Fundamentals of Programming for Computational Linguists

Part 4.1: Programming – Putting everything together

<u>Note</u>: Please log in and start Eclipse!

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

Recap: Algorithm of the day

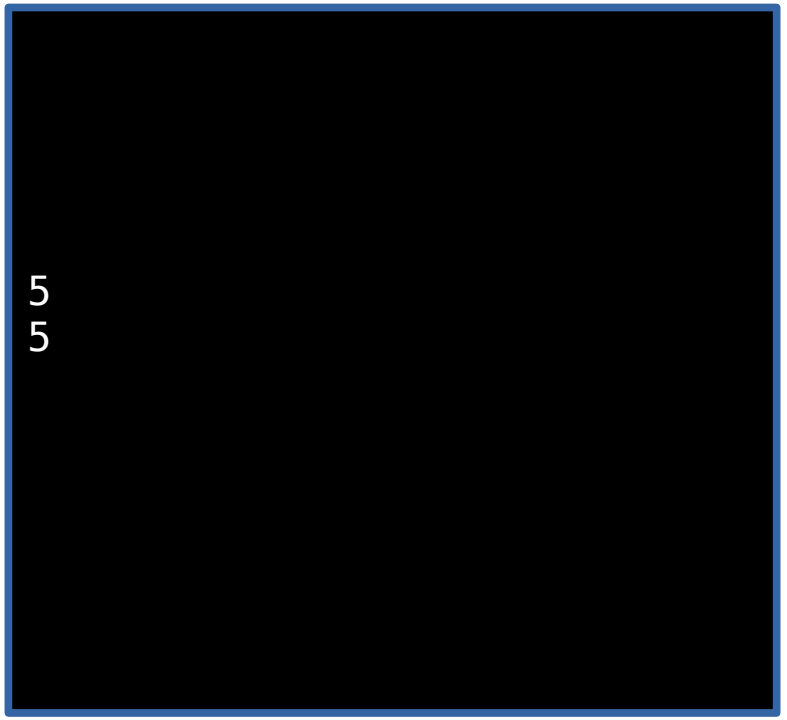
```
students = [“Peter”, “Mary”, “Lisa”, “John”]
x = -1
for each position from 1 to 4:
        if students[position] == “Lisa”:
                x = position

print x
```

What does this algorithm do? What will x contain?

Recap: 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 …

Recap: 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 }
```

Looping with while

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

```
0
1
2
3
```

Done!

Looping with while

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

```
3
6
```

Done!

Looping with while

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

```
3
6
```

Done!

Looping with while

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

```
bla
bla
bla
bla
bla
```

Looping with while

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

```
bla
blabla
blablabla
blablablabla
blablablablabla
```

Looping with while

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

```
1
3
6
10
15
```

Yesterday's assignments

Looping with for

There's a special loop for counting things

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

```
10
```

- **Question:** What does this program print?
- The for loop has three parts:
  - declaration of the **loop counter** int i = 0;
  - the loop condition i < 5;
  - the update i++;

i++ means
i = i + 1

Example: Looping with for

What does this loop print?

```
1 for (int i = 0; i < 7; i++) {
2   System.out.println(i);
3 }
```

```
0
1
2
3
4
5
6
```

Example: Looping with for

What does this loop print?

```
1 for (int i = 0; i < 7; i++) {
2   System.out.println(i + 2);
3 }
```

2
3
4
5
6
7
8

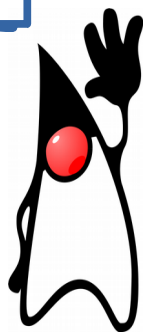
Example: Looping with for

What does this program print?

