Module 1. Foundations of Java

1.1 History

s2

History

- Java evolved from C++
- C was developed by Dennis Ritchie & Kernighan at Bell Labs in the 70s
- C++ is a superset of C, OO
- Developed by Stroustrup at Bell Labs in the 80s
- Most OS are developed in either C or C++

s2

History

- Java is currently the most used language
- Java developed by James Gosling at Sun in the 90s
- Initially called Oak and conceived as language for intelligent electronic devices (appliances)

History

- C or C++ code is compiled for a chip.
  - If the chip changes, the code changes
- Java didn't succeed in this area because of the slow market growth of such electronic devices
- The rise of the WWW made Java succeed as a language to develop Web systems

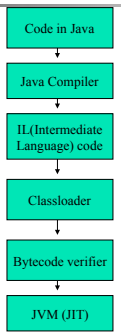
Performance

- In some cases Java can outperfom C++
- In other cases C++ outperforms Java
- It is also the case in which performance is similar
- A comparative benchmark can be checked out at [keith 03]
- Some claim that C++ can be superior if compilation is optimised [Tudor 10]

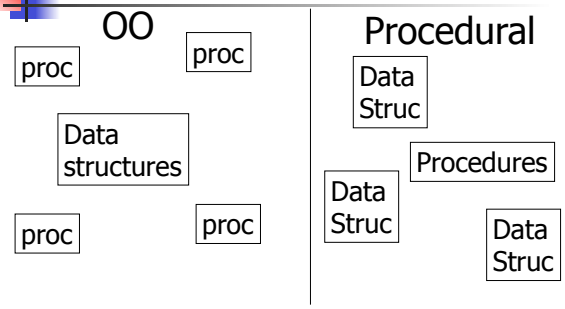
Performance

- Java advantages
  - The JIT compiler can better optimise compilation
  - Garbage Collection mechanism can speed new memory access
- Java drawbacks
  - Startup takes longer (class loading)
  - Compilation to native code at runtime
  - Pointers are not supported
  - Not good for Hardware dependent apps (driver, video game, etc)

Development Environment



OO vs Procedural



s2

1.2 Data types, operators and expressions

Source Files

- The extension of all java source files is *.java*
- Each file can contain at most one public class and the public class name must match the filename
- Each file can contain unlimited non-public classes
- Three top level elements known as compilation elements appear in this order:
  - Package declaration
  - Import statements
  - Class, interface or enum definition

Source Files

```
package myjava.basic;
import java.util.*;
import java.awt.Frame;

public class MainClass {
}

class subClass{
}
```

Comments

- Old style
  - /* this is a commented line*/
  - May include multiple lines
- New style
  - // this is a commented line
  - Only includes one line
- Documentation style
  - /** documentation comment */

s2

Comments

- The multiple lines comment is treated as a block of comments. So the code below works just fine

```
System.out.println(/* comment */ x);
System.out.println /*commnet */ (x);
System./* comment */out.println(x);
System/* comment */.out.println(x);
```

Block of Code

- The code statement is terminated with a semicolon (;). One statement can be written in multiple lines.
- The code below means the same

```
int sum = a+b+c;

int sum = a+
b+
c;
```

Block of code

- A block of code is bounded by braces ({…})
- The ; could either be added or omitted after the block of code

```
{
        sum=a+b;
        System.out.println(sum);
}

{
        sum=a+b;
        System.out.println(sum);
};
```

Keywords and Reserved Words

- A keyword is a word that cannot be used as an identifier
- A reserved word is one which is not in current use but it is not allowed to use as an identifier
  - There are two reserved words: goto and const

Keywords

abstract	continue	for	new	switch
assert***	default	goto*	package	synchronized
boolean	do	if	private	this
break	double	implements	protected	throw
byte	else	import	public	throws
case	enum****	instanceof	return	transient
catch	extends	int	short	try
char	final	interface	static	void
class	finally	long	strictfp**	volatile
const*	float	native	super	while

*	not used
**	added in 1.2
***	added in 1.4
****	added in 5.0

Keywords

- true, false and null are not keywords. They are literal values
- All keywords are only written in lowercase. Which one is a keyword?
  - instanceof, instanceOf

Identifiers

- Used to name variables, constants, functions, classes, etc.
- Restrictions:
  - Start with a letter, underscore (_) or dollar sign ($)
  - Can be composed of alphabetic character or digit, dollar sign and underscore.
  - It is case sensitive
  - There is no maximum length
  - It cannot be a keyword

s2

Identifiers

- Which ones are legal identifiers?
  - int _a;
  - int $c;
  - int e#;
  - int 7g;
  - int $this_is_illegal$$;
  - int this_is_a_very_long_name_as_identifier;
  - int @mail variable;
  - int _____2_f_W;
  - integer int_1;
  - int bytes;

Primitive Types

- A primitive type is a simple non-object data type that represents a single value
- There are eight primitive data types
  - Logical: boolean
  - Textual: char
  - Integral: byte, short, int, long
  - Floating point: float, double

s2

Data Types

Type	Bytes
boolean	?
byte	1
short	2
int	4
long	8
float	4
double	8
char	2

s2

Class variable, Instance Variable

- Variables defined outside a method but within a class are either a class variable or an instance variable
- Class variable is declared using static. It is created when the class is loaded. Can be manipulated by
  - All objects
  - Without creating an instance of the class

s2

Class variable, Instance Variable

- Instance variable or member variable is declared without static. It continues to exist for as long as the object exists
  - Are initialised automatically when created
  - If not explicitly initialised, they are assigned to default value

Class snapshot

```
Class DemoScope {
        int a;
        int b = 10;
        static int c = 10;
        void methodOne(int a) {
                int b=5;
        }
        void methodTwo(int a) {
                int b = 5;
                int c = 10;
        }
}
```

Instance variables

Class variable

Local variables

Class variable, Instance Variable

- You must always initialise local variables before use
- Class or instance variable is implicitly initialised to the default value, unless you explicitly initialise to other value

My first program

```
public class HelloWorld {
   public static void main( String args[] ){
      System.out.println("Hello World\n");
   }
}
```

s2

My first program

- Quiz: Use the command prompt to compile and run HelloWorld.java

Configuring Environment

- Linux
  - Edit .bash_profile or /etc/rc.local
    - CLASSPATH=".;"
    - export CLASSPATH
    - JAVA_HOME=/usr/java/jdk1.6.0
    - export JAVA_HOME

Configuring Environment

- Windows
  - Inicio->Mi PC->Propiedades->Opciones Avanzadas->Variables de Entorno
  - classpath: .;
  - JAVA_HOME: C:\Archivos de programa\Java\jdk1.5.0_22\
  - Path:C:\Archivos de programa\Java\jdk1.5.0_22\bin