Programmazione Orientata agli Oggetti in Linguaggio Java

Classi e Oggetti: Conclusioni Parte a

versione 2.2

G. Mecca – Università della Basilicata – mecca@unibas.it

Sommario

- Struttura del Codice di una Classe
  - ⇨Package
  - ⇨La Clausola Import
- La Piattaforma Java
- La Classe Console
  - ⇨java.util.Scanner

Struttura del Codice di una Classe

- Codice di un'applicazione a oggetti
  - ⇨composto dal codice delle sue classi
  - ⇨il codice delle classi definisce il comportamento di tutti i componenti
- Esempio: applicazione Java
  - ⇨collezione di file .java che contengono il codice delle classi dell'applicazione
  - ⇨opportunamente organizzati in package

Struttura del Codice di una

ATTENZIONE
all'uso del termine classe

- Classe come componente
  - ⇨componente dell'applicazione di tipo classe
  - ⇨"chiedo alla classe Circonferenza di creare un oggetto"
- Classe come tipo di uno o più oggetti
  - ⇨struttura di una collezione di oggetti
  - ⇨"utilizzo un array di riferimenti a oggetti di tipo Circonferenza"
- Classe come unità di codice sorgente
  - ⇨file di codice sorgente che descrive la classe
  - ⇨"compilo la classe Circonferenza"

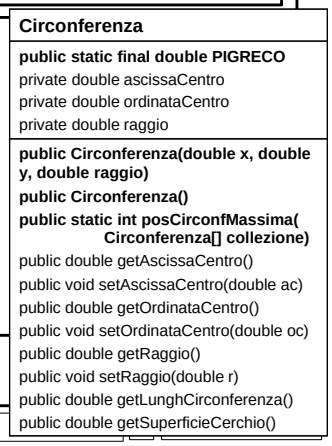
Struttura del Codice di una Classe

- Struttura del codice della classe
  - ⇨due tipologie di istruzioni diverse
  - ⇨codice relativo ai compiti della classe: consente alla classe di svolgere i suoi compiti
  - ⇨codice relativo alla descrizione degli oggetti: descrive il tipo delle istanze della classe
- Nel sorgente
  - ⇨le due tipologie di codice sono “mischiate”

Struttura del Codice di una Classe

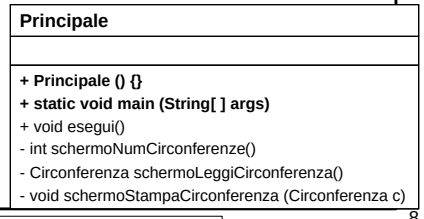
- Codice relativo ai compiti della classe
  - ⇨eventuali costanti statiche
  - ⇨eventuali proprietà statiche
  - ⇨eventuali metodi statici (tra cui il main)
  - ⇨costruttori (sempre almeno uno)
- Codice relativo alla tipo degli oggetti
  - ⇨eventuali proprietà degli oggetti della classe
  - ⇨eventuali metodi degli oggetti della classe

```
package circonferenzea;
public class Circonferenza {
    public final static double PIGRECO = 3.14;
    public static int posizioneMassimaCirconferenza(Circonferenza[ ] collezione) { ... }
    public Circonferenza(double ascissaCentro, double ordinataCentro, double raggio)  { ... }
    private double ascissaCentro, ordinataCentro, raggio;
    public double getAscissaCentro() { ... }
    public void setAscissaCentro(double ac) { ... }
    public double getOrdinataCentro() { ... }
    public void setOrdinataCentro(double oc) { ... }
    public double getRaggio() { ... }
    public void setRaggio(double r) { ... }
    public double getLunghezzaCirconferenza() { ... }
    public double getSuperficieCerchio() { ... }
}
```




```
package circonferenzea;
public class Principale {
    public void esegui() { ... }
    private int schermoNumCirconferenze() { ... }
    private Circonferenza schermoLeggiCirconferenza() { ... }
    private void schermoStampaCirconferenza(Circonferenza c) { ... }
    public static void main(String args[]) {
        Principale principale = new Principale();
        principale.esegui();
    }
}
```

Convenzione UML:
+ : public
- : private



Struttura del Codice di una Classe

- Codice sorgente di una classe
  - ⇨in Java: file di testo con estensione .java
  - ⇨es: Circonferenza.java
- Attenzione
  - ⇨in Java le scelte fatte nel codice sorgente hanno varie ripercussioni sulla struttura fisica dei file su disco
  - ⇨in particolare, bisogna rispettare varie regole

Struttura del Codice di una Classe

- Regole sul nome dei file
  - ⇨il file .java che contiene il codice di una classe deve avere lo stesso nome della classe
  - ⇨incluse minuscole e maiuscole (es: calcolatrice.java è errato)
  - ⇨inoltre, il file deve essere inserito in una struttura di cartelle del disco che riflette il nome del package (>>)

