Fundamentals of Programming for Computational Linguists

Part 3.1: Programming – Conditions and loops

Note: Please log in and start Eclipse!

Institut für Maschinelle Sprachverarbeitung, WS 17/18
Camilo Thorne (based on slides by C. Scheible)

Recap: Algorithm

Algorithm: a sequence of “simple” instructions

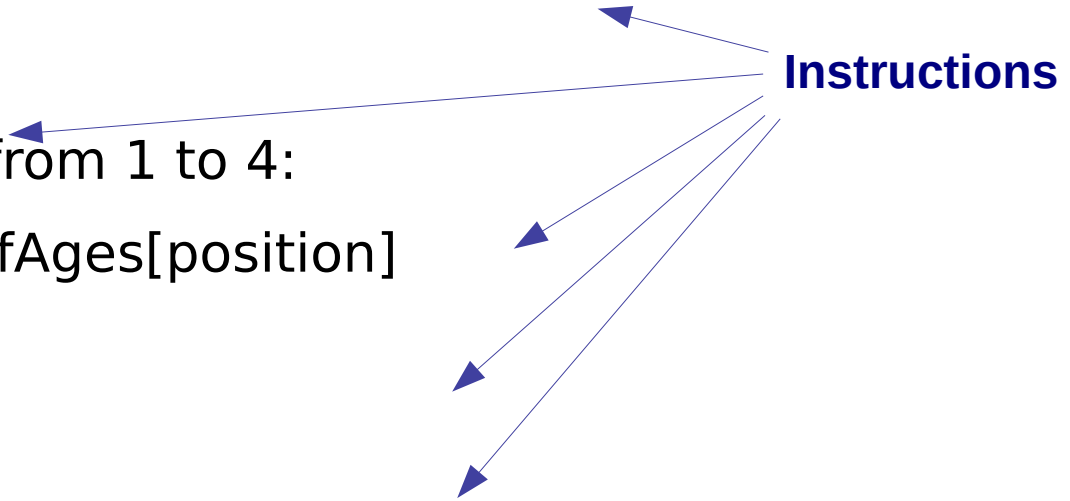
- **Unique order of steps:** The order in which things are done is fixed
- **Unique result:** The effect of each step is known and fixed
- **Think:** An algorithm is like a recipe for solving a problem

Recap: Algorithm of the day

Variables

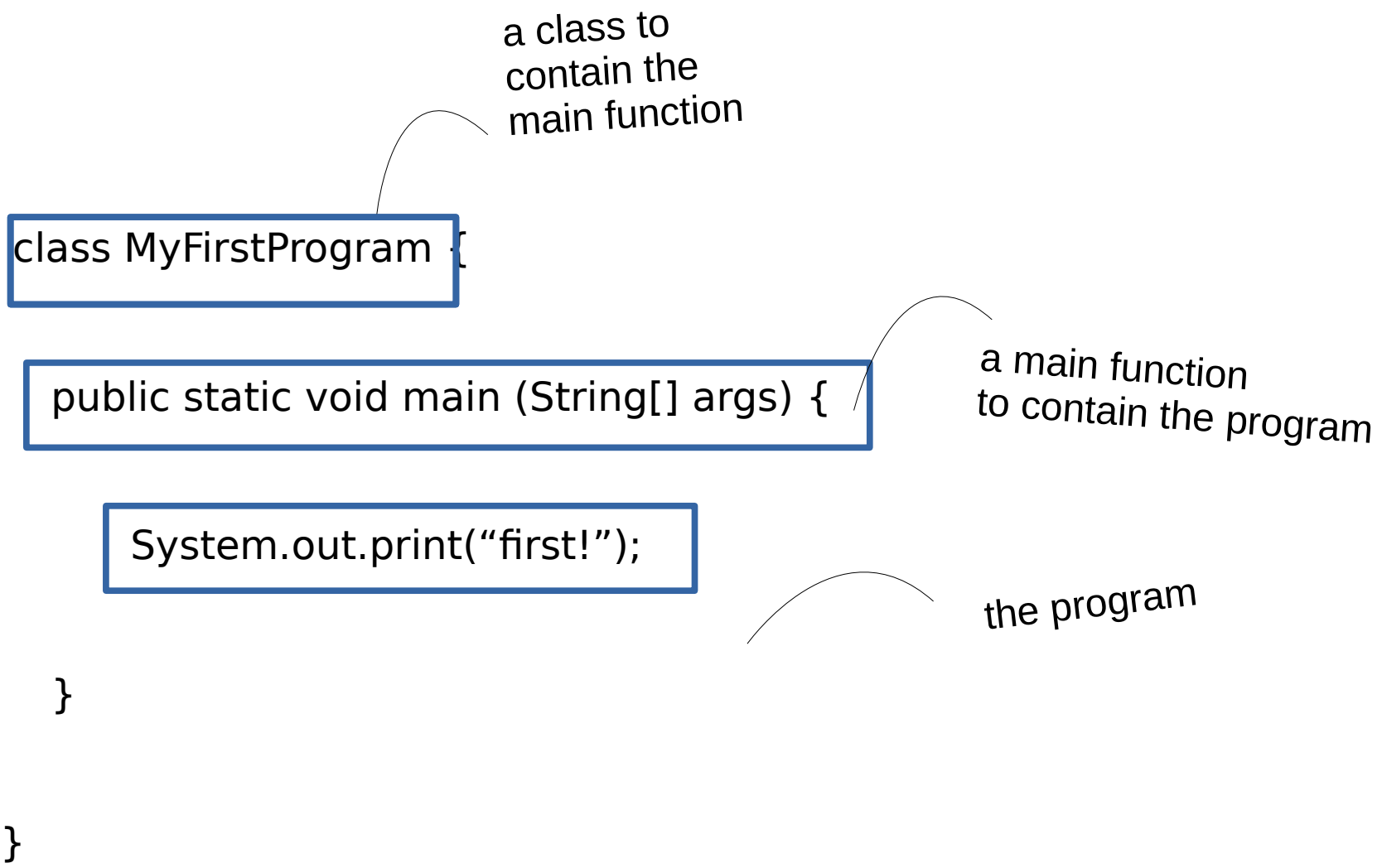
```
listOfAges = [22, 21, 24, 23]
x = 0
for each position from 1 to 4:
        x = x + listOfAges[position]


print x / 4
```



