Fondamenti di Informatica L-B
Esercitazione n°3
Introduzione a JAVA

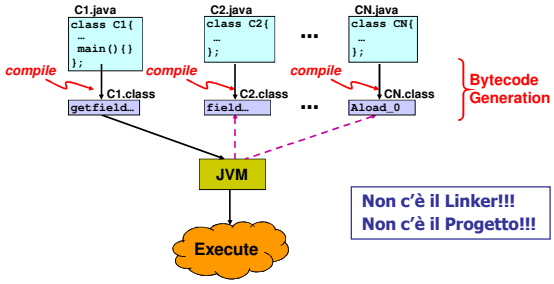
A.A. 2007/08

Tutor: Barbara Pettazzoni

barbara.pettazzoni@studio.unibo.it

Java: Compilazione e Traduzione

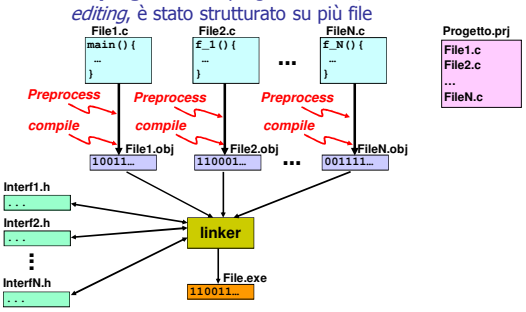
- Editing, compilazione, esecuzione in **Java**



Non c'è il Linker!!!
Non c'è il Progetto!!!

C: Editing, compilazione, esecuzione

- Un **progetto** è un programma che, *durante la fase di editing*, è stato strutturato su più file



Java Virtual Machine

- Il **bytecode** non è Linguaggio-Macchina. Per diventarlo deve subire un'ulteriore trasformazione che viene operata dall'interprete Java in modalità JIT (Just In Time)
  - **Pro**: indipendente dalla piattaforma (portabilità)
  - **Contro**: l'ulteriore trasformazione rallenta l'esecuzione
- Non c'è il **file-progetto**. La JVM esegue solo il file che contiene il `main()`.
  - La JVM cerca i file (.class) nel momento in cui servono (*collegamento dinamico*)
  - È possibile specificare il percorso nel quale cercare tramite la variabile *classpath* (in laboratorio è già impostata)

Strumenti Utilizzati

- **Java™ 6 Software Development Kit (J6-SDK)**
  - Ambiente di sviluppo fornito gratuitamente dalla Sun
  - Comprende diversi tool, in particolare:
    - Un compilatore java (**javac**)
    - Una Java Virtual Machine (**java**)
    - Altre Utility (**javadoc, jar, etc...**)
  - Questi tool sono utilizzabili da linea di comando (MS-DOS)
- **Text Editor qualsiasi (Es Blocco Note – TextTool)**
  - Esistono anche ambienti integrati – IDE (Come Borland JBuilder o Eclipse)

Dal 2006 Java e' Open Source!

Il Linguaggio Java

- **È un linguaggio *totalmente a oggetti:* tranne i tipi primitivi di base (int, float, ...), *esistono solo classi e oggetti***
- **È fortemente ispirato al C++, ma riprogettato *senza il requisito della piena compatibilità col C* (a cui però assomiglia...)**
- **Un programma è un insieme *di classi***
- **Non esistono funzioni definite (come in C) a livello esterno, né variabili globali esterne**
- ***Anche il main è definito dentro a una classe!***

Il concetto di classe

Una *CLASSE JAVA* riunisce le proprietà di:

- ***componente software:* può essere dotata di suoi propri *dati - operazioni***
- ***moduli:* riunisce dati e relative operazioni, fornendo idonei *meccanismi di protezione***
- ***tipi di dato astratto:* può fungere da "stampo" per *creare nuovi oggetti***

Una *classe Java* è una entità *sintatticamente simile alle struct* del C...

