Java

Just a quick reminder

The Java we will be using

- Java: J2SE 1.4.2
- Embedded Java means hardware-related restrictions (smallish windows, no menus, no scrolling, ...)
- Embedded Java means software-related restrictions (AWT only, deprecated methods, ...)

Basics

- Main Class
- public static void main(String [] args)
- System class
  - Input/Output
  - OS interactions (environment, librairies,...)
  - Garbage Collector

Magic

- The JVM is a compiled program
- Java bytecode is loaded on the fly
- JAR archives are in a pre-defined location

Magic (continued)

- The user specifies the main class (command line switch, Manifest, ...)
- Dynamic loading is based on the Class class
  - Identification: getName(), getSuperclass(), ...
  - Components : getFields(), getMethods(),...
  - New objects creation: newInstance()
- Retrieves the main method and executes

"Hello World 2"

- AWT Interface
  - Frame
  - Label
  - Button
- First version : "external" instanciation
  - new Frame
  - new elements
  - add to frame
  - return

Demo

Magic

- The OS provides the windows and other graphical items
- Default behavior (focus, movements, resizing, closing,...)
- As long as you have one visible window, and you don't call *System.exit(int v)*, the program is active

"Panic Button"

- A frame with a "quit" button
- External feedback
- Internal handling of events

Demo

Magic

- Frame-extending class
  - Default behavior different from system behavior (pre-loaded widgets, button actions, ...)
- MouseListener implementation
  - Handling the basic mouse actions (click, enter, exit, press, release)

Magic (continued)

- Interfaces
- Java has strong types
- An interface is a class where **none** of the methods are implemented
- Interfaces can be implemented in very different classes

Interesting Interfaces

- EventListener class (KeyListener, ActionListener, MouseListener, etc...)
- Comportemental (Runnable, Serializable, Clonable,...)
- "Drivers" (SQL drivers, hardware, etc...)
- "Design Patterns"

"Read/Write Heads"

- Reading a file into a frame
- Synchronous I/O

Demo

Magic

- JVM acts as a wrapper for OS calls
- File is a catalog entry, not a regular file
- FileReader is a buffer pointing to the inside of a file
- BufferedReader is a formatter for another reader

Magic (continued)

- new BufferedReader(new FileReader(new File("path")));
- File is a pointer to a file, FileReader is a byte-cruncher, and BufferedReader formats the bytes for later use
- You have many *Reader (String, Object, Data, etc...)

But!

- Careful!
- Be **really** careful!
- As soon as you call foreign code, you have to be extra careful

But!!

- Calling foreign code can trigger a lot of side-effects
- Remember you are in a dynamic environment
- Remember you are in a parallel environment

Exceptions

- Java uses exception styled like C++

- ```
  try {
  // code
  } catch(SomeException e1) {
  } catch(SomeOtherException e2) {
  } (...)
  finally {
  }
  ```
  ```

Exceptions (continued)

- Whatever you might be doing in the try, it is **stopped** by an exception
- The first catch that matches the raised exception is called
- Then, the "finally" block is executed
- The compiler warns you of uncaught exceptions

Exceptions (Continued)

- If a method raises an exception, it is in its prototype:
  void fun(void) throws MyFunkyException
- MyFunkyException is a full-blown class :
  it has methods, variables, etc...
- If you want to raise an exception :
  throw new MyFunkyException()
- Anything that implements the *Throwable* interface can be raised

Exceptions (perspectives)

- Exceptions are not only about real errors, they can be used as *interrupting messages*
- If you don't take care of the exceptions, the program will actually quit

Questions