```
1 int result = 0;
2
3 for (int i = 1; i < 4; i++) {
4   result = result + i;
5 }
```

1
3
6

Exercises 3.1 & 3.2



Variable scope

loop can access variables from outside ...

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

but variables from within the loop will be gone

- Variables live where they were declared!
- Variable **in loop** exists only **in loop**
- After the loop ends, the variable is gone

Arrays

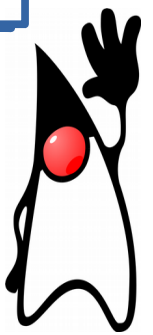
Multiple variables of the same type that belong together

```
1 String[] studentNames = {“Mary”, “Peter”, “John”};
```

0 1 2

a sequence of 3 String variables

Declare a new int array and put some numbers into it



Arrays

What if we don't know the values before?

```
1 String[] studentNames = new String[3];
2
3 // later ...
4 studentNames[0] = “Mary”;
5 studentNames[1] = “Peter”;
6 studentNames[2] = “Lisa”;
```

a sequence of 3 String variables

the “index”

Arrays

What if we access an element beyond the size of the array?

```
1 String[] studentNames = new String[3];
2
3 // later ...
4 studentNames[4] = “Frank”;
```

the 4th element?!

```
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: 4
at de.ims.javaclass.Testing.main(Arrays.java:4)
```

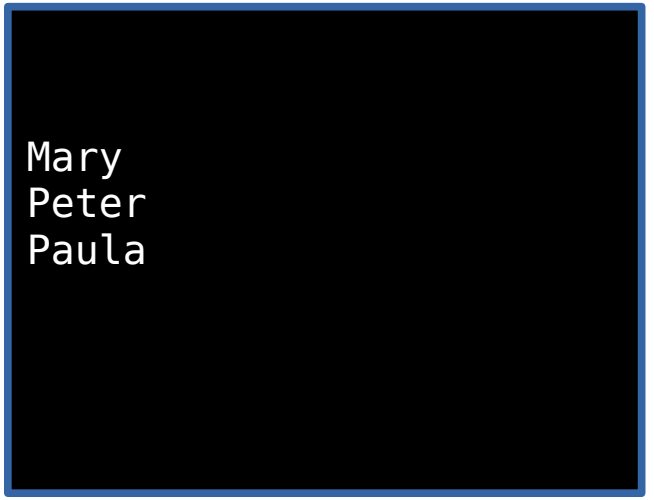
Using arrays

```
1 String[] studentNames = {“Mary”, “Peter”, “John”};
2
3 System.out.println(studentNames[1]);
4
5 studentNames[2] = “Paula”;
6
7 System.out.println(studentNames.length);
8
9 System.out.println(studentNames);
```

```
Peter
3
[Ljava.lang.String;@15db9742
```

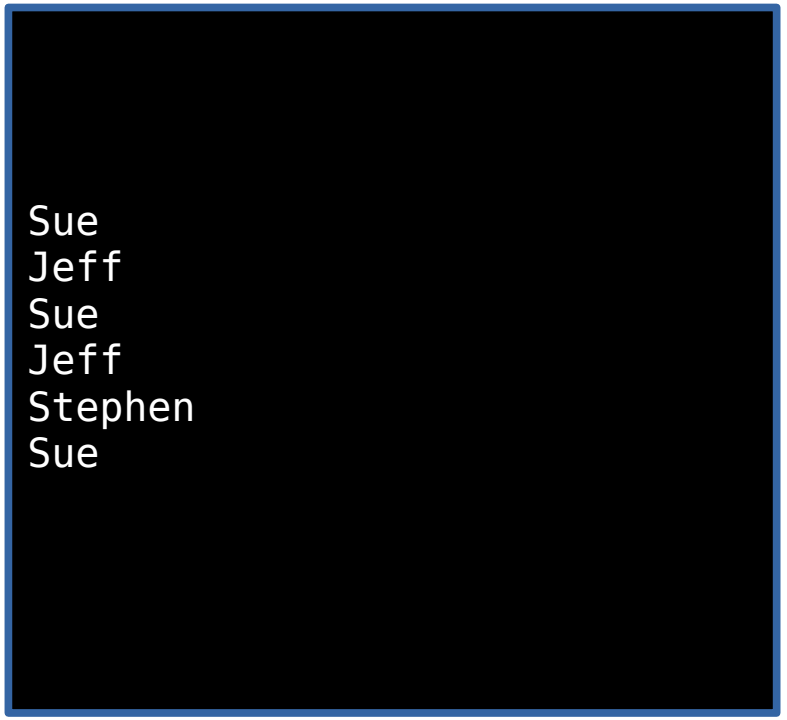
Looping over arrays