What does this algorithm do? What will x contain?

Recap: Program structure in Java



Recap: Variables

Some examples, not all are correct

```
1  int age = 24;
2  double height = 1.6;
3  String name = “Peter”
4  boolean isStudent = true;
5
6  age = 23;
7  name = “Paul”;
8  isStudent = false
9  boolean isHappy = true;
10 weight = 2.45;
11 age = -14;
12
13 age = age + 30;
14 int weight = 77;
15 int height = 184;
```

Recap: Math operations

```
1  int a = 4;
2  int b = 10;
3  int x;
4
5   x = a + b * 3;
6   x = a – 3 * b;
7   x = a * b;
8   x = a / b;
9   x = a % b;
10 x = -a + 1;
```

Recap: Booleans

Boolean variables: yes or no?

```
1 boolean a = true;
2 boolean b = false;
3 boolean x;
4
5 x = a & b;    // a and b
6 x = a | b;    // a or  b
7 x = !a;       // not a
```

Recap: Booleans

```
1  boolean a = true;
2  boolean b = false;
3  boolean x;
4
5  x = a & !b;
6
7  x = b & !a;
8
9  x = !b | !a;
10
11 x = b & !(a | b);
12
13 x = (a & !b) | (!a & b);
14
15 x = (b & !b);
```

Recap: Comparisons

```
1  int x = 10;
2  int y = 20;
3  int z = 5;
4
5  boolean comp = x > 5;
6
7  comp = y <= 20;
8
9  comp = (z > x);
10
11 comp = (y > z) & (x < y);
12
13 comp = (z > x) & (x < z);
14
15 comp = (z < x) & (z > x);
```

Conditions with if

Sometimes, we want to do things only in certain cases

```
1 boolean dogIsHungry = false;
2
3 if (dogIsHungry) {
4   System.out.println(“Feed the dog!”);
5 }
```

- The `if` statement tests whether a **condition** is true
- Only in the `true` case, the code in the `if` is executed
- We can put any boolean we want into the parentheses

Conditions with if (2)

We can also specify an alternative

```
1 boolean dogIsHungry = false;
2
3 if (dogIsHungry) {
4   System.out.println(“Feed the dog!”);
5 } else {
6   System.out.println(“Better don't feed ...”);
7 }
```

Do this if the condition is satisfied ...

... otherwise, do this

- We can put any boolean we want into the parentheses

Fundamentals of Programming for Computational Linguists

Part 3.2: Programming – Arrays, loops, functions

Note: Please log in and start Eclipse!

