
Introduction to Java

Frédéric Haziza
(daz@it.uu.se)

Aug 31, 2004

Plan

- Where to get Java
- Launching a program
- Object instances
- Classpath
- Structures of Control
- Syntax
 - Variables
 - Methods
 - this, super
 - Constructors
- Scope
- API

Where to get Java

Your friend:
<http://java.sun.com>

Java 2 Editions

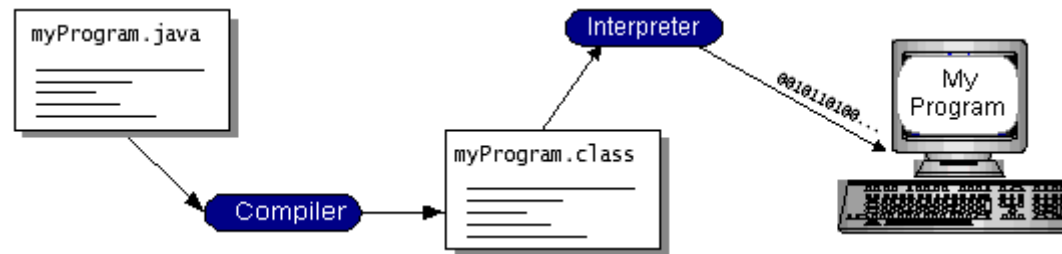
- J2SE : Standard Edition
 - Suitable for 'usual' Desktops
- J2EE : Enterprise Edition
 - Suitable for Servers
- J2ME : Minimal Edition
 - Suitable for Mobile

Java 2 SDK

- Standard Development Kit
 - Latest stable version : 1.4.2
 - <http://java.sun.com/j2se/1.4.2/download.html>
- JRE : Java Runtime Environment (java command)
- Compilation (javac command)

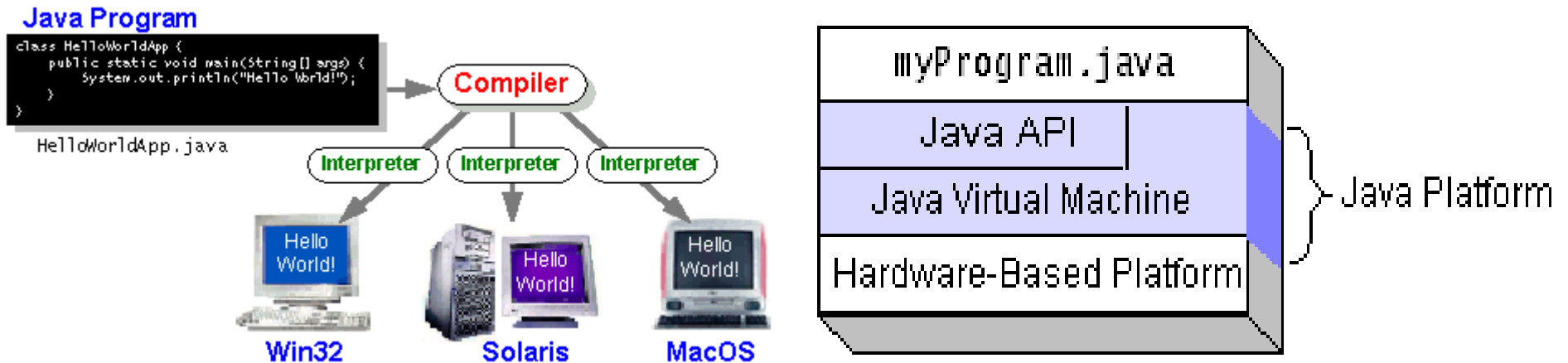
The Java Programming Language

- Object-Oriented
- See :
<http://java.sun.com/docs/books/tutorial/java/concepts/index.html>
- Compilation is necessary
- Compilation = Translation of human readable code into machine code



Java Virtual Machine

- JVM : Java Virtual Machine
- Bytecode
- “Written once, Runnable everywhere”



Notions of Type and Object

Cooking an egg

- 1. Name of the recipe : Cooking egg
- 2. The materials :
 - A pot (No holes and empty)
 - Water (sufficiently, no nitrates)
 - A stove (Heat power mini 1000 Watts)
 - an egg (Not rotten)
- 3. Actions :
 - Put water in the pot
 - Boil the water
 - Put the egg in the water
 - Wait 3 minutes
 - Take out the egg from the water

Algorithm

- Algorithm :
 - 1. Name
 - 2. Data, which are used in the algorithm
 - 3. Behaviour, description of the chained actions
- The person knows what actions are possible with the specified objects
 - Put the pot in the egg,
 - boil the egg,
 - cut the pot in small piece.
- The pot can be in different states : empty, full of water or milk, to a specific temperature
- All pots have properties that distinguish them from oven.

