Introduction to Java

Frédéric Haziza
(daz@it.uu.se)

Summer 2005

Outline

- Where to get Java
- Compilation
- Notions of Type
- First Program
- Java Syntax
- Scope – Class example
- Classpath
- Object - Instance
- API

- Where to get Java
  - Compilation
  - Notions of Type
  - First Program
  - Java Syntax
  - Scope – Class example
  - Classpath
  - Object - Instance
  - API

Language Evolution

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

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

Java 2 SDK

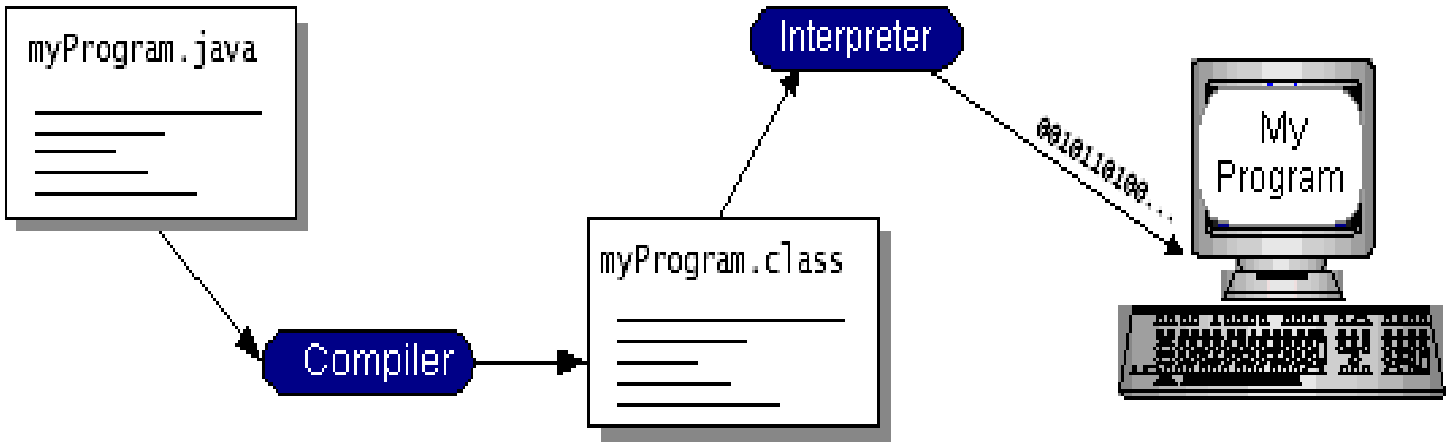
- Standard Development Kit
  - Latest stable version : 1.4.2 and 5.0
  - http://java.sun.com/j2se/index.jsp

- JRE : Java Runtime Environment (java command)
  - Able to run Java programs only
- Compilation (javac command)
  - Comes with the SDK, able to build and run Java programs

- Where to get Java
- Compilation
  - Notions of Type
  - First Program
  - Java Syntax
  - Scope – Class example
  - Classpath
  - Object - Instance
  - API

