1) Si costruisca una gerarchia di classi per rappresentare veicoli su terra considerando le seguenti entità: **Veicolo, VeicoloAMotore, Ciclomotore, Automobile, Bicicletta**.
   Un **Veicolo** è caratterizzato da una posizione, una velocità iniziale e un'accelerazione (bidimensionali).
   In base ai valori di velocità e accelerazione, un **Veicolo** segue la legge di moto uniformemente accelerato:

$$x = x_0 + v_{0x} * t + a_x * t * t$$
$$y = y_0 + v_{0y} * t + a_y * t * t$$

   (si utilizzi il metodo *muovi(double t)* dove *t* rappresenta il tempo e che aggiorna la posizione del veicolo).
   VeicoloAMotore è un **Veicolo** caratterizzato da un motore con una cilindrata predefinita.
   Ciclomotore è un **VeicoloAMotore** caratterizzato dal numero di telaio (*long*).
   Automobile è un **VeicoloAMotore** caratterizzato dal numero di targa (*String*).
   Bicicletta è un **Veicolo** caratterizzato dal modello (*String*).
   Scrivere un'applicazione che consenta di testare la gerarchia delle classi e di simulare il movimento nel tempo di una **Bicicletta**, un**'Automobile** e un **Ciclomotore** avviati tutti allo stesso istante di tempo.
   (fate semplicemente un ciclo for t che va da 0 a 10 e stampate a schermo le posizioni dei vari oggetti)

   Si modifichi la gerarchia in modo da implementare classi astratte dove possibile. Rendere abstract il metodo *muovi()* della classe **Veicolo**. Si supponga che i **veicoliAMotore** si muovano di moto uniformemente accelerato come visto nell'esercizio precedente, mentre le **Biciclette** seguano la seguente legge di moto:

$$x = x_0 + v_{0x} * t$$
$$y = 0$$

   Si utilizzi una classe **Main** per testare le nuove classi in una lista eterogenea di veicoli che contiene veicoli di vario tipo e in cui, all'interno del ciclo for t che va da 0 a 10, invocate il metodo muovi per tutti gli oggetti della lista e verificate che per le biciclette venga invocato il metodo specializzato.

2) Modificate il problema dell'esercitazione precedente per la gestione della videoteca utilizzando il polimorfismo. In particolare verificate che è sufficiente utilizzare un'unica lista di **Abbonato** per memorizzare sia oggetti **Abbonato** che **AbbonatoPremium** e che l'invocazione del metodo acquista tramite un reference generico **Abbonato** comporta l'esecuzione del metodo dell'oggetto a cui si fa riferimento.
   Si riporta per comodità il testo dell'esercitazione precedente:

   *In un'applicazione per la gestione di una videoteca, i clienti sono memorizzati in oggetti della classe **prg.es05.Persona**. Creare una classe **Abbonato** che estenda la classe **Persona** memorizzando in un'opportuna variabile d'istanza sconto la percentuale di sconto a cui l'abbonato ha diritto su ogni acquisto effettuato. Prevedere opportuni metodi per l'incapsulamento di questa variabile.*
   *Creare inoltre una classe **AbbonatoPremium** che, oltre ad aver diritto allo sconto, ha diritto ad un bonus di 5€ ogni volta che accumuli una spesa complessiva di 100€. Scrivere una classe per testare le classi **Abbonato** e **AbbonatoPremium** che memorizzi due liste di oggetti e abbia funzionalità per aggiungere nuovi abbonati, stampare i dati degli abbonati, gestire il costo degli acquisti in base al tipo di abbonato (ci sarà un metodo acquista che riceve come parametro l'importo dell'acquisto e ritorna l'importo da pagare effettivamente in base allo sconto ed eventuale bonus).*

NOTE PER COMPILAZIONE E TEST A RIGA DI COMANDO IN AMBIENTE LINUX:

`javac -d . es08src/nomeClasse.java`	*compila e genera il bytecode*
`java nomePackage.nomeClasse`	*esegue il bytecode sulla JVM*