Java Crash Course
Part II

School of Business and Economics
Institute of Information Systems
HU-Berlin WS 2005

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 (*boolean expression*) *statement(s)*
  - if (*boolean 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 and reusing 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 );
```

Summation Calculator

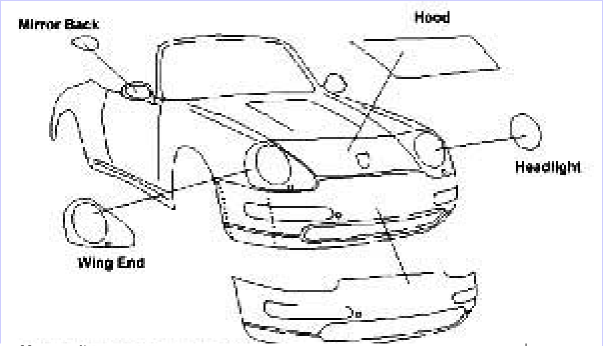
```
/* This calculates the summation of a given integer
 * result = 1 + 2 + 3 + ... + number
 *        = n * (n+1) * 0.5  (Gauss sum)
 */

class SumClass {
    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 = 0;         // the output

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

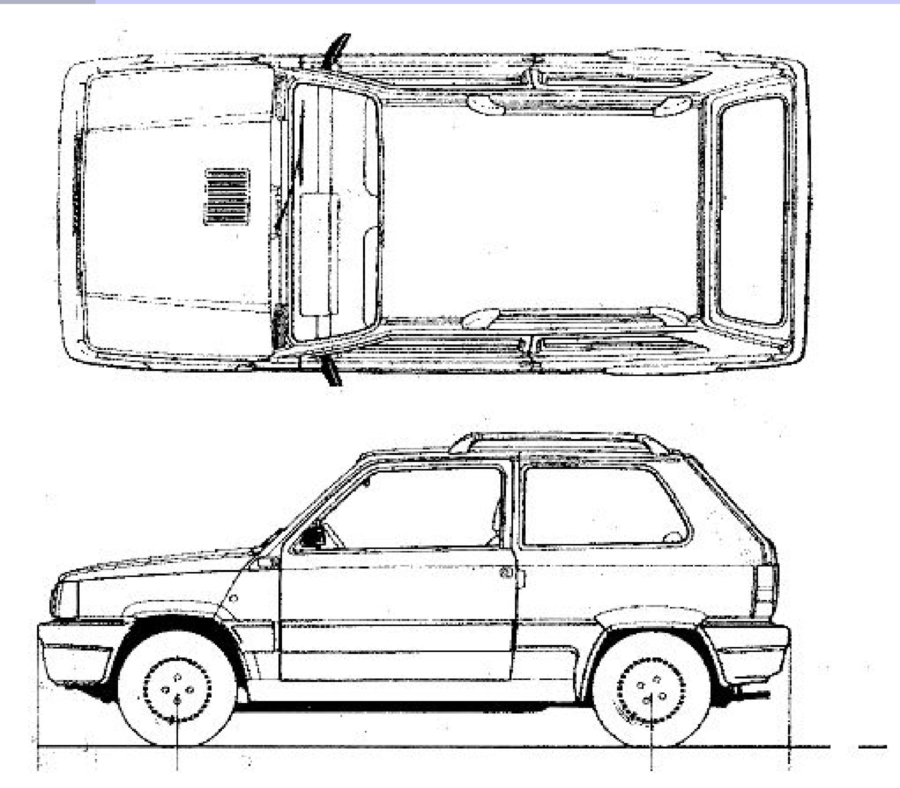
About classes and objects

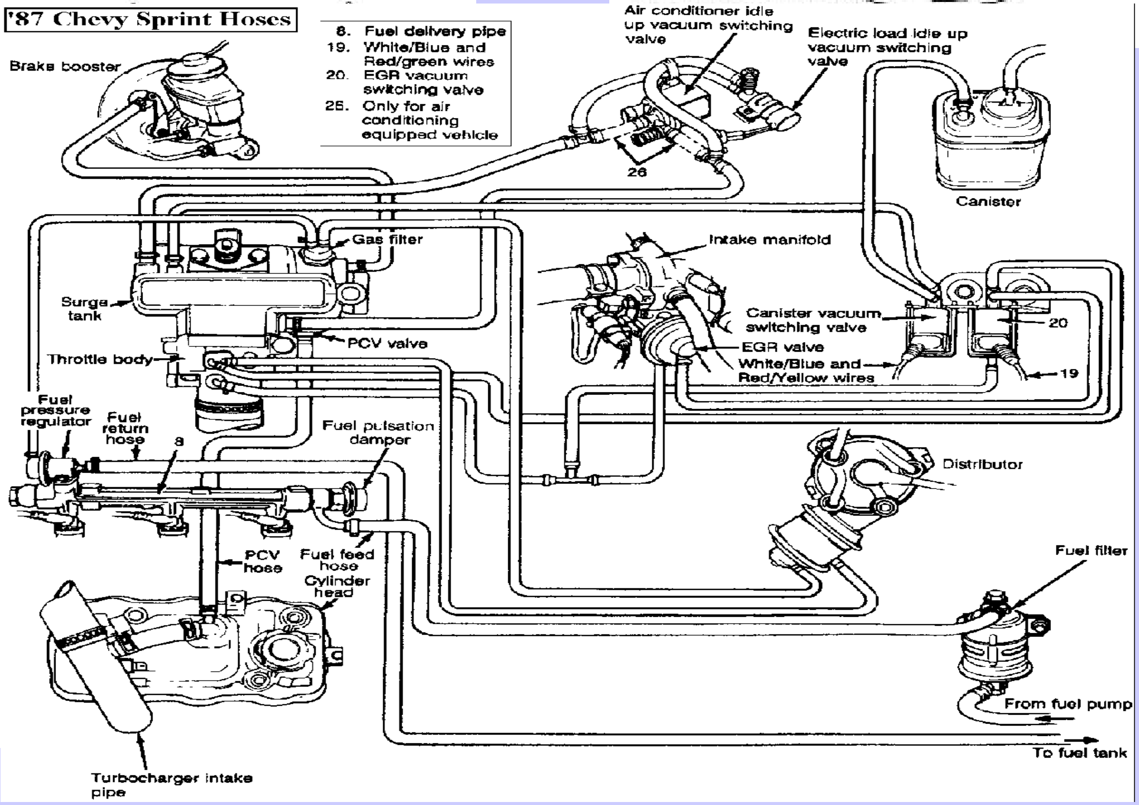
- Understand classes as an prototype abstraction of a real world thing
- Classes defines behavior and capabilities common to all objects of a certain kind
  - *The concept of classes are pure virtual! Like a blueprint or schematic diagram*
- An object is a instance of a class
  - Realization (or building) of an arcticle based on a blueprint
- Objects have capabilities (defined in class) and a state

Example: A Car

- The idea:
  - A vehicle for moving fast, comfortable, ...
  - Usable for transport, traveling, ...
  - Nice “looking”
- After 150 years of invention:







Example: A (virtual) Car

- The schematic design determines:
  - Properties (transport capacity, design, velocity, ...)
  - Behavioral Possibilities (Oil temperature display, headlight on/off, breaks, ...)

- But what is not determined?
  - Color
  - Cargo
  - Configuration
  - ...

Example: A (real) Car

Now build a “real” car from blueprint

Example: A (real) Car

- The “real” car:
  - Has all capabilities and behaviors from scheme
  - Additional states:
    - Characteristics (***persistent state***):
      (color, configuration of seats, roof, etc.)
    - ***Transient state:***
      (fuel, water, clean/dirty, broken, in use, lights on/off, ...)
  - From a users point of view
    - The car only “shows” its “user-interface” (steering wheel, lamps, knobs, buttons, ...) other functions are hidden!
    - The internal function of a car is mostly unknown to the driver (***opaque design***, ***encapsulation***)