```
1 String[] studentNames = {“Mary”, “Peter”, “Paula”};
2
3 for (int i = 0; i < studentNames.length; i++) {
4   System.out.println(studentNames[i]);
5 }
```



Putting things together

```
1  String[] names = {“Sue”, “Jeff”, “Lisa”, “Mary”, “Daniel”};
2
3  String firstName = names[0];
4  String secondName = names[1];
5
6  System.out.println(firstName);
7  System.out.println(secondName);
8
9  System.out.println(names[0]);
10 System.out.println(names[1]);
11
12 String replacement = “Stephen”;
13 names[0] = replacement;
14
15 System.out.println(names[0]);
16 System.out.println(firstName);
```



Putting things together

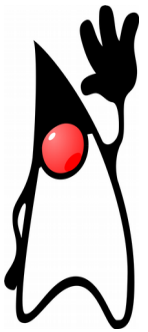
```
1  String[] names = {“Sue”, “Jeff”, “Lisa”, “Mary”, “Daniel”};
2
3  int numberOfNames = names.length;
4  int highestIndex = numberOfNames – 1;
5
6  System.out.println(highestIndex);
7
8  String lastName = names[highestIndex];
9
10 System.out.println(lastName);
```

Putting things together

```
1  String[] names = {“Sue”, “Jeff”, “Lisa”, “Mary”, “Daniel”};
2
3  System.out.println(names[0]);
4  System.out.println(names.length);
5
6  System.out.println(names[2]);
7  System.out.println(names[1 + 2]);
8
9  int i = 4;
10 System.out.println(names[i]);
11
12 i = 2;
13 System.out.println(names[i]);
14
15 int aNumber = 3;
16 System.out.println(names[aNumber]);
17
18 System.out.println(names[aNumber – 2]);
19
20 System.out.println(names[names.length - 1]);
```

```
Sue
5
Lisa
Mary
Daniel
Lisa
Mary
Jeff
Daniel
```

Exercises 3.3 & 3.4



Recap: Looping with for

What does this loop print?

```
1 for (int i = 0; i < 7; i++) {
2   System.out.println(i);
3 }
```

```
0
1
2
3
4
5
6
```

Recap: Looping with for

What does this loop print?

```
1 for (int i = 0; i < 7; i++) {
2   System.out.println(i + 2);
3 }
```

2
3
4
5
6
7
8

Fundamentals of Programming for Computational Linguists

Part 4.2: Algorithms and functions

Note: Please log in and start Eclipse!

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

Recap: Arrays

```
1  String[] names = {“Sue”, “Jeff”, “Lisa”, “Mary”, “Daniel”};
2
3  String nameAt2 = names[2];
4
5  System.out.println(nameAt2);
6
7  nameAt2 = “Amy”;
8
9  System.out.println(nameAt2);
10 System.out.println(names[2]);
11
12 System.out.println(names.length);
```

Recap: Looping with for

What does this loop print?

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

```
0
1
2
3
4
```

Recap: Looping with for

What does this loop print?

```
1 for (int i = 6; i > 1; i--) {
2   System.out.println(i);
3 }
```

```
6
5
4
3
2
```

Recap: Looping with for

What does this loop print?

```
1 for (int i = 0; i < 7; i++) {
2   System.out.println(7 - i);
3 }
```

```
7
6
5
4
3
2
1
```

Exercise: Looping with for

What does this loop print?

