

5. Defining Classes and Methods

Harald Gall, Prof. Dr.
Michael Würsch, Dr.
Institut für Informatik
Universität Zürich
<http://seal.ifi.uzh.ch/info1>



Learning Objectives

- Get familiar with the object-oriented terminology
- Learn how to define classes, attributes, and methods
- Learn how to obtain classes, attributes and methods from a natural language description
- Learn to use the class String and the Java API in general

© 2008 Pearson Education, Inc., Walter Savitch and Frank Carrano

2

Assembly Language – The old fashioned Way

```

.model tiny
.code
org 100h

main proc

    mov ah,9           ; Display String Service
    mov dx,offset hello_message ; Offset of message (Segment DS is the right
                           ; segment in .COM files)
    int 21h           ; call DOS int 21h service to display message at
                           ; ptr ds:dx

    retn             ; returns to address 0000 off the stack
                           ; which points to bytes which make int 20h (exit
                           ; program)

hello_message db 'Hello, world!'

main endp
end main

```

© 2008 Pearson Education, Inc., Walter Savitch and Frank Carrano

3

Assembly Language – The old fashioned Way

```

.section .rodata
string:
.ascii "Hello, world!\n"
length:
.quad .-string #Dot = 'here'

.section .text
.globl _start #Make entry point visible to linker
_start:
movq $4, %rax #4=write
movq $1, %rbx #1=stdout
movq $string, %rcx
movq length, %rdx
int $0x80 #Call Operating System
movq %rax, %rbx #Make program return syscall exit status
movq $1, %rax #1=exit
int $0x80 #Call System Again

```

Object-Oriented Terminology

- Objects/Instances
 - are an abstraction of real-world things
 - have a state, behavior, and identity
 - are instances of a single class
- Classes
 - are blueprints for a family of objects
 - define attributes and methods
- Attributes/Instance Variables
 - their values at runtime represent the state or data of an object
- Methods
 - define the actions/behavior of an object of a class
 - access/change the state of an object
 - call other methods

Example: Automobile

- A class **Automobile** as a blueprint

```

Class Name: Automobile
Data:
amount of fuel _____
speed _____
license plate _____
Methods (actions):
accelerate:
How: Press on gas pedal.
decelerate:
How: Press on brake pedal.

```

Class and Method Definitions

First Instantiation:
Object name: patsCar
amount of fuel: 10 gallons
speed: 55 miles per hour
license plate: "135 XJK"

Second Instantiation:
Object name: suesCar
amount of fuel: 14 gallons
speed: 0 miles per hour
license plate: "SUES CAR"

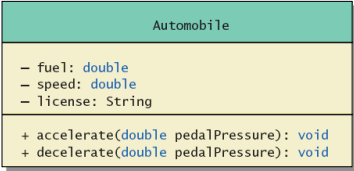
Third Instantiation:
Object name: ronsCar
amount of fuel: 2 gallons
speed: 75 miles per hour
license plate: "351 WLF"

Objects that are instantiations of the class **Automobile**

© 2008 Pearson Education, Inc., Walter Savitch and Frank Carrano 7

Class and Method Definitions

- A class outline as a UML class diagram



```
classDiagram
    class Automobile {
        - fuel: double
        - speed: double
        - license: String
        + accelerate(double pedalPressure): void
        + decelerate(double pedalPressure): void
    }
```

© 2008 Pearson Education, Inc., Walter Savitch and Frank Carrano 8

Example: Automobile Code

© 2008 Pearson Education, Inc., Walter Savitch and Frank Carrano 9

Attributes

- Attributes or instance variables are variables defined in a class (outside of a method)
- Each object of the class has a separate copy
- They live in memory for the life of the object
- They can be accessed from anywhere in the class

Methods

- Signature:
`<return type> <identifier>(<param list>) { }`
- Two kinds of Java methods
 - Return a single item, i.e. **return type**
 - No return type: a **void** method
- The method **main** is a **void** method
 - Invoked by the system
 - Not by the program

Primitive Types as Formal Method Parameters

- Parameters are a means of passing information from a caller of a method to the method itself
- A method can have zero or more parameters of different types
- Parameters are variables, they are local to a method
- Callers must provide the correct number of values/ types, automatic type conversion is carried out where appropriate

Object Analysis
From Problem Descriptions in Natural Language to Object-Oriented Designs

Look out for different parts of speech to obtain a first set of candidates for classes, attributes and methods:

- Nouns
Candidates for classes and attributes
- Verbs
Relations or behaviors (methods)
- Adjectives
Define or restrict ranges of values

© 2008 Pearson Education, Inc., Walter Savitch and Frank Carrano 13

Exercise: University Information System

Domain Description:

"Students have a first and a last name. Each student can be uniquely identified by his or her student number. The year of their first semester enrollment is recorded. This information is then used to report every year how long students remain at the UZH in average."