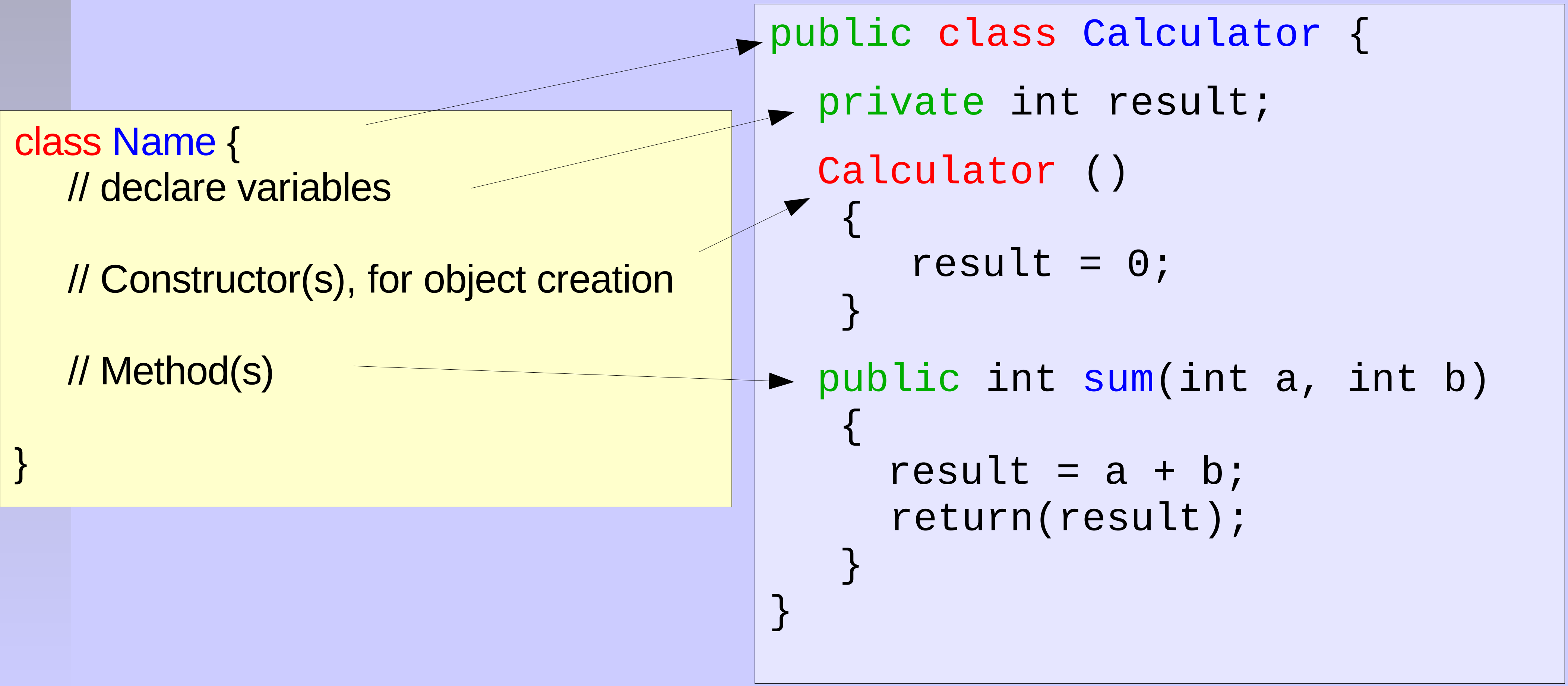
OOP

- Back to classes and objects:
  - A ***class*** can be understood as abstract view of an article/thing (a blueprint or schematic diagram)
    In IT: a module of a computer program that has a specific, separated functionality

  - An ***object*** is the article/thing itself built on the basis of a class.
    For every object a corresponding class exists!
    But you can have/create any number of objects from a given class
  - An object is also called an ***instance*** of a class

