Programmazione II

Esercitazione 01: Attributi e metodi statici e dinamici

Alessandro Mazzei

Slides Credit: Daniele Radicioni

breve sintesi dei temi oggetto dell’esercitazione

- Classe
- Oggetto
- Statico e dinamico nel mondo reale e in Java
- Visibilità
- Modificatori di accesso: attributi e metodi private
- Final
- Asserzioni
- Java Visualizer, un esempio
- Matita e Elicottero: final, static/dinamico, public/private

Classe: …

?

Classe

- è un costrutto che serve a selezionare come rilevanti alcuni elementi di un certo dominio (un pezzo di mondo) e permette di definire oggetti che hanno la caratterizzazione della classe stessa:
  - permette di descrivere questi oggetti per mezzo di blocchi di informazione più semplice, che chiamiamo ‘campi’,
  - permette di descrivere gli oggetti specificando quali operazioni possono essere condotte sui vari oggetti, e anche sulla classe in sé; chiamiamo ‘metodi’ (dinamici/di istanza e statici/di classe, rispettivamente) queste operazioni.

Classe

- Una classe è la definizione di un **tipo** di oggetto.
- Una classe specifica il nome e il tipo delle **variabili di istanza** degli oggetti. Specifica anche **variabili statiche**, o della classe.
- Una classe specifica i **metodi dei suoi oggetti**. Specifica anche **metodi statici**, o della classe.
- Un oggetto di una classe è una **istanza** della classe.

Oggetto

- Un oggetto è l'indirizzo in memoria di un gruppo di dati più semplici, che sono la rappresentazione in memoria dei campi, disposti consecutivamente in memoria.

‘statico’ e ‘dinamico’

Consideriamo queste due asserzioni:

1. **« l’elefantino Bertha pesa 600 kg »**
2. **« gli elefanti sono una specie in via di estinzione »**

- Nel primo caso sto descrivendo un individuo (Bertha); altri individui hanno peso diverso. la proprietà del peso è associata a ciascuna istanza, cioè ogni individuo ha un proprio peso.
- Nel secondo caso stiamo invece enunciando una proprietà associata alla classe degli elefanti:
  - questa proprietà, pur vera per la specie degli elefanti, può non essere vera per gli individui appartenenti alla classe: è pertanto associata alla classe ma non alle sue istanze.

‘statico’ e ‘dinamico’

Consideriamo queste due asserzioni:

1. **« l’elefantino Bertha pesa 600 kg »**
2. **« gli elefanti sono una specie in via di estinzione »**

- Nel primo caso sto descrivendo un individuo (Bertha); altri individui hanno peso diverso. la proprietà del peso è associata a ciascuna istanza, cioè ogni individuo ha un proprio peso.

attributo dinamico/di istanza

- Nel secondo caso stiamo invece enunciando una proprietà associata alla classe degli elefanti:

attributo statico/di classe

  - questa proprietà, pur vera per la specie degli elefanti, può non essere vera per gli individui appartenenti alla classe: è pertanto associata alla classe ma non alle sue istanze.

dinamico *vs* statico

```
[public|private] static <tipo> <metodo> (<tipo1 parametro1>, ...)
```

- **dinamico**: di istanza (qualsiasi metodo non dichiarato statico si intende dinamico)
- **statico**: di classe; non appartenente a un dato oggetto, ma all'intera classe.
  - un metodo statico *esiste indipendentemente dall'esistenza di oggetti di una data classe*;
  - I metodi statici non possono fare riferimenti a campi di oggetto
  - NB: 'statico' ≠ 'costante'
- Per invocare un metodo statico (pubblico) dall'esterno della classe invocare **Classe.metodo()**, come Math.min(x,y)

dinamico *vs* statico

- operativamente, **DICHIARAZIONI**
  - per dichiarare un attributo statico utilizziamo la sintassi

```
[public|private] static <tipo> nomeAttributo
```

  - per dichiarare un metodo statico utilizziamo la sintassi

```
[public|private] static <tipo> <nomeMetodo> (<tipo1 parametro1>, ...)
```

- operativamente, **INVOCAZIONI**
  - Per invocare un metodo statico (pubblico) dall'esterno della classe utilizziamo **Classe.metodo()**, come Math.min(x,y), e per accedere a un attributo statico **Classe.attributo**; dall'interno della classe drittamente **metodo()** e **attributo**.

Visibilità

- le classi sono contenute in file; un file può contenere più classi.
- una qualunque classe X può essere resa visibile alle altre classi del programma

1. definizione di **class X**: in questo caso X è utilizzabile solo da classi definite in file memorizzati nella stessa cartella (*package*).
2. definizione di **public class X**: in questo caso la classe può essere usata da classi in altri file che stanno anche in altri package (importando il package, assegnando una variabile d'ambiente, o fornendo al compilatore informazioni su dove trovare la classe in questione).