- **però, contiene *non solo i dati...* ma anche *le funzioni che operano su quei dati***
- **e ne specifica *il livello di protezione***
  - ***pubblico*: visibile anche dall'esterno**
  - ***privato*: visibile solo entro la classe**
  - **...**

Programmi in Java

Un programma Java è *un insieme di classi e oggetti*

- **Le classi sono componenti *statici*, che *esistono già* all'inizio del programma.**
- **Gli oggetti sono invece componenti *dinamici*, che *vengono creati al momento del bisogno, durante l'esecuzione.***

Il più semplice programma Java è dunque costituito da una singola classe e come minimo tale classe dovrà definire una singola funzione statica, il main

Per convenzione in Java i nomi delle classi iniziano sempre con la lettera maiuscola e i nomi dei singoli campi (dati e funzioni) iniziano sempre con la lettera minuscola

Il MAIN in Java

Il main in Java è una funzione *pubblica* con la seguente *signature obbligatoria*:

```
public static void main(String args[])
  {
      ......
  }
```

Deve essere obbligatoriamente dichiarato **public**, **static**, **void**

Non può avere valore di ritorno (è void)

Deve sempre prevedere gli argomenti dalla linea di comando, *anche se non vengono usati,* sotto forma di *array di* ***String*** *(il primo non è il nome del programma)*

Hello World

```
public class Esempio0
{
   public static void main(String args[])
   {
      System.out.println("Hello World");
   }
}
```

Notare l'uso in java della notazione puntata

```
System.out.println("Messaggio");
```

Il messaggio println("Messaggio")è inviato all'oggetto out che è un membro (statico) della classe predefinita System

Qualche semplice esempio (2)

```
public class EsempioBase
{
   public static void main(String args[])
   {
      int x = 3, y = 4;
      int z = x + y;
      System.out.print("La somma vale: "+z);
   }
}
```

La classe che contiene il main dev'essere ***pubblica***

Classi & File

In Java esiste una ben precisa corrispondenza fra *nome di una classe pubblica* e *nome del file* in cui essa dev'essere definita

- **Una classe *pubblica* deve essere definita in un file *con lo stesso nome della classe* ed estensione .java**

Esempi

classe EsempioBase → file EsempioBase.java

classe Esempio0 → file Esempio0.java

da ciò deriva che è essenziale poter usare *nomi di file lunghi e rispettare maiuscole/minuscole.*

IL JAVA E' CASE-SENSITIVE

Compilazione ed Esecuzione

Usando il JDK della Sun:

- ***Compilazione:***

  javac Esempio0.java

  (produce **Esempio0.class**)

- ***Esecuzione:***

  java Esempio0

La Documentazione

- **È noto che un buon programma dovrebbe essere ben documentato..**
- ***ma l'esperienza insegna che quasi mai ciò viene fatto***
  - ***"non c'è tempo", "ci si penserà poi"...***
  - ***... e alla fine la documentazione non c'è***
- **Java prende atto che la gente *non scrive* documentazione...**
- **..e quindi fornisce uno strumento per *produrla automaticamente* a partire dai *commenti* scritti nel programma:** `javadoc`. **Lo vedremo ora all'opera**

Hello World Completo

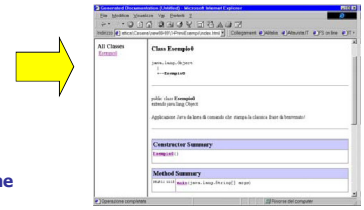
```
/** File Esempio0.java
*  Applicazione Java da linea di comando
*  Stampa la classica frase di benvenuto
*  @author Barbara Pettazzoni
*  @version 1.0, 12/05/2008
*/
public class Esempio0
{
   public static void main(String args[])
   {
    System.out.println("Hello World!");
   }
}
```


Produciamo la documentazione

Per produrre la relativa documentazione:

`javadoc Esempio0.java`

Produce una serie di file HTML



Si consulti la documentazione di javadoc per i dettagli.

ADT – A simple Counter

Il campo **val** è privato:può essere acceduto solo dalle operazioni definite nella medesima classe (reset, inc, getValue), e da nessun altro. Si garantisce l'incapsulamento.

