Programmazione II-A 2020-21

Esercitazione a turni riuniti

Attilio Fiandrotti
attilio.fiandrotti@unito.it

10 Marzo 2021

Esercitazioni a turni riuniti

- Tre esercitazioni da due ore l'una
- A turni riuniti (tutto corso A)
- Date delle due prossime esercitazioni
  - Settimana 29 Mar-2 Apr *OR* 5-9 Apr
  - Settimana 19-23 Apr
- Orario ?

Orario prossime esercitazioni

Orario delle lezioni del primo anno del Corso A

Ora	Lun	Mar	Mer	Gio	Ven	Sab
9-10	AnMat A (Aula A)	ArchElab A T2 (Laboratorio Turing)	AnMat A (Aula A)	ArchElab A T1 (Laboratorio Turing)	Prog II A (Aula A)	
10-11	AnMat A (Aula A)	ArchElab A T2 (Laboratorio Turing)	AnMat A (Aula A)	ArchElab A T1 (Laboratorio Turing)	Prog II A (Aula A)	
11-12	ArchElabA (Aula A)	ArchElab A T2 (Laboratorio Turing)	Prog II A (Aula A)	ArchElab A T1 (Laboratorio Turing)	AnMat A (Aula A)	
12-13	ArchElabA (Aula A)		Prog II A (Aula A)		AnMat A (Aula A)	
13-14		AnMat A (Aula A)		Prog II A T1 (Laboratorio Turing)		
14-15	Prog II A T2 (Laboratorio Turing)	AnMat A (Aula A)		Prog II A T1 (Laboratorio Turing)	AnMat tutA (Aula A)	
15-16	Prog II A T2 (Laboratorio Turing)	ArchElabA (Aula A)		Prog II A T1 (Laboratorio Turing)	AnMat tutA (Aula A)	
16-17	Prog II A T2 (Laboratorio Turing)	ArchElabA (Aula A)	Ingl I (Aula E)		AnMat tutA (Aula A)	
17-18			Ingl I (Aula E)			
18-19			Ingl I (Aula E)			

Attributi *Statici*

- Indicati come

[public|private] <u>static</u> tipo nome

- Un attributo statico é condiviso da tutte le *istanze* di una classe
  - se modificato dalla istanza *A*, la modifica sarà visibile anche all'istanza *B*
- Per richiamare un attributo statico pubblico fuori dalla sua classe scrivete ***Classe.attributo***
- Dentro la classe scrivete semplicemente ***(this.)attributo***

Metodi *Statici*

- Indicati come

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

- Un metodo statico esiste indipendentemente dall'esistenza di (*anche senza*) istanze di una data classe
- I metodi statici non fanno riferimenti alle variabili d'istanza
- Per richiamare un attributo statico pubblico fuori dalla sua classe scrivete ***Classe.metodo()***
- Dentro la classe scrivete semplicemente ***metodo()***

Attributi *Final*

- Indicati come

[public|private] [static] <u>final</u> tipo nome

- Rende una variabile non più modificabile

Esercizio *Matita*

- Scrivere una classe pubblica Matita. Una matita è definita come uno stelo (lunghezza intera in millimetri da ***minStelo*** a ***maxStelo***) seguita da una punta (un intero da 0 a ***maxPunta***).
- Fissate nella definizione della classe dei valori per massimi e minimi, per esempio: ***minStelo=10, maxStelo=200, maxPunta=5***.
- ***(i) minStelo, maxStelo, maxPunta*** sono attributi interi ***pubblici***, ***statici*** e ***final*** della classe Matita (non legati a un oggetto ma alla classe). Invece stelo e punta sono attributi interi ***dinamici***.
- ***(ii)*** Il costruttore di ***Matita*** consente di costruire una matita con punta di lunghezza massima dato lo stelo. Un assert impedisce lunghezze non accettabili dello stelo.

Esercizio *Matita*

- ***(iii)*** La classe ha i metodi ***get*** per stelo e punta e nessun metodo ***set***: non consento di cambiare la lunghezza a una matita.
- ***(iv)*** Un metodo “disegna” restituisce “true” (successo) se la matita ha almeno 1mm di punta, e “false” (fallimento) altrimenti. Nel primo caso usa la matita fino a ridurne la punta di un 1mm.
- ***(v)*** Un metodo “tempera” restituisce “true” (successo) se la matita è più lunga del minimo e “false” (fallimento) altrimenti. Nel primo caso riduce lo stelo di 1mm e allunga la punta di 1mm, a meno che la lunghezza della punta sia già il massimo. In questo caso la matita si accorcia ma la punta resta invariata.