Institut für Maschinelle Sprachverarbeitung, WS 17/18
Camilo Thorne

Motivation: Why Java

Java is (relatively) fast, e.g., compared to Python

https://benchmarksgame.alioth.debian.org/u64q/python.html

Java is strictly typed

- Variable types always need to be specified
- Good for *beginners*: you will be aware of types
- Good for *advanced* programmers: code “documents itself”

Java gives you access to relatively low-level programming features

- Arrays (rather than just lists)
- Number types of varying size

→ **Java is good for larger projects**

Motivation: Why *not* Java

Java is (relatively) slow, e.g., compared to C++

http://benchmarksgame.alioth.debian.org/u64q/java.html

Java syntax is ugly/weird

- Lots of brackets
- Some weird-looking expressions
- Long development history → not everything looks seemless

Java programs always need the virtual machine to run

Not as low-level as other languages (e.g., C++)

Recap: Types

Some of the most basic variable types:

- int (= integer)

```
1 int age = 12;
```

- String (“text”)

```
2 String name = “Fido”;
```

- double (≈ real number)

```
3 double height = 1.03;
```

- boolean (yes or no?)

```
4 boolean dogIsHappy = true;
```

Recap: Types and values

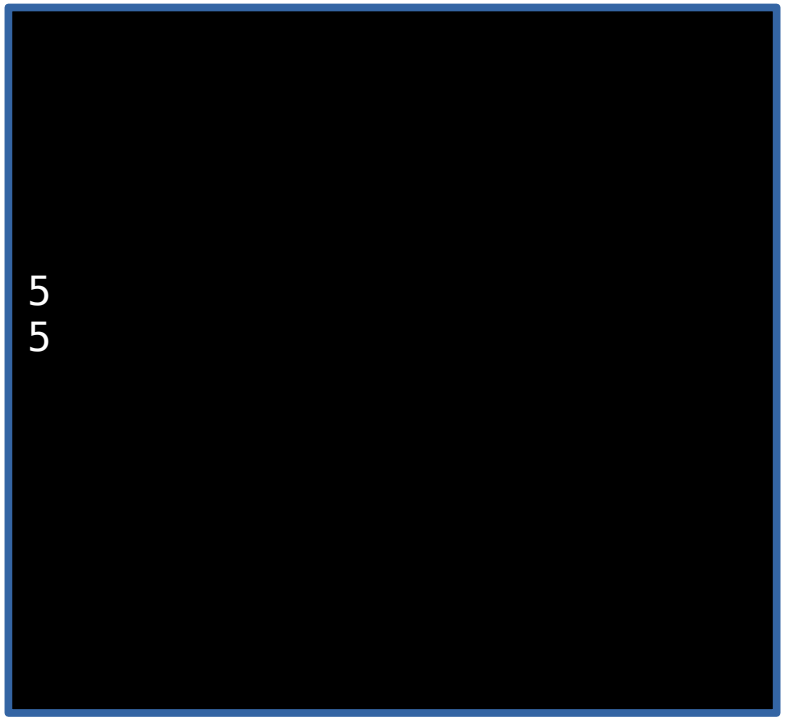
```
1  int i = 5;
2  double j = 6;
3  int aNumber = -12;
4  String name = “Peter”;
5  String anotherName;
6
7  j = -9.23;
8  boolean isDogHappy = true;
9  int numberOfPeople = 7;
10 boolean isDogNotHungry;
11 anotherName = “Lisa”;
```

For these variables, what is the

- type?
- name?
- value?

Exercise: Variables

```
1 int a = 10;
2 int b = 5;
3
4 // swap the two numbers
5 a = b;
6 b = a;
7
8 System.out.println(a);
9 System.out.println(b);
```



Not what we may have expected …

Exercise: Variables

```
1  int a = 10;
2  int b = 5;
3
4  // actual swap
5  int temp;
6  temp = a;
7  a = b;
8  b = temp;
9
10 System.out.println(a);
11 System.out.println(b);
```

Recap: if

```
1 boolean a = true;
2 boolean b = false;
3
4 if (a & ! b) {
5   System.out.println(“a and not b”);
6 }
7 else {
8   System.out.println(“the opposite”);
9 }
```

Recap: if

```
1  int x = 15;
2  int y = 20;
3
4
5  if (y < x) {
6    System.out.println(“y is smaller”);
7  }
8  else {
9    System.out.println(“y is not smaller”);
10 }
```