Unico costrutto linguistico per dati e operazioni

```
public class SimpleCounter
{
    private int val;
    public void reset() { val= 0; }
    public void inc()   { val++; }
    public int getValue() { return val;}
}
```

Dati

Operazioni

Creazione di oggetti

Per creare un oggetto:

- prima si definisce un riferimento, il cui tipo è il nome della classe che fa da modello
- poi si crea dinamicamente l'oggetto tramite l'operatore new (simile a malloc)

Esempio:

```
Contatore c; // def. del riferimento
...
c = new Contatore(); // creazione oggetto
```


Uso di oggetti

Ad esempio, se c è un Counter, un oggetto cliente potrà scrivere:

```
c.reset();
c.inc(); c.inc();
int x = c.getValue();
System.out.println("Il contatore vale "+x);
```


Problema

- Scrivere un applicazione Java che sfrutti l'ADT Contatore. L'applicazione deve:
  - Creare un nuovo contatore e inizializzarlo
  - Incrementare il contatore e visualizzare, ogni volta, il valore contenuto al suo interno. Continuare a incrementarlo finché non raggiunge lo stato di errore
  - Terminare

Problema: Un Contatore in Java

- Cosa può essere *salvato* del precedente progetto in C?
  1. **Cos'è** un contatore
  2. Quali sono le **operazioni** relative ad un contatore
  3. **Rappresentazione interna** del contatore
  4. **Interfaccia** del contatore
  5. **Implementazione** del contatore
- Si può ancora parlare di **ADT**?
- Quali sono le maggiori differenze fra **C** e **Java**?

ADT Contatore

1. **Cos'è** un contatore

Un contatore è un oggetto che ha: un valore attuale, *un* valore massimo *di conteggio, uno* stato interno *che indica eventuali condizioni d'errore*

Il valore massimo è selezionabile dall'utente. Se si tenta di superarlo, viene modificato lo stato interno, che può essere letto mediante una funzione specifica

2. **Operazioni** relative ad un contatore

```
void setMaxVal(int valore)
void azzera()
void incrementa()
void visualizza()
int leggi()
boolean isError()
```


ADT Contatore

3. **Rappresentazione interna** dell'ADT Contatore
   - Il contatore in C:

```
typedef struct Contatore
{
  unsigned int valore;
  unsigned int valore_max;
  unsigned int stato;
}
```

   - Il contatore in Java:
     - Una `struct` del C equivale ad un `class` Java (non è vero!!!)

```
public class Contatore
{
  private int valore;
  private int valore_max;
  private boolean errore;
  ...
}
```


ADT Contatore

4. **Interfaccia** dell'ADT Contatore

```
public class Contatore
{
  ...
  public void setMaxVal(int val);
  public void azzera();
  public void incrementa();
  public void visualizza();
  public int leggi();
  public boolean isError();
}
```

- Le operazioni diventano **metodi di classe** e parte integrante dell'ADT Contatore

ADT Contatore

5. **Implementazione** dell'ADT Contatore
   - Una struttura C **non equivale** ad una classe Java
   - Il costrutto `class` è più completo
     - Maggiore capacità espressiva e più intuitivo
     - Incapsulamento
     - Realizza veramente un ADT

Contatore
•int valore •int valore_max •boolean errore
•void setMaxVal(int valore) •void azzera() •void incrementa() •void visualizza() •int leggi() •boolean isError()

ADT Contatore (1)

Implementazione dell'ADT Contatore

```
// "Contatore.java"
public class Contatore
{
   private int valore;
   private int valore_max;
   private boolean errore;

   public void setMaxVal(int val)
    {
        if(val<=0) errore=true;
        else valore_max = val;
    }

   public void azzera()
    {
        valore=0;
        valore_max=0;
        errore=false;
    }

   public boolean isError()
    {
        return errore;
    }
```


ADT Contatore (2)

Implementazione dell'ADT Contatore

```
public class Contatore
{
   ...
   public void incrementa()
   {
       if(valore>=valore_max) errore=true;
       else valore++;
   }

   public void visualizza()
   {
       if(!isError())System.out.println("Valore="+valore);
       else System.out.println("Errore: val. max raggiunto.");
   }

   public int leggi()
   {
       return valore;
   }
}; // Fine Contatore
```


Soluzione

```
// "ApplContatore.java"

public class ApplContatore
{
  public static void main(String[] argv)
  {
    int i=0;
    Contatore cont=new Contatore();
     cont.azzera();
    cont.setMaxVal(10);
    while(!cont.isError())
    {
      cont.visualizza();
      cont.incrementa();
    }
    cont.visualizza();
  } // Fine main()

}  // Fine ApplContatore
```

Javadoc….

- Documentazione indispensabile per programmare.
- Per ogni classe spiega a cosa serve, come si usa e quali metodi fornisce.
- Sul web
  - http://java.sun.com/javase/6/docs/api/
- In laboratorio
  - C:\programmi\java\jdk1.5.0.1\docs

29