Java Tutorial
pp68-86, i.e. Chapter 3 *Language Basics* from *Expressions, Statements and Blocks* to the end of the chapter.