

Esercitazione n° 2

Obiettivi



Progettazione di una applicazione Java

- Utilizzo di classi come “schemi”
- Utilizzo di costruttori
- Utilizzo di stringhe
- Uguaglianza tra oggetti
- Utilizzo di classi come componenti software statici



Utilizzo di **JCreator** come Editor

Per la compilazione ed esecuzione usare ancora gli strumenti JDK...

Problema



Scopo

Progettare un'applicazione Java: un Gioco Specifica

Si progetti un gioco che simuli il combattimento tra entità **Marziane** e **Terrestri**. Il comportamento astratto di tali entità è specificato nelle slide n°3 e n°6. Il componente software che gestisce il gioco dovrà creare un oggetto di tipo 'Marziano' con matricola non definita ed uno invece con matricola "x102". Dovrà creare inoltre un oggetto di tipo 'Terrestre' dal nome sconosciuto ed uno di nome "Ken". L'oggetto "x102" deve attaccare i Terrestri mentre "Ken" deve attaccare i Marziani. A seconda che vincano o perdano l'indice di vitalità di "x102" e "Ken" può salire o scendere. Inoltre, il componente software che gestisce il gioco dovrà farli entrambi attaccare due volte stampando in uscita il loro stato al termine di ogni attacco.

L'entità MARZIANO (1)

Ogni marziano ha un numero di **matricola**. Il suo stato è rappresentato da un'**autonomia**, un livello di **Intelligenza Artificiale** e di **tecnologia**. Un marziano può attaccare i terrestri.



Più in dettaglio...

- La **matricola** è una stringa alfanumerica.
- L'**autonomia** è rappresentata da un indice compreso tra 1 e 100. Quando tale indice è 100 l'autonomia è massima, quando è zero il marziano è morto.
- L'**AI** è rappresentata da un indice compreso tra 1 e 10.
- La **tecnologia** è rappresentata da un indice compreso tra 1 e 10.

? *Come astrarre tali entità marziane?*

L'entità MARZIANO (2)



*Occorrerà una classe **Marziano** che definisce il tipo di dato astratto 'Marziano'.*



- **Parte Nascosta (stato):** matricola, AI, tecnologia e autonomia non devono essere accessibili direttamente dall'esterno...
- **Parte Visibile (costruttori):**
 - Marziano() //parametri di default: matricola = "indefinito"
//autonomia = 50; AI = 5; tecnologia=5
 - Marziano (String matricola, int[] param)
// param è un array di 3 interi che corrispondono ai valori di // autonomia, AI e tecnologia di inizializzazione

Al termine della inizializzazione ogni costruttore deve stampare in uscita: "Il Marziano [matricola] entra in gioco."

L'entità MARZIANO (3)

○ Parte Visibile (operazioni):



- void attaccaTerrestri() //vedi slide 11
- boolean attivo() // **true** se autonomia>0; **false** altrimenti
- String toString() // restituisce la matricola e il livello di autonomia
// corrente es: "Marziano x102. Autonomia = 50"
- boolean equals (Marziano x) ←

Ai fini del gioco due oggetti 'Marziani' sono uguali se hanno le stesse 'potenzialità offensive'. Ossia se, indipendentemente dalla matricola, hanno lo stesso livello di autonomia, AI e tecnologia. Se uno solo di questi parametri è diverso i due Marziani non si equivalgono.

L'entità TERRESTRE (1)

Ogni terrestre ha un **nome**. Il suo stato è rappresentato dalla **vita** che ha ancora a disposizione, dalla **velocità** di combattimento e da un **armamento**. Un terrestre può attaccare i marziani.



Più in dettaglio...

- Il **nome** è una stringa di caratteri.
- La **vita** disponibile è rappresentata da un indice compreso tra 1 e 100. 100 è l'indice massimo di vita disponibile, quando la vita è zero il terrestre è morto.
- La **velocità** è rappresentata da un indice compreso tra 1 e 10.
- L'**armamento** è rappresentato da un indice compreso tra 1 e 10.

? *Come astrarre le entità terrestri?*

L'entità TERRESTRE (2)



Occorrerà una classe *Terrestre* che definisce il tipo di dato astratto 'Terrestre'.



- **Parte Nascosta (stato):**

Nome, vita, velocità e armamento non devono essere accessibili direttamente dall'esterno...

- **Parte Visibile (costruttori):**

- Terrestre() // parametri di default: nome = "sconosciuto"

- // vita = 50; velocità = 5; armamento=5

- Terrestre (String nome, int[] param)

- // param è un array di 3 interi che corrispondono ai valori di
 - // vita, velocità e armamento di inizializzazione

Ai termine della inizializzazione ogni costruttore deve stampare in uscita: "Il Terrestre [nome] entra in gioco."

Laboratorio di Fondamenti I TLC – Esercitazione II

7

L'entità TERRESTRE (3)

- **Parte Visibile (operazioni):**

- void attaccaMarziani() // vedi slide 11

- boolean vivente() // **true** se vita>0; **false** altrimenti

- String toString() // restituisce il nome e il livello di vita
// corrente es: "Terrestre Ken. Vita = 75"

- boolean equals (Terrestre x) ←



Ai fini del gioco due oggetti 'Terrestri' sono uguali se hanno le stesse 'potenzialità offensive'. Ossia se, indipendentemente dal nome, hanno lo stesso livello di vita, velocità e armamento. Se uno solo di questi parametri è diverso i due Terrestri non si equivalgono.

Laboratorio di Fondamenti I TLC – Esercitazione II

8

L'applicazione 'GiocoBattaglia'(1)

Il componente software che gestisce il gioco dovrà...