```
1 String[] words = {"A", "B", "C"};
2
3 for (int i = 0; i < words.length; i++) {
4   System.out.println(words[i]);
5 }
```

A
B
C

Exercise: Looping with for

What does this loop print?

```
1 String[] words = {"A", "B", "C", "D", "E"};
2
3 for (int i = 2; i < words.length; i++) {
4   System.out.println(words[i]);
5 }
```

C
D
E

Exercise: Looping with for

What does this loop print?

```
1 String[] words = {"A", "B", "C"};
2 String result = “”;
3
4 for (int i = 0; i < words.length; i++) {
5   result = result + words[i];
6 }
7
8 System.out.println(result);
```

ABC

Exercise: Looping with for

What does this loop print?

```
1 String[] words = {"A", "B", "C"};
2 String result = “”;
3
4 for (int i = 0; i < words.length; i++) {
5   result = result + words[words.length - i];
6 }
7
8 System.out.println(result);
```

ERROR !

Exercise: Looping with for

What does this loop print?

```
1 String[] words = {"A", "B", "C"};
2 result = “”;
3
4 for (int i = 0; i < words.length; i++) {
5   result = result + words[words.length – (i + 1)];
6 }
7
8 System.out.println(result);
```

CBA

Nested loops

```
1 for (int i = 1; i < 6; i++) {
2   for (int j = 1; j < 6; j++) {
3     System.out.print(i * j);
4     System.out.print(“ “);
5   }
6
7   System.out.println();
8 }
```

```
1 2 3 4 5
2 4 6 8 10
3 6 9 12 15
4 8 12 16 20
5 10 15 20 25
```

More loops

```
1 String[] words = {"A", "B", "C"};
2
3 for (int i = 0; i < words.length; i++) {
4   for (int j = 0; j < words.length; j++) {
5     System.out.println(words[i] + " " + words[j]);
6   }
7 }
8
```

```
A A
A B
A C
B A
B B
B C
C A
C B
C C
```

More loops

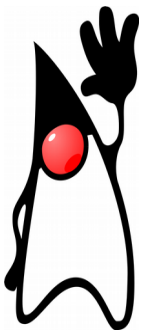
```
1  int[] numbers1 = {1, 3, 5};
2  int[] numbers2 = {2, 1, 3};
3
4  for (int i = 0; i < numbers1.length; i++) {
5    for (int j = 0; j < numbers2.length; j++) {
6      if (numbers1[i] == numbers2[j]) {
7        System.out.println("Yes!");
8      }
9    }
10 }
11
```

```
Yes!
Yes!
```

More loops

```
1 String[] words1 = {"the", "book", "is", "long"};
2 String[] words2 = {"a", "parrot"};
3
4 for (int i = 0; i < words1.length; i++) {
5     for (int j = 0; j < words2.length; j++) {
6       System.out.println(words1[i] + " " + words2[j]);
7     }
8 }
9
```

Exercise 4.3 + 5



```
the a
the parrot
book a
book parrot
is a
is parrot
long a
long parrot
```

More array operations

Sorting an array