Modificatori di accesso: attributi e metodi privati

- information hiding: **nascondere i dettagli implementativi** di una classe (idea: esporre funzionalità senza far sapere come queste sono effettivamente implementate)
- se i campi sono privati, possono essere acceduti/modificati solo dall'interno della classe
  - metodi **get** e **set** servono ad accedere ai campi dell'oggetto (per leggerne e assegnarne i valori, rispettivamente)
  - *naming convention*: `getNomeCampo` e `setNomeCampo`;
  - metodi set sono lo strumento per verificare i valori assegnati alle variabili private, e quindi per mantenere la correttezza del programma e/o notificare all'utente casi di assegnamenti inappropriati

Modificatori di accesso: attributi e metodi privati

- un uso tipico dei modificatori d'accesso è mantenere le variabili di istanza private;
  - questo rende il nome della variabile non più accessibile dall'esterno della classe in cui questa è definita
- i metodi possono essere pubblici o privati,
  - metodi privati sono metodi che saranno utilizzati solo da altri metodi della classe

final

- Indicati come

```
[public|private] [static] final <tipo> <nome simbolico>
```

- Rende un campo **non più modificabile una volta assegnato il valore**
- Campo sempre accessibile in lettura ma non in scrittura
- Un tipico utilizzo è memorizzare le costanti numeriche
  - Esempio: i campi *pigreco* ed *enepero* di una ipotetica classe Matematica

asserzioni

- asserzione: istruzione che specifica un'ipotesi sullo stato del programma.
  - per esempio, il valore della variabile n deve essere 1:

```
assert n == 1;
```

- È possibile inoltre aggiungere un messaggio informativo (sarà stampato in caso l'asserzione fallisca sollevando un'eccezione che terminerà il programma), raffinando l'espressione riportata sopra come segue:

```
assert n == 1: "valore di n diverso da 1: " + n;
```

asserzioni

- il controllo delle asserzioni può essere abilitato/disabilitato a piacere
  - per esempio possiamo attivarlo durante il debugging
- di default non vengono mandate in esecuzione: per attivare il controllo delle asserzioni usiamo il comando

```
java -ea Programma
```

- che corrisponde all'opzione `-enableassertions`.
  - studiamo il primo semplice esempio presente nella classe `TestAsserzioni.java`.

- analizziamo qualche esempio con Java Visualizer

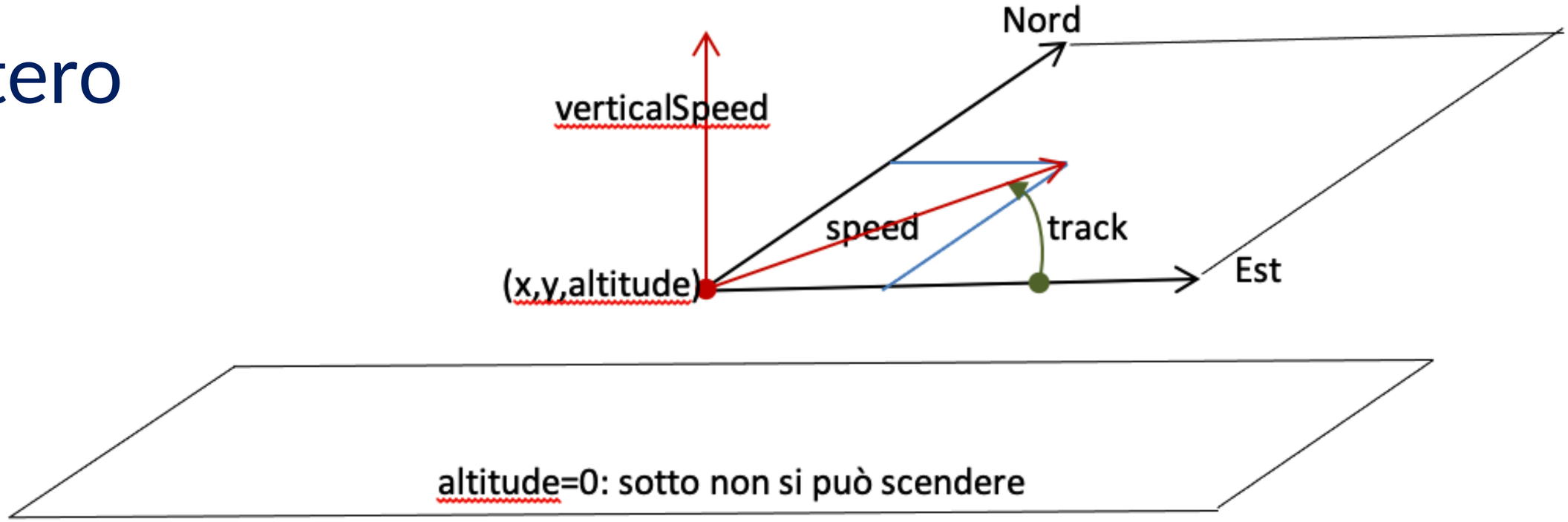
https://cscircles.cemc.uwaterloo.ca//java_visualize/#mode=display

esercizi

matita

- scrivere una classe pubblica Matita per rappresentare virtualmente matite. Una matita è definita come
  - uno **stelo** (una lunghezza intera espressa in millimetri, che varia da un minimo **minStelo** a un massimo **maxStelo**); seguito da
  - una **punta** (un intero **da 0** a un massimo **maxPunta**).