Example: Comparisons and if

```
1
2  int age = 20;
3
4  if (age > 150) {
5    System.out.println(“age is too large”);
6  } else if (age < 0) {
7    System.out.println(“age is too small”);
8  } else {
9    System.out.println(“age is okay!”);
10 }
11
```

```
age is okay!
```

Example: Comparisons and if

```
1  int x = 10;
2  int y = 20;
3  int z = 5;
4
5  if ((y < 100) & (x < 100)) {
6    System.out.println(“x and y are smaller than 100”);
7  }
8
9  if (y < 1000) {
10   System.out.println(“y is smaller than 1000”);
11 }
```

```
x and y are smaller than 100
y is smaller than 1000
```

Example: Comparisons and if

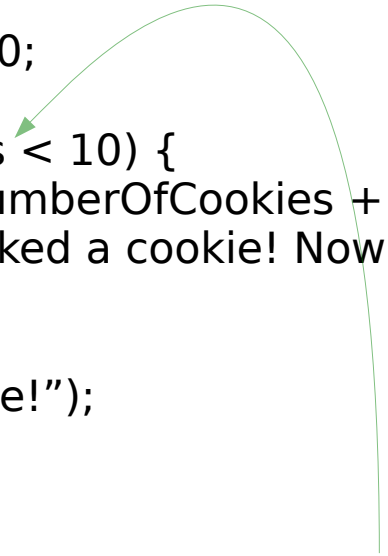
```
1 int x = 10;
2 int y = 20;
3 int z = 5;
4
5 if ((y < 100) & (x < 100)) {
6   System.out.println(“x and y are smaller than 100”);
7 } else if (y < 1000) {
8   System.out.println(“y is smaller than 1000”);
9 }
```

```
x and y are smaller than 100
```

Looping with while

Sometimes, we need to repeat something while some condition is met

```
1 int numberOfCookies = 0;
2
3 while (numberOfCookies < 10) {
4   numberOfCookies = numberOfCookies + 1;
5   System.out.println(“Baked a cookie! Now we have ” + numberOfCookies);
6 }
7
8 System.out.println(“Done!”);
```



- The while loop repeats the code in it as long as a **condition** is true
- How is this different from if?
- Basically a **repeated** if
- Each round in the loop is called an **iteration**
- **Question:** When does this program end?

Looping with while

```
1 int i = 0;
2
3 while (i < 10) {
4   System.out.println(i);
5   i = i + 1;
6 }
```

```
0
1
2
3
4
5
6
7
8
9
```

Done!

Looping with while

```
1 int i = 0;
2
3 while (i < 10) {
4   i = i + 1;
5   System.out.println(i);
6 }
7
8 System.out.println(i);
```

```
1
2
3
4
5
6
7
8
9
10
10
```

Done!

Looping with while

```
1 int i = 10;
2
3 while (i < 20) {
4   i = i + 1;
5   System.out.println(i);
6 }
```

```
11
12
13
14
15
16
17
18
19
20
```

Done!

Looping with while

```
1 int i = 0;
2
3 while (true) {
4   i = i + 1;
5   System.out.println(i);
6 }
```

→ **Infinite loop!**

Looping with while

```
1 int i = 5;
2
3 while (i < 5) {
4   i = i - 1;
5   System.out.println(i);
6 }
```

Nothing printed, because ...

... i is not < 5

Looping with while

```
1 int i = 5;
2
3 while (i <= 5) {
4   i = i - 1;
5   System.out.println(i);
6 }
```

```
4
3
2
1
0
-1
-2
...
```

→ **Infinite loop!**

Looping with while

```
1 int i = 5;
2
3 while (i > 0) {
4   i = i - 1;
5   System.out.println(i);
6 }
```

```
4
3
2
1
0
```

Done!

Looping with while

```
1  int i = 0;
2
3  while (i < 10) {
4
5    if (i > 3) {
6      System.out.println(i);
7    }
8
9    i = i + 1;
10 }
```

```
4
5
6
7
8
9
```

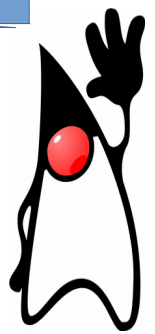
Done!

Looping with while

```
1 int i = 0;
2
3 while (i < 10) {
4   if (i % 2 == 0) {
5     System.out.println(i);
6   }
7   i = i + 1;
8 }
```

```
0
2
4
6
8
```

Exercises 2.3 & 2.4



Done!