==> The item pot is of type "pot", or belongs to the class "pot"

Type

- In algorithm, all items have a type. This type combines:
 - a set of values/attributes/states that this item can have/take
 - a set of actions, which can be applied to those item of that type, allowing to get/set/modify the values.
- Because the operation "cut in small pieces" hasn't been defined for the type "pot", it is impossible, and even forbidden, to cut a pot in small pieces.

Predefined types

- int, long
 - float, double
 - boolean
 - char
 - byte
-
- Self contained type
 - Determines the set of value it can have

Primitive Type Example

- For byte, from -128 to 127, inclusive (8bits)
- For short, from -32768 to 32767, inclusive (16 bits)
- For int, from -2147483648 to 2147483647, inclusive (32 bits)
- For long, from -9223372036854775808 to 9223372036854775807, inclusive (64 bits)
- For char, from '\u0000' to '\uffff' inclusive, that is, from 0 to 65535 (16 bits -- unsigned)

Literal	Data Type	Literal	Data Type
178	int	26.77e3	double
8864L	long	'c'	char
37.266	double	true	boolean
37.266D	double	false	boolean
87.363F	float		

Launching

Main

- `public static void main(String[] arg){ }`
- **Compilation**
 - `prompt>javac MyExample.java`
- **Execution**
 - `prompt>java MyExample`
 - `prompt>java MyExample param1 param2`

Examples

“Hello World”
and
“Print arguments”

Object - Instance

Object - Instance

- A class is a module that regroups several actions and encapsulates data, but it's also a type, defined by the user
- Once the class is declared, we can create instances of it, the same way we declare a variable of a primitive type

```
int i;
```

```
int j,k;
```

```
BankAccount myAccount, yourAccount;
```

Reference - Instanciación

- `myAccount` and `yourAccount` are not objects but references to objects. Those objects are not created yet.
- They point to nothing, they have the value "null"
- By using the operator 'new', we create an instance (and allocate the necessary memory space).
- The result of that operation can be appointed to the `myAccount` reference.

==> `myAccount = new BankAccount();`

- This instance can be designated by other references:

==> `yourAccount = myAccount;`

Null - Affectation

See blackboard



Once the instance is created we can pass messages to it, meaning we can call the methods that it embodies. It is important that this reference designates an instance before we pass messages onto it.

(it will result a **NullPointerException** which stops the JVM)

Classpath – File Naming

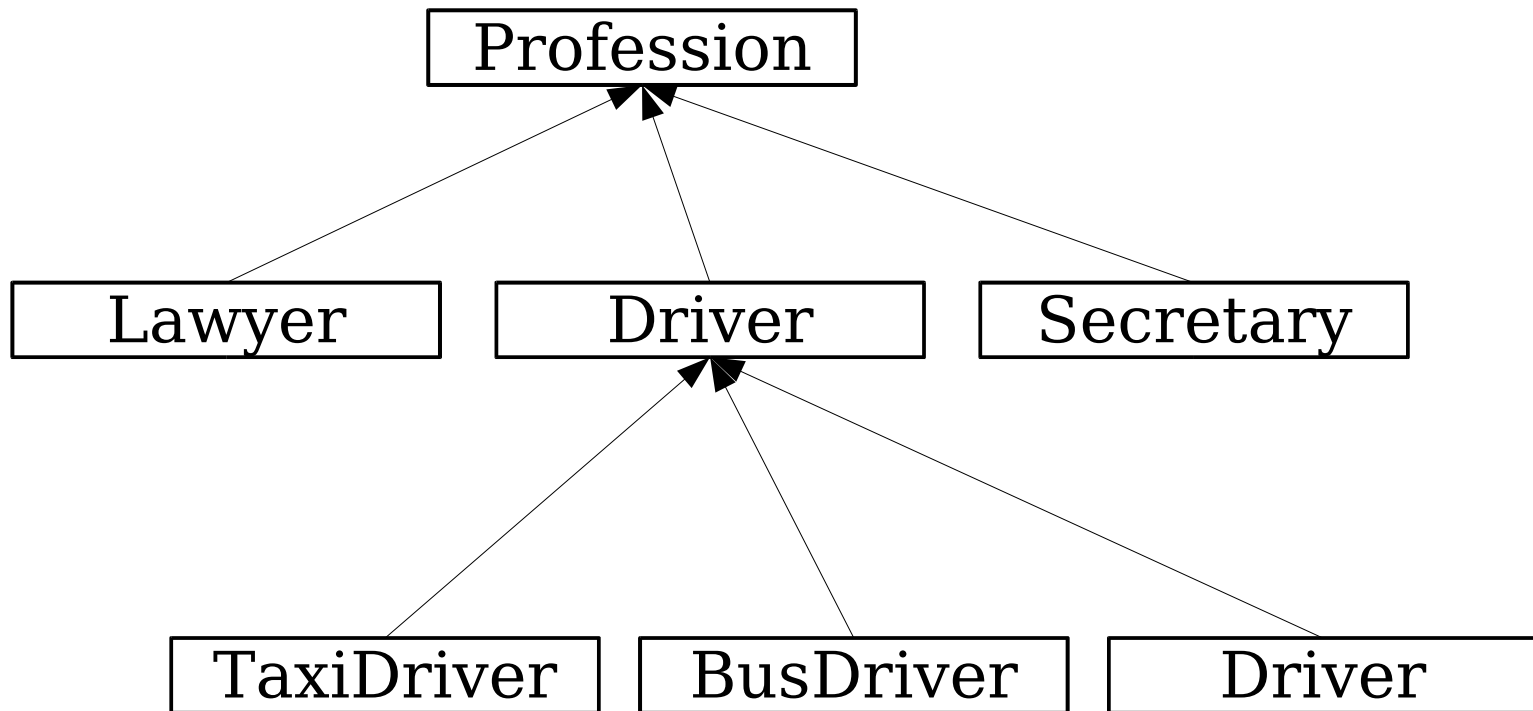
Where from?

