Java Crash Course
Part II

Institut für Wirtschaftsinformatik
HU-Berlin WS 2004

Sebastian Kolbe
skolbe@wiwi.hu-berlin.de

Overview

- Repetition
- Control structures in Java
- About classes and objects
  - General concept
  - Implementation in Java

What you already should know

- How to log in, compile and start a Java application in the computer lab
- Java-Syntax
  - Variables and data types
  - Operators
  - Simple output of data

Control structures

- Control structures are for controlling the “program flow”. With these structures you can selectively execute program code based on some criteria or use the same code more than one time.
- Selective execution
  - If ... then ... else
- Loops
  - for
  - while
  - do

If/then/else

- Syntax (formal)
  - if (*expression*) *statement(s)*
  - if (*expression*) *statement(s)* else *statement(s)*
- Example in Java

```
{
   int i = 3;
   int j = 4;
   if (i < j) {
      System.out.println (“i is less than j!”);
   }
   else {
      System.out.println (“i is more than or equal to j!”);
   }
}
```

Loops

- Repeating some directives in program
  - for (initialization ; termination ; increment) statement(s)

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

  - while (boolean expression) statement(s)

```
int i = 0;
while (i < 5) {
   System.out.println(“i = “ + i);
   i++;
}
```

  - do statement(s) while (expression)

```
int i = 0;
do {
   System.out.println(“i = “ + i);
   i++;
} while (i < 5);
```

Factorial Calculator

```
/* This calculates the factorial of a given integer
 * result = number!
 *        = 1 * 2 * 3 * ... * number
 */

class Factorial {
    public static void main(String [] args) {
        // at first we declare some variables
        int number = 5;          // the input number
        int i = 1;               // "running" variable
        int result = 1;          // the output

        while (i <= number) {
                result = result * i;
                i++;
        }
        System.out.println("The factorial of " +
                           number + " is " + result);
    }
}
```

About classes and objects

- Understand classes as an prototype abstraction of a real world thing
- Classes defines behavior and capabilities common to all objects of a certain kind

- An object is a instance of a class!
- Objects have capabilities (defined in class) and a state

Real world example

- An electric kettle
  - Prototyp: A thing that can cook water
    - Capabilities: Can cook water; Can signalize state
    - Behavior: Uses electric energy, form
  - Object (the real kettle):
    - form, colour
    - State (on/off/defective, water level, …)
    - This object can cook water.
      BUT: From the user point of view it is needless to know how this object warms the water!
      -> *encapsulation*
      The kettle only shows its “user-interface” (switch, plug, lamp,...), other details are hidden.

Classes in Java

- Keyword: class
- Behavior and capabilities are expressed by variables and methods

```
class Name {
    // declare variables

    // Constructor(s), create an object

    // Method(s)

}
```

```
public class Calculator {
  private int result;
  Calculator () {
     result = 0;
  }
  public int sum(int a, int b) {
   result = a + b;
   return(result);
 }
}
```

Classes in Java

- Variables
  - <modifiers> datatyp name
  - public String myname;
- Methods:
  - <modifiers> datatyp name (Argumentlist)
  - private int getResult(int arg1, double arg2);
  - Contains statements and variables
  - Like a mathematical function
  - More than one method with the same name is possible, when using different argumentlist
  - Variables defined inside and noted in arguments are only locally available and usable

Modifiers

- static
  - Methods: Method can be called without creating an instance (object) of the class [-> main-method]
  - Variables: Variable can be used without initialization and contains the same value in all objects
- private, protected, public
  - Access rules for methods, variables and classes