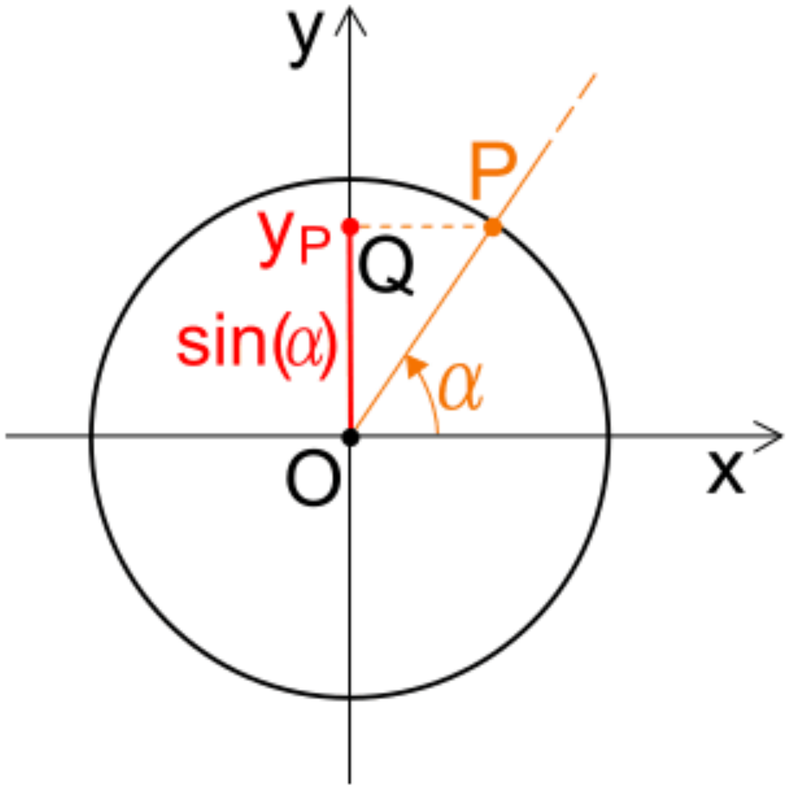
elicottero

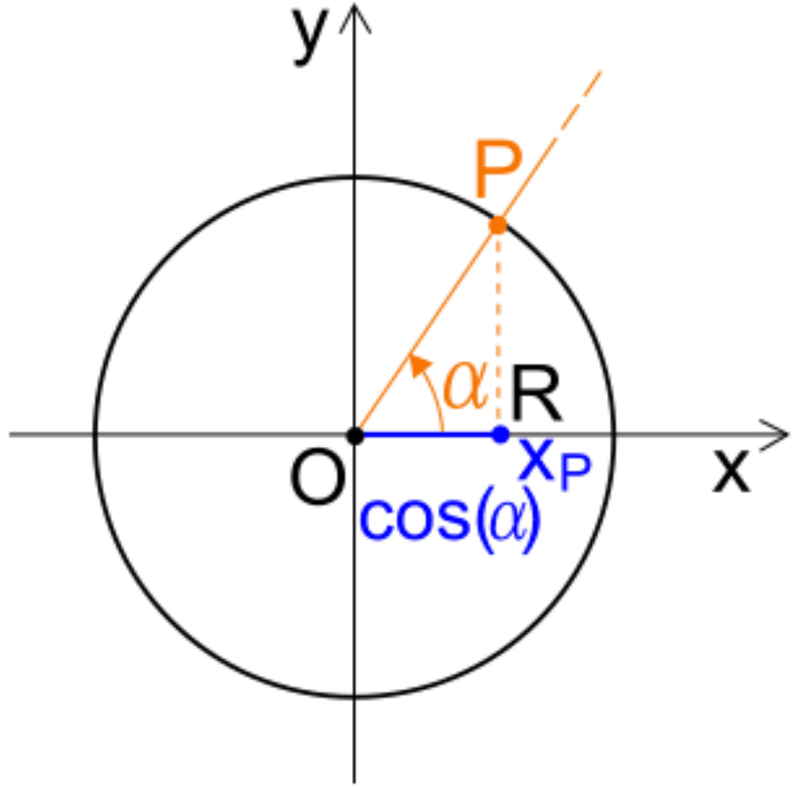


- scrivere una classe Elicottero per rappresentare virtualmente elicotteri. Un elicottero è definito con **tre coordinate** (intere, in km): x,y e altitude (non negativa), **due velocità** (intere, in hm/h), speed (orizzontale e non negativa) e verticalSpeed (verticale), e una direzione orizzontale **track** (un reale, un angolo in radianti tra 0 e 2π)

elicottero

Relazioni trigonometriche fondamentali, per aggiornare la posizione





$$\sin(\alpha) = \frac{OQ}{OP} = \frac{OQ}{1} = Y_P$$

$$\cos(\alpha) = \frac{OR}{OP} = \frac{OR}{1} = X_P$$

Fine