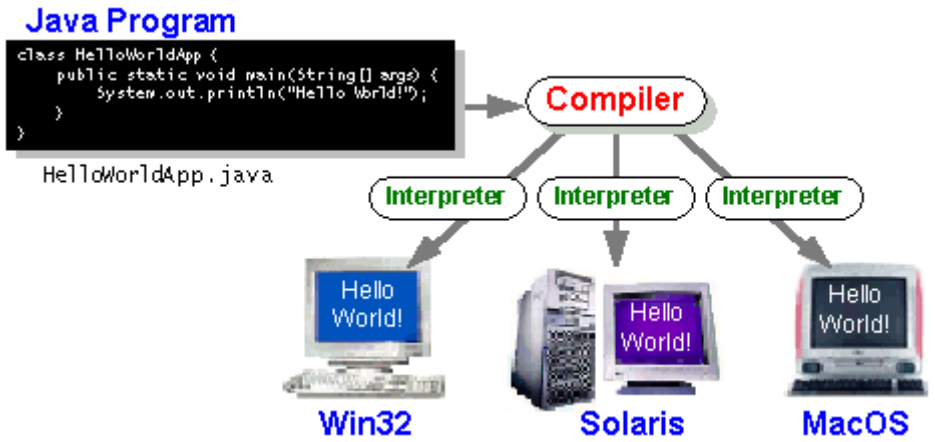
The Java Programming Language

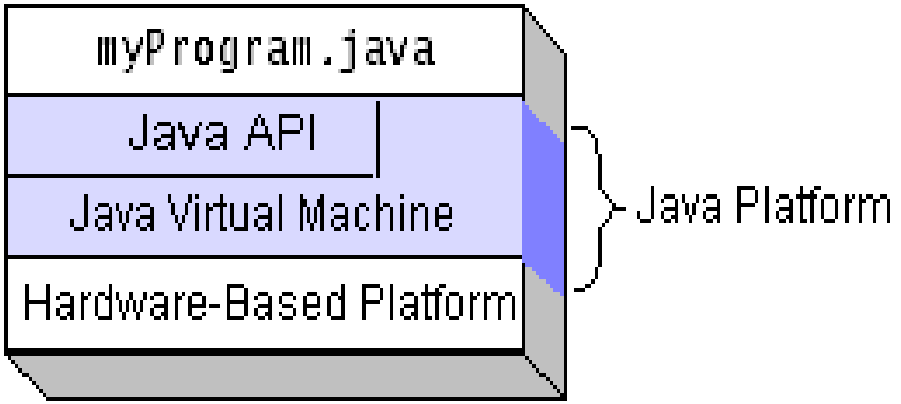
- Object-Oriented
- See :
  http://java.sun.com/docs/books/tutorial/java/concepts/index.html
- Compilation is necessary
- Compilation = Translation of human readable code into machine code



Java Virtual Machine

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





- Where to get Java
- Compilation
- Notions of Type
  - First Program
  - Java Syntax
  - Scope – Class example
  - Classpath
  - Object - Instance
  - API

Cooking an egg

- 1. Name of the recipe : Cooking egg
- 2. The materials :
  - A pot (No holes and empty)
  - Water (sufficiantly, no nitrates)
  - A stove (Heat power mini 1000 Watts)
  - an egg (Not roten)
- 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 algorithmics, 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, short
- float, double
- boolean
- char
- byte
- void

- 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		

Strings

- Character chains
- Special class in Java, just for our convenience :)
- String str1 = “Salut”;
- String str2 = “comment ça va?”;
- String str3 = str1 + “, ” + str2;
- str3 ----> Salut, comment ça va?

Type Conversion

- Type conversion = change from one type to another
- byte i; (max of 127)
- int j = 468;
- i= j;
- Value of i ?
  - 468 in binary is 111010100
  - Depending on the system:

Type Cast

- Type cast = identify as belonging to a certain type
- int i; short j; float k;
- i = 468;
- j= (short)i; (No problem, 468 spans within the range)
- k = 3.2;
- i = (int)k; NB: i is 3 only!
- String str = “3”; i = str3; ==> Cast error
- Idem for classes (cf. later)
- Conclusion:
  - Not always possible, could lead to mishandlings
  - Permits to view temporarily a typed variable as another type

- Where to get Java
- Compilation
- Notions of Type
- First Program
  - Java Syntax
  - Scope – Class example
  - Classpath
  - Object - Instance
  - API

“Hello World!” example

```
class FirstProgram{


   public static   void   main (String[] args)
         {

            System.out.println(“Hello World!”);

         }


}
```

“Hello World!” revisited

```
class SecondProgram {

   public static void main(String[] args)
   {
      if(args.length>0)
      {
         System.out.println(“Welcome to the real world, ” + args[0]);
      }
      else
      {
         System.out.println(“No arguments, welcome anyway!”);
      }
   }
}
```

- Where to get Java
- Compilation
- Notions of Type
- First Program
- Java Syntax
  - Scope – Class example
  - Classpath
  - Object - Instance
  - API

Java Syntax

- Statements are followed by a semi-colon
  - System.out.println(“Hello World!”);
- Blocks are embraced in curly braces {}
- Parameters are supplied within brackets ()
- List items are accessed through []
  - List[3] is the 4th item in the list, if it exists…

Identifiers

- Examples:
  - String i3 MAX_VALUE isLetterOrDigit
- No start with a digit
- No start with ?:';^(\)”<>$&|-+/*%!
- Keywords
  - abstract default if private this boolean do implements protected throw break double import public throws byte else instanceof return transient case extends int short try catch final interface static void char finally long strictfp volatile class float native super while const for new switch continue goto package synchronized

Comments

- There are two kinds of comments:
  - /* text */ all the text in between is ignored
  - // text all the text to the end of the line is ignored
- Comments do not nest
- Special comments: Javadoc comments
  - /** javadoc comments */