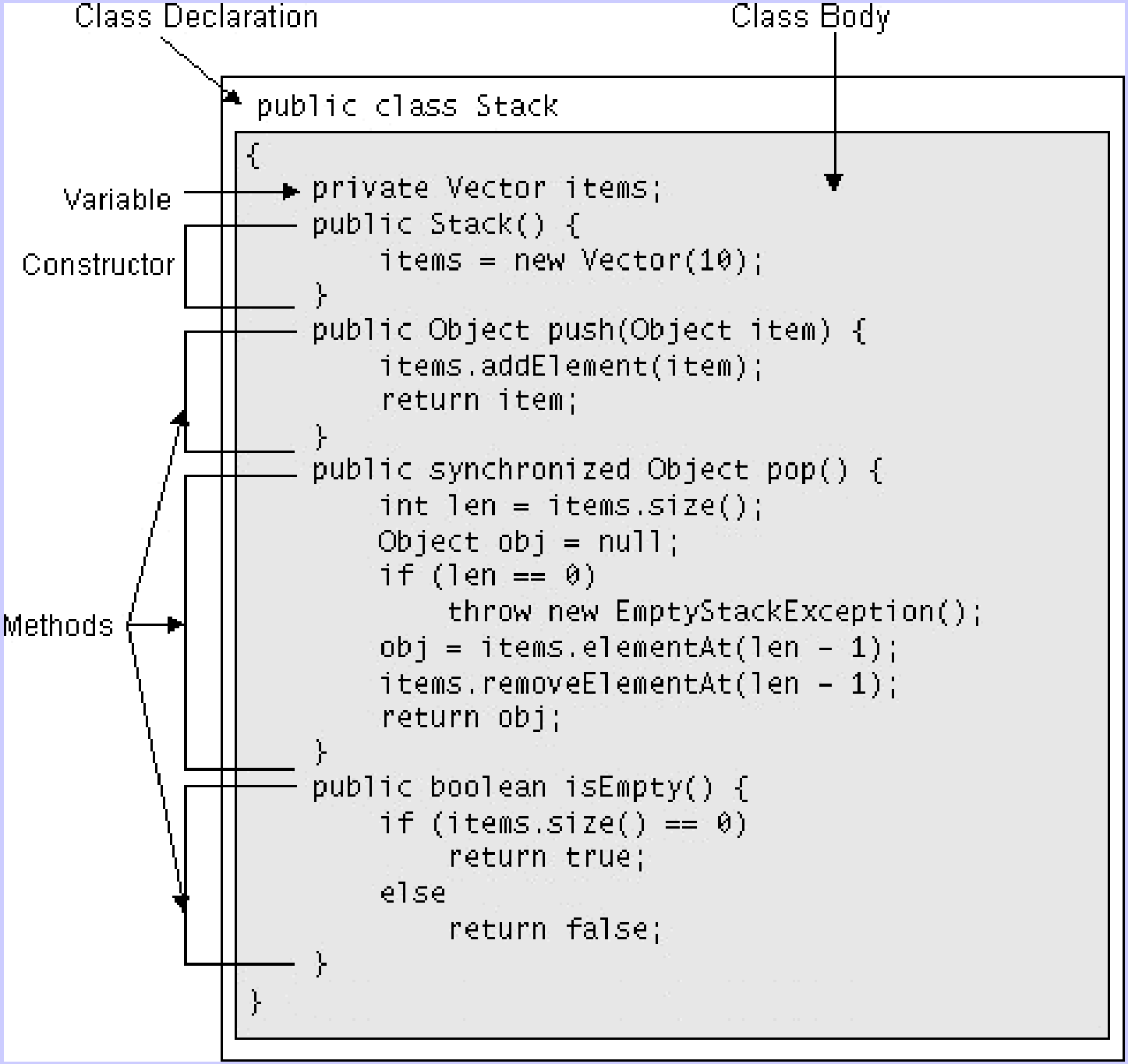
Specialized methods

- main
  - always a static method
  - the beginning of the program
- Constructor
  - creates (constructes) objects from classes
  - no return value (returns an object)
  - Call with the *new* operator

```
FactorialEnhanced facCalculator = new FactorialEnhanced(number);
```

Conventions

- Classes have the same name as file
- Classnames begin with a uppercase letter
- Methodnames begin with a lowercase letter
- Constructors use the same name as the class


Class Declaration
Class Body
public class Stack
{
Variable
private Vector items;
Constructor
public Stack() {
items = new Vector(10);
}
public Object push(Object item) {
items.addElement(item);
return item;
}
public synchronized Object pop() {
int len = items.size();
Object obj = null;
if (len == 0)
throw new EmptyStackException();
obj = items.elementAt(len - 1);
items.removeElementAt(len - 1);
return obj;
}
Methods
public boolean isEmpty() {
if (items.size() == 0)
return true;
else
return false;
}
}