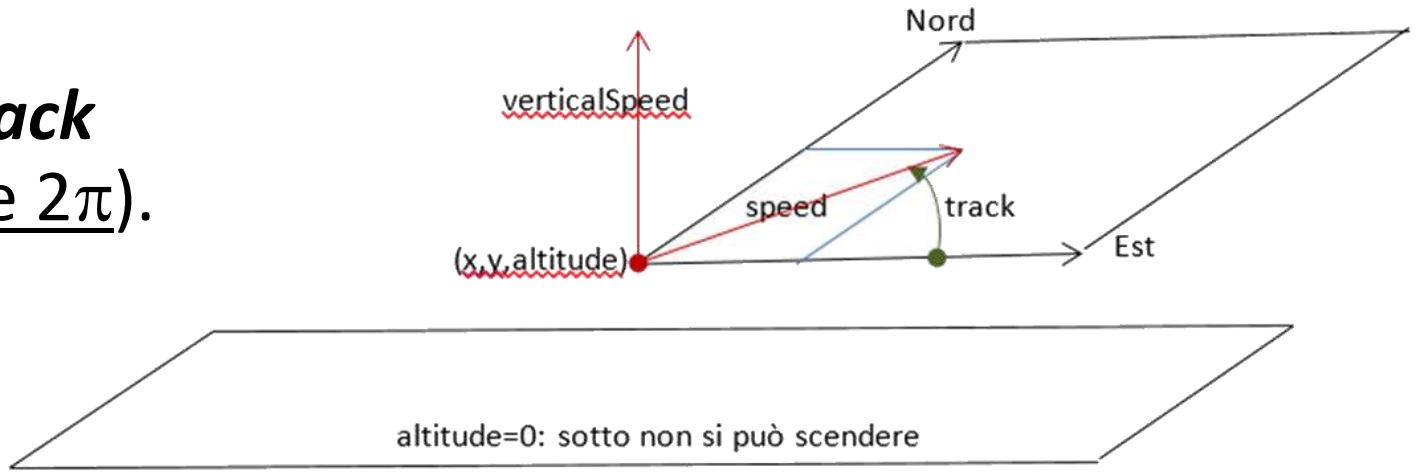
Esercizio *Matita*

- Scrivete ***Matita.maxStelo*** per richiamare il massimo dello stelo (attributo statico).
- Scrivete ***Math.min*** per richiamare il metodo statico min(x,y) della classe Math, che calcola il minimo.
- Includiamo una classe TestMatita per sperimentare la classe Matita: eseguitela e controllate che i risultati siano sensati.

Esercizio *Elicottero*

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)
  - ***verticalSpeed*** (verticale)
- una direzione orizzontale ***track*** (un angolo in radianti tra 0 e 2π).



Esercizio *Elicottero*

La classe ha i seguenti metodi. Usate degli *assert()* per impedire valori non accettabili degli attributi.

- ***(i)*** Il costruttore di Elicottero definisce un elicottero fermo in cielo, date le coordinate (x, y, altitude), con velocità nulle e angolo di direzione nullo.
- ***(ii)*** La classe ha i metodi get per ogni attributo e metodi set per velocità e direzione, ma non per x, y, altitude.
  Non consentiamo a un elicottero di cambiare la posizione se non spostandosi con lo scorrere del tempo.

Esercizio *Elicottero*

- ***(iii)*** Un metodo ***void elapse(double time)*** modifica la posizione dell'elicottero dato il tempo trascorso, in base alle velocità e alla direzione, usando le formule della trigonometria. Quando assegnate il risultato a delle coordinate intere dovrete arrotondarlo, scrivendo: *(int) espressione*.
- Utlizzare ***Math.sin(), Math.cos()*** per i metodi statici per seno e coseno della classe Math. Includiamo una classe ***TestElicottero*** per sperimentare la classe Elicottero: eseguitela (richiede la classe Elicottero) e controllate che i risultati siano sensati.

Esercizio *Elicottero*

$$\sin(a) = \frac{AC}{AB}$$

$$\cos(a) = \frac{BC}{AB}$$