```
1 int[] someNumbers = {7, 99, 3, 2, 15};
2
3 Arrays.sort(someNumbers);
```

```
[2, 3, 7, 15, 99]
```

Printing an array

```
4 System.out.println(Arrays.toString(someNumbers))
  ;
```

Implementing an algorithm

Imagine the following problems

1. Find out whether the word “platypus” occurs in *Harry Potter*

2. Find out whether the word “platypus” occurs in an English dictionary

Is one of these problems easier than the other?
Why?

In the dictionary, words are ordered alphabetically!

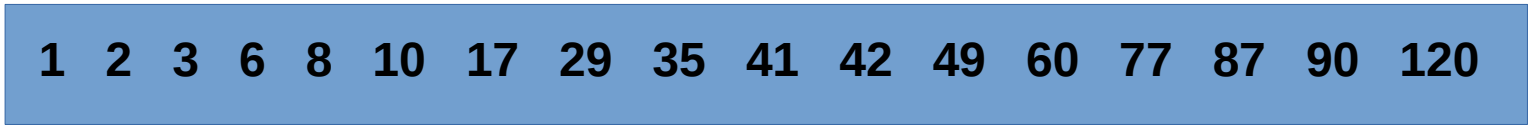
Binary search

1	2	3	6	8	10	17	29	35	41	42	49	60	77	87	90	120

Question: is the number 77 in the list?

Idea: “rule out” parts of the list

Binary search (simplified)



INPUT: integer array A, integer t
OUTPUT: true, if t is in A

```
1  l = 0
2  r = n − 1              // n is the length of A
3  while (not(l > r))
4        m = floor((l + r)/2)
5        if A[m] < t
6              l = m + 1
7        if A[m] > t
8              r = m − 1
9        if A[m] = t
10             print true
11             break
```

in Java?

Summary: Arrays

```
1 String[] studentNames = {“Mary”, “Peter”, “John”};
```

- An array is a sequence of variables of the same types
- We can access elements by index
- We can change elements
- The number of elements is fixed at the beginning
  - We cannot increase array with extra elements!
  - We cannot delete elements to decrease array!

Functions

```
1  class Example {
2    public static void main (Sting[] args) {
3      String[] words1 = ...;
4      String[] words2 = ...;
5      String[] words3 = ...;
6      String[] words4 = ...;
7
8      for (int i = words1.length - 1; i >= 0; i--) {
9        System.out.println(words1[i]);
10     }
11   }
12 }
```

First question: what does this program do again?

What if I want to print the other arrays as well?

Functions

```
1  class Example {
2    public static void main (Sting[] args) {
3      String[] words1 = ...;
4      String[] words2 = ...;
5      String[] words3 = ...;
6      String[] words4 = ...;
7
8      for (int i = words1.length - 1; i >= 0; i++) {
9        System.out.println(words1[i]);
10     }
11
12     for (int i = words2.length - 1; i >= 0; i++) {
13       System.out.println(words2[i]);
14     }
15
16     for (int i = words3.length - 1; i >= 0; i++) {
17       System.out.println(words1[i]);
18     }
19
20     for (int i = words4.length - 1; i >= 0; i++) {
21       System.out.println(words4[i]);
22     }
23   }
24 }
```

Functions

```
1     for (int i = words1.length - 1; i >= 0; i++) {
2       System.out.println(words1[i]);
3     }
4
```

```
1     for (int i = someArray.length - 1; i >= 0; i++) {
2       System.out.println(someArray[i]);
3     }
4
```

```
1  static void printArray (String[] someArray) {
2     for (int i = someArray.length - 1; i >= 0; i++) {
3       System.out.println(someArray[i]);
4     }
5  }
```

Functions

```
1  class Example {
2    static void printArray (String[] someArray) {
3      for (int i = someArray.length - 1; i >= 0; i--) {
4        System.out.println(someArray[i]);
5      }
6    }
7
8    public static void main (String[] args) {
9      String[] words1 = ...;
10     String[] words2 = ...;
11     String[] words3 = ...;
12     String[] words4 = ...;
13
14     printArray(words1);
15     printArray(words2);
16     printArray(words3);
17     printArray(words4);
18   }
19 }
```

Functions with return values

```
1 class Example {
2   public static void main (Sting[] args) {
3     System.out.println(4 * 4 + 3);
4     System.out.println(3 * 3 + 2);
5     System.out.println(5 * 5 + 4);
6     System.out.println(12 * 12 + 11);
7   }
8 }
```

Print statements do not pass information

We cannot reuse results in later computations

We want to be able to pass information to other methods/functions

We want to make our program more concise, write a line of code instead of possibly k lines for each possible variable

Functions with return values

We need to specify the value to be returned and its type:

```
1 static int squarePlusSomething (int number) {
2    int result = number * number + (number – 1);
3    return result;
4 }
```

Functions with return values

- **Example:** a **function** that sums up all numbers up to n

```
1  class OurProgram {
2
3    static int sumUpTo(int n) {
4      int sum = 0;
5
6      for (int i = 1; i <= n; i++) {
7        result = result + i;
8      }
9
10     return sum;
11   }
12
13
14   public static void main(String[] args) {
15     // ...
16   }
17 }
```

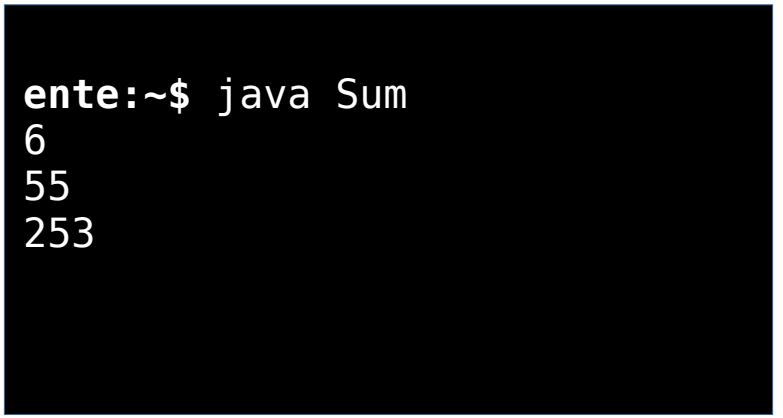
our program now has > 1 functions

Functions with return values

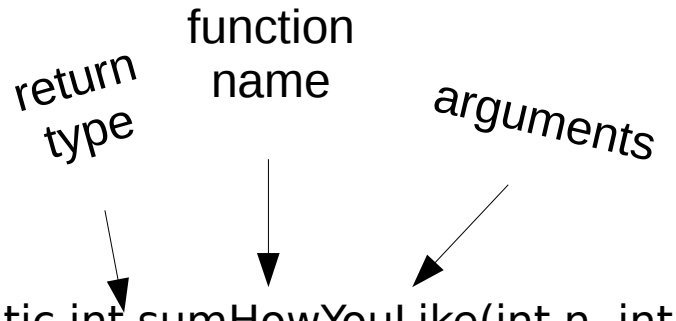
```
1 static int sumUpTo(int n) {
2   int sum = 0;
3
4   for (int i = 1; i <= n; i++) {
5     sum = sum + i;
6   }
7
8   return sum;
9 }
```

Our function can be used in our main program as follows

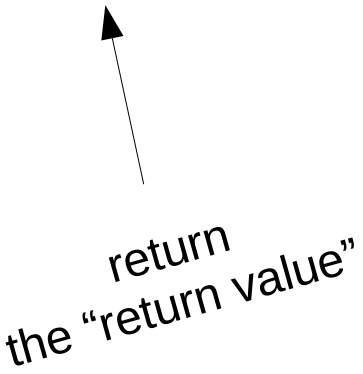
```
1 public static void main (String[] args) {
2   System.out.println(sumUpTo(3));
3   System.out.println(sumUpTo(10));
4   System.out.println(sumUpTo(22));
5 }
```

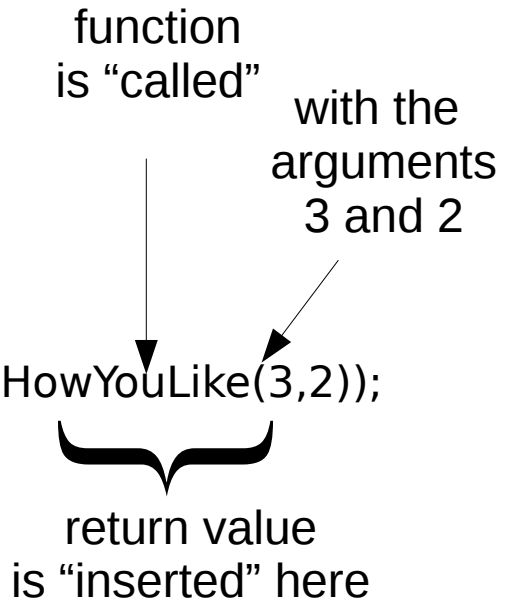


Functions (summing up)