Operators

= > < ! ~ ? :

== <= >= != && || ++ --

+ - * / & | ^ % << >> >>>

+= -= *= /= &= |= ^= %= <<= >>= >>>=

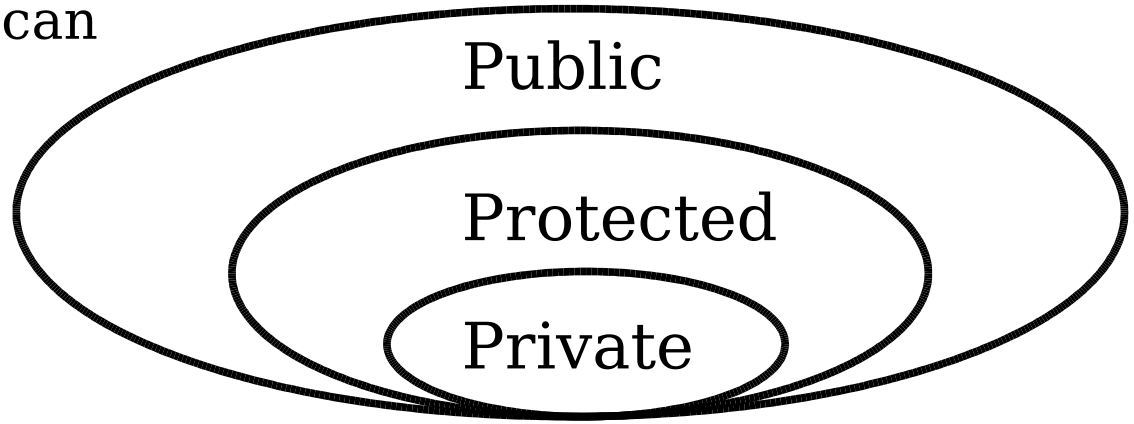
Coding Conventions

- Class names start with a capital letter
- Method and variable names start with a small letter
- If the name is composed of several words (to give some sort of sens to it), each word begins with a capital letter, no space, no underscore
  - extractSomePetrolFromTheGround(Technic t){}
  - BankAccount familyAccount;
- Indentation : “SimpleScreenManager.java”

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){…}

==> <AccessModifiers…> <Return> <Name>(<Typed Param…>){…}

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];
- int[][] myMatrix = new int[2][3];
- BankAccount[] myArrayListOfAccount = new BankAccount[5];

- Where to get Java
- Compilation
- Notions of Type
- First Program
- Java Syntax
  - Scope – Class example
- Classpath
- Object - Instance
- API

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
```

The class Person

- Implement an entity that represents a “Person”
- Properties :
  - Name, P-nummer
  - Address, City
  - Phone number
- Requirements :
  - Can have several phone numbers
  - Can change address, city, phone numbers
  - Implement getters and setters
  - Return the age of the person

The class Person

- "Person"
- Properties :
  - Name, P-nummer
  - Address, City
  - ~~Phone number~~
- Requirements :
  - ~~Phone numbers~~
  - Change ~~address~~, city, ~~phone numbers~~
  - getters/setters
  - ~~age of the person~~

```
public class Person {

      private String name;
      private String address;
      private String city;
      private String pNum;
      private int[] phoneNumbers=new int[5];

      public Person(String aName, String anAddress,
                        String aCity, String aPNum)
      {
          name = aName;
          address = anAddress;
          city = aCity;
          pNum = aPNum;
      }