- ① Segnalare in uscita che il gioco ha inizio.
- ② Definire e creare un oggetto 'Marziano' con matricola e parametri di default (costruttore senza parametri).
- ③ Definire e creare un oggetto 'Terrestre' con nome e parametri di default (costruttore senza parametri).
- ④ Definire e creare il marziano "x102" con autonomia=50, AI=5, tecnologia=5.
- ⑤ Definire e creare il terrestre "Ken" con vita=75, velocita=10, armamento=6.

L'applicazione 'GiocoBattaglia'(2)

Il componente software che gestisce il gioco dovrà...

.....

- ⑥ Confrontare l'equivalenza ai fini del gioco dei due Terrestri creati ("Ken" e "sconosciuto") e stampare in uscita un messaggio con il risultato del confronto.
- ⑦ Ripetere il punto 6 per i due Marziani creati.
- ⑧ Mostrare in uscita lo stato di vita di Ken e di autonomia di x102 con l'indicazione che si tratta dello stato prima dell'attacco.
- ⑨ Far attaccare (se vivi) 2 volte Ken e x102 e mostrare ogni volta in uscita il loro stato di vita e autonomia con l'indicazione che si tratta dello stato dopo dell'attacco.
- ⑩ Segnalare in uscita che il gioco è terminato.

Simulazione di un attacco

Viene qui riportato il codice del metodo **attaccaMarziani()** nella classe **Terrestre**. Si basa per semplicità sui valori delle variabili di istanza.

Da utilizzare nella implementazione della classe **Terrestre** e da adattare per la classe **Marziano**.

```
/**
 * Simulazione di un attacco.
 */
public void attaccaMarziani(){
    if (!vivente()) System.out.println("Non posso attaccare:
                                     Sono morto!");

    else {
        // algoritmo di attacco semplificato
        vita = vita - 20 + 2 * velocita + armamento;
    }
}
```

Laboratorio di Fondamenti I TLC – Esercitazione II

11

Esercitazione n°2

Come procedere?

- 1 Scaricare il file **GiocoBattaglia.java** dal sito Web del corso e salvarlo in **C:\TEMP**
- 2 Il file **GiocoBattaglia.java** contiene il **main()** completo dell'applicazione.



Capirne il funzionamento e implementare le classi mancanti: **Terrestre.java** e **Marziano.java** sulla base della specifica data. Si usi come editor JCreator.

- 3 Compilare tutto ed eseguire l'applicazione usando i comandi della JDK attraverso la Console per Java.

Laboratorio di Fondamenti I TLC – Esercitazione II

12

Esercitazione n°2

Suggerimento per creare la classe Terrestre.java...




```
/** Classe che astrae un <B>Terrestre</B>. */
public class Terrestre{
    // variabili
    private String nome;
    ....
    //costruttori
    public Terrestre(){ ... }
    public Terrestre(String name, int[] param) { ...}
    // metodi
    public void attaccaMarziani(){ // come visto }
    public boolean vivente(){ if (...) return true;
                               else return false; }
    public String toString(){...}
}
```

Analogamente per Marziano.java...

Laboratorio di Fondamenti I TLC – Esercitazione II

13

Esercitazione n°2: facoltativo

-  Documentare (**javadoc**) tutti i metodi e le variabili delle classi Marziano.java e Terrestre.java.
-  Cambiare il componente software che gestisce il gioco:
 - Creare più personaggi.
-  Varianti nei tipi di dati astratti:
 - Cambiare la visualizzazione dello stato (metodo toString()) degli oggetti stampando in uscita, per esempio, anche il valore corrente di AI e tecnologia per i Marziani e di velocità e armamento per i Terrestri.

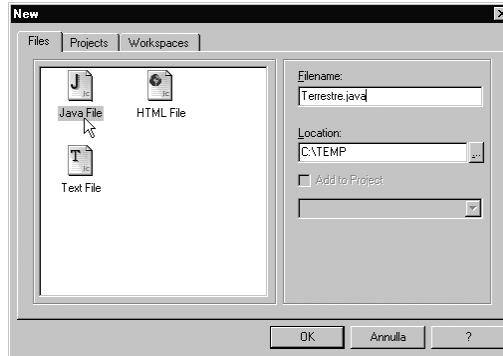
Laboratorio di Fondamenti I TLC – Esercitazione II

14

Appendice: JCreator come Editor (1)

CREAZIONE NUOVO FILE

Per scrivere un nuovo file usando JCreator cliccare su **File** → **New** o sul bottone corrispondente



Selezionare quindi nella finestra **Files** l'icona denominata **Java File** ed inserire il nome del file completo nel campo **Filename:** ed il percorso in cui vogliamo crearlo in **Location:**

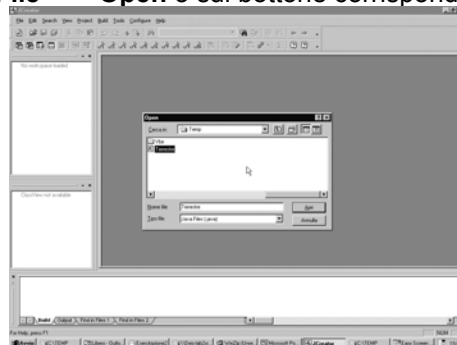
Laboratorio di Fondamenti I TLC – Esercitazione II

15

Appendice: JCreator come Editor (2)

APERTURA FILE ESISTENTE

Per aprire un file già esistente usando JCreator cliccare su **File** → **Open** o sul bottone corrispondente



Selezionare quindi dalla finestra il File desiderato.



Ricordarsi di salvare (File → Save) dopo ogni modifica prima di compilare!

Laboratorio di Fondamenti I TLC – Esercitazione II

16