Package

- Package
  - ⇨strumento per organizzare le classi
- Due aspetti diversi
  - ⇨organizzazione logica dei nomi: ogni package è uno “spazio di nomi” (namespace) distinto
  - ⇨organizzazione fisica dei file: ogni package è una cartella distinta del disco

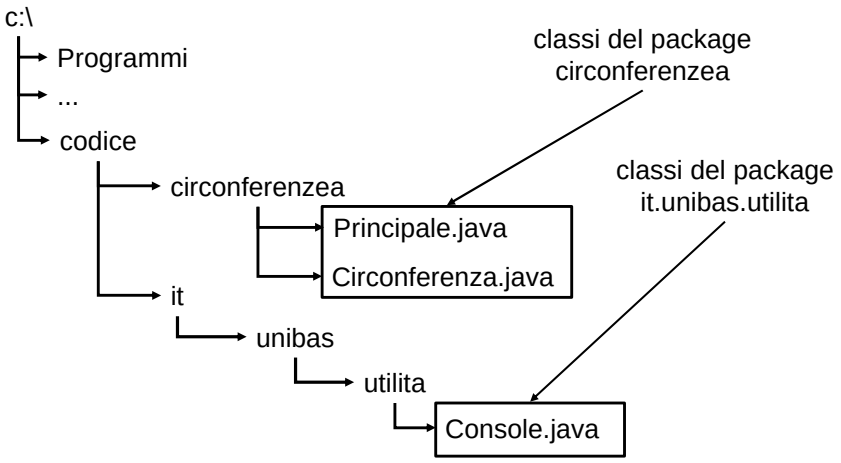
Package

- Sintassi
  - ⇨package *nomePackage*;
  - ⇨es: package circonferenzea;
  - ⇨deve essere la prima istruzione della classe
- Semantica
  - ⇨le classi del package sono identificabili con *nomePackage.nomeClasse*
  - ⇨es: circonferenzea.Principale

Package

- Regole sulla struttura delle cartelle
  - ciascun package corrisponde ad una cartella del disco
  - es: package circonferenzea: i file delle classi del package devono stare in una cartella chiamata circonferenzea
  - es: package it.unibas.utilita: i file delle classi devono stare in una cartella chiamata utilita contenuta in una cartella unibas contenuta in una cartella it

Package



Package

- Regola di Java
  - ⇨tutte le classi devono appartenere ad un package
  - ⇨se il programmatore non specifica un package, il compilatore assegna la classe ad un package implicito
  - ⇨corrispondente alla cartella in cui il file della classe è posizionato (ma anonimo)
  - ⇨è opportuno specificare sempre il package

Package

- Per utilizzare una classe in un'altra
  - ⇨è necessario fare riferimento alla classe con il nome completo: *nomePackage.nomeClasse*
- Con alcune eccezioni
  - ⇨classi dello stesso package: è possibile utilizzare semplicemente il nome
  - ⇨il package java.lang (che include System, String, Math ecc.)
  - ⇨classi fondamentali di java, molto usate

La Clausola import

- La clausola import
  - ⇨per abbreviare il codice, è possibile utilizzare la clausola import per accorciare i nomi
- In sintesi
  - ⇨è possibile includere una serie di istruzioni import nel codice di una classe
  - ⇨per dichiarare una serie di nomi di classi per i quali si utilizzeranno nomi abbreviati (solo il nome della classe senza package)

La Clausola import

- Utilizzo di import
  - ⇨riferimento ad una classe specifica
  - ⇨import *nomePackage.nomeClasse;*
  - ⇨es: import java.util.Date;
  - ⇨da quel momento posso scrivere solo Date
  - ⇨riferimento ad un intero package
  - ⇨import *nomePackage.**;
  - ⇨es: import java.util.*;

La Clausola import

```
package circonferenzea;

public class Principale {

    private int schermoLeggiCirconf() {
        System.out.println("Quante circonf. ?");
        System.out.print("---> ");
        int numeroCirconferenze =
                it.unibas.utilita.Console.leggiIntero();
        while (numeroCirconferenze < 0) {
            System.out.println("Errore.");
            System.out.print("Ripeti. ---> ");
            numeroCirconferenze =
                it.unibas.utilita.Console.leggiIntero();
        }
        return numeroCirconferenze;
    }

}
```

```
package circonferenzea;

import it.unibas.utilita.Console;

public class Principale {

    private int schermoLeggiCirconf() {
        System.out.println("Quante circonf. ?");
        System.out.print("---> ");
        int numeroCirconferenze =
                Console.leggiIntero();
        while (numeroCirconferenze < 0) {
            System.out.println("Errore.");
            System.out.print("Ripeti. ---> ");
            numeroCirconferenze =
                Console.leggiIntero();
        }
        return numeroCirconferenze;
    }
}
```