© 2008 Pearson Education, Inc., Walter Savitch and Frank Carrano 14

The Class String

- It is part of the Java class library, but it is not a primitive type.
- A value of type `String` is a sequence of characters treated as a single item.
- Strings are immutable

© 2008 Pearson Education, Inc., Walter Savitch and Frank Carrano 15

Declaring and Printing Strings

- **declaring**
`String greeting;`
`greeting = "Hello!";`
- **or**
`String greeting = "Hello!";`
- **or**
`String greeting = new String("Hello!");`
- **printing**
`System.out.println(greeting);`

Concatenation of Strings

- Two strings are *concatenated* using the + operator.
`String greeting = "Hello";`
`String sentence;`
`sentence = greeting + " officer";`
`System.out.println(sentence);`
- Any number of strings can be concatenated using the + operator.

Concatenating Strings and Integers

```
String solution;  
solution = "The temperature is " +  
72;  
System.out.println (solution);  
  
> The temperature is 72
```

String Methods

Method	Description	Example
<code>charAt(int index)</code>	Returns the character at the specified index.	<code>String str = "Java"; char c = str.charAt(1); // c is 'a'</code>
<code>concat(String str)</code>	Concatenates the specified string to the end of this string.	<code>String str = "Java"; String s = str.concat(" is fun."); // s is "Java is fun."</code>
<code>contains(CharSequence s)</code>	Tests if this string contains the specified sequence of characters.	<code>String str = "Java is fun." boolean b = str.contains("is fun"); // b is true</code>
<code>endsWith(String suffix)</code>	Tests if this string ends with the specified suffix.	<code>String str = "Java is fun." boolean b = str.endsWith("fun"); // b is true</code>
<code>indexOf(int ch)</code>	Returns the index of the first occurrence of the specified character.	<code>String str = "Java is fun." int i = str.indexOf('a'); // i is 1</code>
<code>indexOf(String str)</code>	Returns the index of the first occurrence of the specified substring.	<code>String str = "Java is fun." int i = str.indexOf("is fun"); // i is 4</code>
<code>length()</code>	Returns the length of this string.	<code>String str = "Java is fun." int l = str.length(); // l is 10</code>
<code>replace(CharSequence target, CharSequence replacement)</code>	Replaces all occurrences of the specified target sequence with the specified replacement sequence.	<code>String str = "Java is fun." String s = str.replace("is fun", " is great"); // s is "Java is great"</code>
<code>replaceAll(String regex, String replacement)</code>	Replaces all occurrences of the specified regular expression with the specified replacement.	<code>String str = "Java is fun." String s = str.replaceAll("is fun", " is great"); // s is "Java is great"</code>
<code>startsWith(String prefix)</code>	Tests if this string starts with the specified prefix.	<code>String str = "Java is fun." boolean b = str.startsWith("Java"); // b is true</code>
<code>trim()</code>	Returns a new string that is the same as this string but with leading and trailing spaces removed.	<code>String str = " Java is fun "; String s = str.trim(); // s is "Java is fun"</code>

© 2008 Pearson Education, Inc., Walter Savitch and Frank Carrano

The Method length()

- The method `length()` returns an `int`.
- You can use a call to method `length()` anywhere an `int` can be used.

```
int count = solution.length();
System.out.println(solution.length());
spaces = solution.length() + 3;
```

© 2008 Pearson Education, Inc., Walter Savitch and Frank Carrano

Positions in a String

- positions start with 0, not 1.
 - The 'J' in "Java is fun." is in position 0

© 2008 Pearson Education, Inc., Walter Savitch and Frank Carrano

Positions in a String, cont.


- A position is referred to as an *index*.
 - The 'f' in "Java is fun." is at index 9.

The twelve characters in the string "Java is fun." have indices 0 through 11. The index of each character is shown above it.

0	1	2	3	4	5	6	7	8	9	10	11
J	a	v	a		i	s		f	u	n	.

Note that the blanks and the period count as characters in the string.

Display 2.8
String Indices



© 2008 Pearson Education, Inc., Walter Savitch and Frank Carrano
22

(Not) Changing String Objects

- No methods allow you to change the value of a `String` object.
- But you can change the value of a `String` variable.

```

String pause = "  Hmm  ";           value of pause
pause = pause.trim();              Hmm
pause = pause + "mmm!";           Hmmmmm
pause = "Ahhh";                   Ahhh
    
```



© 2008 Pearson Education, Inc., Walter Savitch and Frank Carrano
23

Escape Characters

- How would you print "Java" refers to a language?
- The compiler needs to be told that the quotation marks (") do not signal the start or end of a string, but instead are to be printed.

```

System.out.println(
    "\\Java\\" refers to a language.");
    
```


© 2008 Pearson Education, Inc., Walter Savitch and Frank Carrano
24

Escape Characters

- \" Double quote.
- ' Single quote.
- \\ Backslash.
- \n New line. Go to the beginning of the next line.
- \r Carriage return. Go to the beginning of the current line.
- \t Tab. Add whitespace up to the next tab stop.

Display 2.10
Escape Characters

- Each escape sequence is a single character even though it is written with two symbols.

Examples

```
System.out.println("abc\\def");  
abc\def  
System.out.println("new\nline");  
new  
line  
char singleQuote = '\'';  
System.out.println(singleQuote);  
'
```