```
1 static int sumHowYouLike(int n, int m) {
2   int sum = 0;
3
4   // some computation ...
5
6   return sum;
7 }
```






```
1 System.out.println(sumHowYouLike(3,2));
```

Functions (summing up)

- You've seen some examples of functions before
- **Example 1:** the printing function

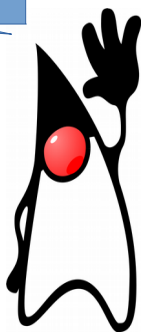
```
1 System.out.println(5);
```

- **Question:** What's different between this function and our sumUpTo(5)?

- **Example 2:** the main function

```
1 public static void main (String[] args) {
2
3 }
```

Exercise 4.3



- **Question:** What's different between this function and the others?

Variable scope for functions

```
1  static double circumference(double r) {
2    r = r * 2; // diameter
3
4    double result = r * 3.14;
5
6    return result;
7  }
8
9  public static void main(String[] args) {
10   int radius = 10;
11
12   double c = circumference(radius);
13
14   System.out.println(radius);
15
16   System.out.println(result);
17 }
18
```

value is copied

r is still 10

error, result doesn't exist in main!

- Variables live where they were declared!
- Variable **in function** exists only **within function**
- Variable **in loop** exists only **in loop**
- Variable **in condition** exists only **in condition**

...

String processing

```
1 String sentence = “This is a nice sentence!”;
```

Lowercasing and uppercasing

```
2 System.out.println( sentence.toLowerCase() );
3 System.out.println( sentence.toUpperCase() );
```

Trimming

```
5 String manySpaces = “                    text                    “;
6 System.out.println( manySpaces.trim() );
```

removes the spaces around the string

Replacement

```
5 String word = “awful”;
6 word = word.replace(“ful”, “some”);
7 System.out.println(word);
```

Splitting at spaces

```
4 String[] tokens = sentence.split(“ ”);
5 System.out.println( Arrays.toString(tokens) );
```

```
this is a nice sentence!
THIS IS A NICE SENTENCE!
text
awesome
[This, is, a, nice, sentence!]
```

Some more string processing

```
1 String sentence = “This is a nice sentence!”;
```

Length of a string

```
2 int length = sentence.length();
3 System.out.println(length);
```

Structure of the string

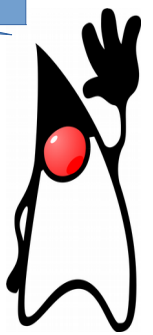
```
4 System.out.println( sentence.contains(“ice”) );
5 System.out.println( sentence.startsWith(“That”) );
6 System.out.println( sentence.endsWith(“!”) );
```

Cutting parts

```
7 String part = sentence.substring(16,20);
8 System.out.println( part );
```

```
24
true
false
true
ente
```

Exercise 4.4



That's all!

What we covered

- Algorithms
- Basic programming
- Basic Java

What's missing

- Practice
- Lots of neat built-in functionality
- Other programming languages
- OOP

What's wrong with this program

```
1 name = “Peter”
2 i = 10
3
4 names = [“Mary”, “Peter”, “Sue”]
5
6 for name in names:
7   print “Hi”, name
```

Answer: Nothing, it's a program in Python

- Types are not explicit anymore
- Loops can go directly over the list, no need for index
- Overall less verbose/cluttered
- **But: Internally, the things you learned here still apply!**
- **This is why you learned things the “hard” way first!**