La Clausola import

ATTENZIONE
al significato di import

- Attenzione
  - ⇨import non ha niente a che vedere con la visibilità di una classe in un'altra
  - ⇨qualsiasi classe pubblica è potenzialmente visibile in un'altra (>>)
  - ⇨import è solo uno strumento sintattico per abbreviare il codice
  - ⇨ma non è in alcun modo indispensabile
  - ⇨differenza rispetto a #include in C/C++

La Piattaforma Java

- Negli esempi
  - ⇨varie classi della libreria standard fornita a corredo della piattaforma Java 2 SE
  - ⇨varie migliaia di classi e metodi
- Per programmare in Java
  - ⇨bisogna conoscere i concetti della programmazione a oggetti
  - ⇨ma anche le classi fornite a corredo della piattaforma

La Piattaforma Java

- Cosa offrono esattamente queste classi ?
  - ⇨è possibile scoprirlo consultando la documentazione delle API di Java
  - ⇨collezione di pagine HTML consultabili a partire da %JAVA_HOME%\docs\index.html
  - ⇨descrive tutti i package, le classi, i metodi, le proprietà delle classi di Java, con opportuni commenti di documentazione

>> %JAVA_HOME%\docs\index.html

La Piattaforma Java

- Le convenzioni di stile
  - ⇨nelle moderne piattaforme ad oggetti sono considerate essenziali
- Le convenzioni di stile ufficiali della Sun
  - ⇨pubblicate assieme al linguaggio, sono adottate consistentemente nelle API
  - ⇨forniscono una serie di regole sul modo di scrivere codice Java

La Piattaforma Java

ATTENZIONE
alle convenzioni
di stile di Java

- Le regole principali (>>)
  - ⇨i nomi di classe cominciano con la maiuscola e seguono la notazione cammello
  - ⇨i nomi di proprietà e di metodi cominciano con la minuscola e seguono la notazione cammello
  - ⇨i nomi di package sono scritti in lettere minuscole
  - ⇨i nomi di costanti sono scritti in maiuscole

La Classe Console

- Gestione dei flussi standard in Java
  - attraverso la classe java.lang.System
- Tre proprietà pubbliche
  - System.out: riferimento ad un oggetto che rappresenta lo standard output
  - System.in: standard input
  - System.err: standard error (flusso di uscita destinato ai messaggi di errore)
  - gli oggetti relativi sono creati dalla m.v.

La Classe Console

- Utilizzo dei flussi di uscita
  - oggetti di tipo java.io.PrintStream
  - metodi print() e println()
- Utilizzo del flusso di ingresso
  - il riferimento System.in
  - riferimento ad un oggetto di tipo java.io.InputStream
  - non consente altrettanto facilmente di lavorare con lo standard input

La Classe Console

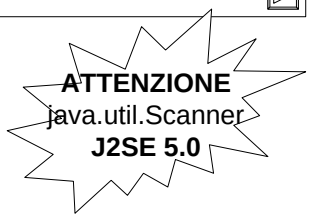
- In particolare
  - ⇨System.in deve essere sottoposto a varie trasformazioni per effettuare input non formattato
  - ⇨e richiede operazioni aggiuntive di conversione per l'input formattato
- La classe it.unibas.utilita.Console
  - ⇨un componente specializzato in operazioni di lettura dallo standard input

La Classe Console

- I metodi di Console
  - ⇨public static String leggiStringa()
  - ⇨public static int leggiIntero()
  - ⇨public static float leggiFloat()
  - ⇨public static double leggiDouble()
  - ⇨public static char leggiChar()

>> Documentazione della classe Console

java.util.Scanner



- In effetti
  - ⇨in J2SE 5.0 è stata introdotta una classe appositamente per l'input formattato e non
- java.util.Scanner
  - ⇨consente di creare oggetti di tipo scanner a partire da System.in
  - ⇨e poi utilizzarli per operazioni di input

java.util.Scanner

- I metodi di java.util.Scanner
  - ⇨public String next()
  - ⇨public int nextInt()
  - ⇨public float nextFloat()
  - ⇨...
- Esempio

```
java.util.Scanner scanner =
                    new java.util.Scanner(System.in);
System.out.println("Immetti un intero");
int i = scanner.nextInt();
```

java.util.Scanner

- Ma...
  - ⇨come tutte le funzionalità di J2SE 1.5, si tratta di una funzionalità non standard
  - ⇨non disponibile nelle vecchie versioni della macchina virtuale
  - ⇨i metodi non sono altrettanto robusti rispetto a quelli di Console
- Di conseguenza, nel corso
  - ⇨continueremo ad utilizzare la classe Console

Riassumendo

- Struttura del Codice di una Classe
  - ⇨Package
  - ⇨La Clausola Import
- La Piattaforma Java
- La Classe Console
  - ⇨java.util.Scanner

Termini della Licenza