A short introduction to JAVA

LEGO Mindstorms @ Chair Brauer

Matthias Kranz
kranzm@in.tum.de

TU München
Foreword
Notation

- Java keywords are printed like this
- Complete Programs are printed like
  this
- Program names are printed - you get it - like
  this
- \n  stands for ’you press enter here’
Introduction
Objects

- An object is a software bundle of variables and related objects.
- An object encapsulates all methods for manipulating the object and its data.
- An object controls access to its variable and data (information hiding).
- An object can be extended simply by adding new methods and variables (modularity).
Classes

A class is a blueprint or prototype that defines the variables and methods common to all objects of a certain kind.

A class can contain definitions of variables and methods that exist once for all object instances (static keyword).

A class can contain definitions of variables and methods that exist once for each object instance.

A class can contain constructors (’method’ for building (and initializing) the object).
Objects and Classes

- class: blueprint of an object
- object: instance of a class
- objects are defined in terms of classes
Inheritance

- properties (variables and methods) of one object are inherited

- in java: only direct inheritance from one superclass (-> sometimes need for interfaces, e.g. Runnable)

- generally: multiple inheritance possible in other object-oriented languages
Example Inheritance

Figure 1: Example: (Single) Inheritance UML class diagram
Basics
Basics - (Primitive) Data Types

- byte, short, int, long
- float, double
- char
- boolean

Note 1: Primitive data types are not subclasses of java.lang.Object (e.g. must be casted to be added to a Vector)

Note 2: All other data types are reference data types
Basics - Variable Names

Legal variable names

- are composed of a series of unicode characters
- must not be java keywords or boolean literals (true and false)
- must not have the same name as other variables which are declared in the same scope
- must not begin with numbers (e.g. 12a)
Basics - Variable Scope

- Member variable
- Local variable
- Method parameter
- Exception-handler parameter
Example Variable Scope

```java
class SomeClass {
    ...
    member variable declaration ...
    ...
    public void someMethod(method parameters) {
        ...
        local variable declaration ...
        ...
        try {
            ...
        } catch (exception handler parameters) {
            ...
            ...
        }
    }
}
```

Figure 2: Example: Variable scope in java
Class and instance variables are member variables.

Class variables are accompanied by the `static` keyword.

Class variables exist once for all objects of this class.

Instance variables: no `static` keyword.

Instance variables exist once for all objects of this class.
Expressions and Control Flow
An expression is a series of variables, operators and method calls that evaluates to a single value.

- `a = 76;
- `b++;
- `a + b / 100;
- `true
Control Flow - Branching

- if (expression) {statement(s)}
  else if (expression) {statement(s)}
  else {statement(s)}

- int var;
  switch (var) {
    case value1: {statement(s)}; break;
    ...
    case valueN: {statement(s)}; break;
    default: {statement(s)}
  }
Control Flow - Looping

- **while** (expression) {statement(s)}
- **do** {statement(s)} **while** (expression);
- **for** (initialization; termination; [in|de]crement) {statement(s)}
- **break**; **label**, **break** label, **continue** label
The First Java Program
Main-Method

- The main method is the entry point for every executable java file (applets are different!).

- public static void main (String[] args) {
  statement(s)}

- At most one main method per class
**Java Tools**

- **javac**: compiles a java file (.java)
- **java**: runs a class file (.class)
- **javadoc**: generates documentation from comments and special tags in source files (.java)
- **jar**: generates archives of java files (e.g. to put multiple java files in class path)
- **jad**: java decompiler (not part of the Java Development KIT (JDK)/ Java Runtime Environment (JRE)!)
public class HelloWorld {
    public static void main (String[] argvs) {
        System.out.println("Hello World!");
    }
}
Compilation and Execution

- compilation:
  
javac HelloWorld.java

(with extension .java)

- execution:
  
java HelloWorld

(without extension .class)
Advanced Topics
Exception Handling

- Exceptions need to be declared to be thrown or caught
- Throwing of exceptions by using the throws-clause
- Exception catching using the try-catch-finally statements
public class ExceptionHandlingExample{
    static void somethingThatCanGoTerriblyWrong() throws AnyException {
        // code that can cause an AnyException to be thrown
    }

    public static void main(String[] argvs) {
        try{
            somethingThatCanGoTerriblyWrong();
        }
        catch (AnyException anyException) {
            // do some exception handling if necessary
        }
        finally {
            // execute the code in the finally part no matter
            // if any exception occurred or not
        }
    }
}
Import of Java-Classes I

- Loading of needed classes (e.g. java.lang.Math for sine and cosine)
- Import of specific classes using the full qualified class name (e.g. java.util.Vector)
- Import of complete packages using * as class name
- Import statements are the first statements in any class (before class declaration)
- Always prefer import of every (really) used class than importing whole packages
Examples:

- import java.util.Vector;
- import java.util.*;
Packages

- Packages help grouping related classes into groups
- Access control on package level (protected keyword)
- Example: `package testclasses;`
Abstract Classes I

- Abstract classes model abstract concepts :-)  
- Illustration: food - in the real world there are only instances of beef, beer and stuff but not of food  
- Abstract classes cannot be instantiated  
- Abstract classes can be subclassed  
- Abstract classes are made abstract by using the `abstract` keyword
Abstract Classes may contain normal methods (with method body) and abstract methods (without method body)

Method declaration as usual with addition **abstract** keyword after access declaration
Abstract Classes III

- Abstract classes may contain constructors
- Abstract classes may contain method declarations and method implementations
- Abstract classes may contain any sort of variables
- Abstract classes may only be subclassed, not instantiated
An interface is a named collection of method definitions (without implementations). An interface can also contain constant declarations.

Interfaces are made by using the interface keyword.
Interfaces II

- Interfaces may not contain implemented methods
- Interfaces may contain constant variables and method declarations
- Interfaces may be implemented by any classes that have not to be hierarchically connected
Interfaces and Inheritance

- Differences
  - Constructors
  - Method declarations and method implementations
  - Variables
- Use abstract classes to model abstract concepts
- Use interfaces to model (common) behavior
Bibliography
This presentation bases heavily on ’The Java Tutorial. Second Edition. Object-Oriented Programming for the Internet.’ (see next foil)

See /doc/java on our computer pool for java documentation

See http://java.sun.com for further reading
References

End