Classes in Java

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



```
class Name {
    // declare variables

    // Constructor(s), for object creation

    // 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

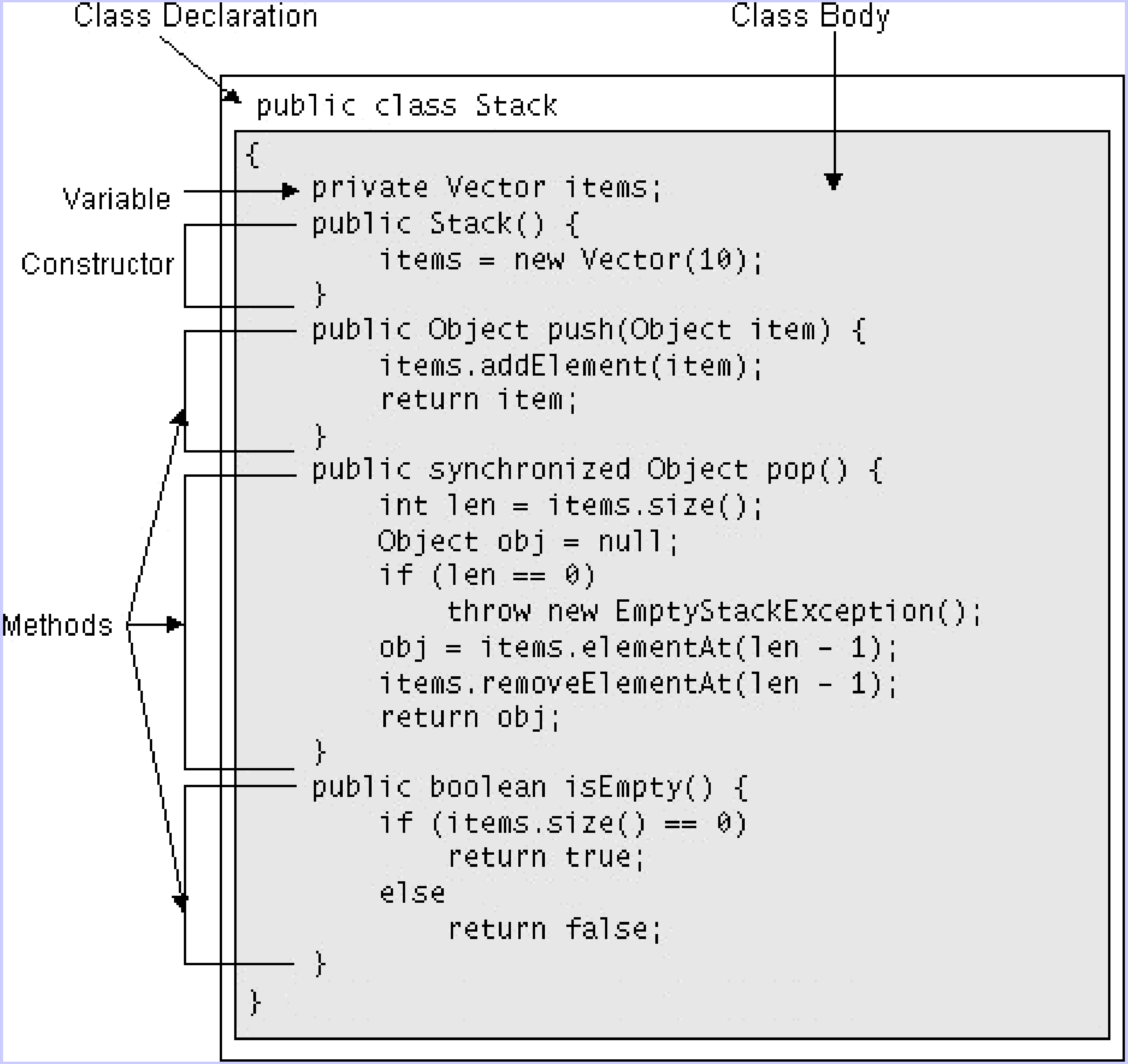
```
FactorialEnhanced facCalculator = new FactorialEnhanced(number);
```

Remember the syntax for arrays?

Conventions

- Classes have the same name as file
- Class-names begin with an uppercase letter
- Method-names begin with a lowercase letter
- Variable-names begin with a lowercase letter
- Constructors always use the same name as the class

- **Use comments and indentation!!!**



```
public class Stack
{
     private Vector items;
     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;
     }
     public boolean isEmpty() {
         if (items.size() == 0)
             return true;
         else
             return false;
     }
}
```

Homework

- Calculate the faculty (n!) of a number given on command line

```
> java  MyFacultyCalc  10
3628800
>
```

- Extend your program that it creates an instance of a class and uses it

```
/* Code Example */
public class MyFacultyCalc {
   // ...
   public static void main ( String[] args )  {
      MyFacultyCalc mfc = new MyFacultyCalc();
      // ...
   }
}
```