- Call the BankAccount class :
where does it the class definition come from?
 - The JVM detects a BankAccount type, so looks for a file named “BankAccount.class” under the classpath.
- If your classpath doesn't point to the BankAccount class definition, the class you're currently writing won't compile.
- The JVM doesn't go down the folder tree, you must simply add folders to the classpath.

==> **prompt>CLASSPATH=folder1:folder2:folder3**

- You can specify the '.' folder into your classpath
 - the JVM will examine the current folder.
- It will parse the classpath in order.
- Example with “Person And Name”

Inheritance



Inheritance (cont'd)

- TaxiDriver inherits from the Driver class. A TaxiDriver is a Driver too.
- class Driver {...}
- class TaxiDriver extends Driver{...}
- TaxiDriver specializes the Driver class.

Structures of Control

Structures of Control

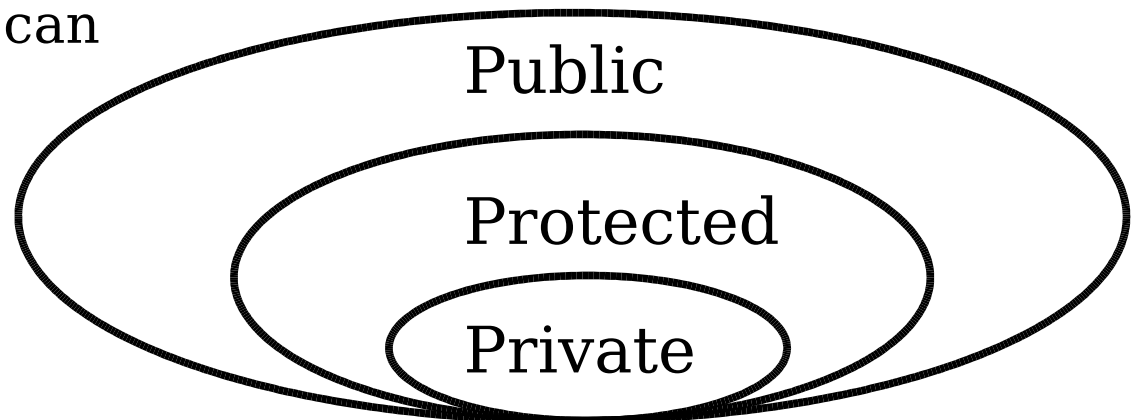
- if statement
- for loop
- while loop
- switch conditional
- return statement
- break statement

==> See your course book !!

Java Syntax

Visibility

- 4 access modifiers
 - Private, protected, public or 'nothing'
(you can see 'friendly' sometimes)
- **Public:** all class can access the attribute / call the method
- **Private:** only the current class can
- **Protected :** children can



Method declaration

- Signature = Combination of return value, name, parameters list and access modifiers
- `public static void main(String[] arg){...}`
- `private static void main(String arg){...}`

==> `<AccessModifier> <Return> <Name>(<Param List>){...}`

Variable declaration

- `int i; // Declaration`
- `i=17; // Affectation`
 - `int i = 17; // Both`
- `int i = 12; // Illegal, i is already defined`

- `int[] myArrayList = new int[3];`
(Collection interface, List interface ||
Vector, Array, array list, LinkedList...)

this and super

- Use the keyword '**this**' to refer to the **current** class
- Use the keyword '**super**' to refer to the **parent** class

Constructors

- Initialization of the class
- Constructor with parameters
- Java Default constructor

==> See blackboard !!

Scope

Global or local ?

- Scope of variable declaration : **global** or **local**
- Example of code

```
if(...)
{
    int i= 17;
}
System.out.println("Here is i="+i); // Error
```

==> “Live” example ...

Import and API

- If you want to reuse code from other class, the user must supply an additional line of code

```
import java.util.Vector;  
import java.util.LinkedList;  
...
```

- <http://java.sun.com/j2se/1.4.2/docs/api/index.html>
- See Sun's API Example