      public String getCity()
      {
          return city;
      }

      public void setCity(String aCity)
      {
          city=aCity;
      }
}
```

Does it fullfil the requirements?

No comments!

- Where to get Java
- Compilation
- Notions of Type
- First Program
- Java Syntax
- Scope – Class example
- Classpath – File Naming
- Object - Instance
- API

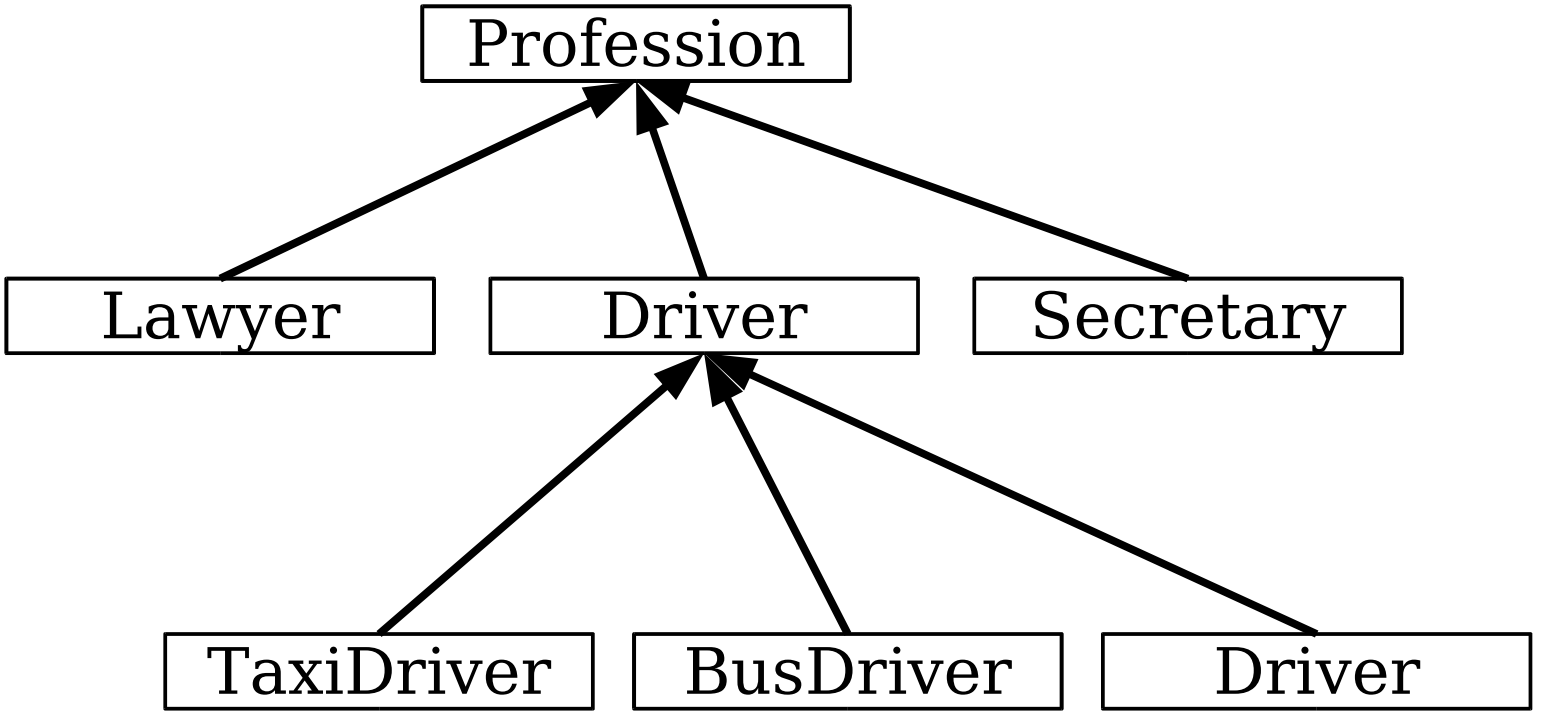
Where from?

- 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 writting 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.

Inheritance



Note : No multi-inheritance...

Inheritance (cont'd)

- TaxiDriver inherites from the Driver class.
  A TaxiDriver is a Driver too.

- class Driver {...}
- class TaxiDriver extends Driver{...}

- TaxiDriver specializes the Driver class.

Extending the Person class

- Recall the Person class
- Extend it with some extra fields and methods
- ==> ExtendedPerson.java specializes Person.java
- See the implementation and code choices

- Where to get Java
- Compilation
- Notions of Type
- First Program
- Java Syntax
- Scope – Class example
- Classpath – File Naming
- Object - Instance
- API

Object - Instance

- A class is a module that regroups several actions and encapsulates data
- Also a user-defined type
- 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 - Instanciation

- 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)

Scope (cont'd)

- Recall that a variable can be global (to the class) or local (to the block, typically the method)
- An object will stay “alive” as long as a reference points to it.
  (No need to clean up though: Garbage Collection)
- Important:
  BankAccount a = new BankAccount();
  BankAccount b = a;
  changeSomethingInBankAccount(a);
  ==> b has also changed !
  b was just another reference to the same object
- For more information: See cloning and deep-cloning

- Where to get Java
- Compilation
- Notions of Type
- First Program
- Java Syntax
- Scope – Class example
- Classpath
- Object - Instance
- Import - API

Import and API

- API = Application Programmable Interface

- 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;
  import java.util.Calendar;
  ...
  ```

- http://java.sun.com/j2se/1.4.2/docs/api/index.html