LabProg 2011-2012 - TESTO ESAME
Andrea Trentini - DICo - UniMi
01 febbraio 2012 (b)

Avvertenza. Non è ammesso l'uso delle classi del package *prog.io* allegato al libro di testo del corso.

1 Testo esame

Lo scopo è realizzare un modello di "taxi" minimale. Le classi da realizzare sono le seguenti (dettagli nelle sezioni successive):

- *Taxi*: il veicolo che trasporta passeggeri
- *Passeggero*: una persona a bordo
- *Main*: contiene il solo metodo *main()* con alcune invocazioni di test (importante è riuscire a testare TUTTI i metodi almeno una volta)

Tali classi dovranno esporre almeno i metodi specificati nelle sezioni seguenti. Eventuali metodi di servizio possono essere aggiunti a piacimento. Ogni classe deve avere il *toString()* che rappresenti lo stato delle istanze e i **costruttori** adeguati per gli attributi che vengono dichiarati. Dato che gli attributi devono essere privati creare opportunamente i metodi di accesso (set&get). Si suggerisce, anche dove non segnalato, di utilizzare, se esistenti e se applicabili, le classi parametriche (es. *ArrayList<E>* invece di *ArrayList*). Alcuni controlli di coerenza vengono suggeriti nel testo, potrebbero essercene altri a discrezione. I tipi di ritorno possono essere variati (ad es. boolean invece di void se si vuole ottenere un feedback sul successo dell'operazione) previa autorizzazione del docente. Consiglio di posporre l'implementazione dei controlli di coerenza, fatelo come ultima operazione, prima realizzate un sistema funzionante.

1.1 Passeggero

Rappresenta un passeggero, ha (attributi) peso, destinazione e nr. bagagli.
Deve inoltre esporre i seguenti metodi **pubblici**:

- *String getDestinazione()*: fornisce la destinazione desiderata
- *int getPeso()*: restituisce il peso
- *int nrBagagli()*: restituisce il nr. di bagagli al seguito

1.2 Taxi

Rappresenta un taxi. Deve memorizzare internamente il nr. di km percorsi e il tempo trascorso perché dovrà calcolare la tariffa in base ad essi.
Deve esporre i seguenti metodi **pubblici**:

- *boolean sale(Passeggero p)*: sale un passeggero
- *void partenza()*: partenza del viaggio
- *void tragitto()*: metodo che serve a incrementare[1] (va bene anche random) il tempo trascorso e i chilometri percorsi, per poi poter calcolare la tariffa finale (cfr. esempio)
- *double tassametro()*: restituisce il valore del tassametro, la tariffa (la funzione di calcolo è a discrezione dello studente) deve tenere conto dei chilometri percorsi, del tempo passato e del nr. di bagagli trasportati.

Esempio (con lacune parametriche):

```
Taxi t=new Taxi(.......);
Passeggero p=new Passeggero(.......);
t.sale(p);

t.partenza();

t.tragitto();
t.tragitto();
t.tragitto();

// tariffa parziale
System.out.println(t.tassametro());

t.tragitto();
t.tragitto();

// tariffa finale (deve essere maggiore della precede
System.out.println(t.tassametro());
```

1.3 Main

Deve contenere il metodo *main()* in cui vanno istanziate le classi realizzate e opportunamente testate.

2 Consegna

Ricordo che le classi devono essere tutte *public* e che vanno consegnati tutti i file *.java* prodotti. NON vanno consegnati i *.class* Per la consegna, eseguite l'upload dei singoli file sorgente dalla pagina web: http://upload.dico.unimi.it.

[1] non potendo usare i thread dobbiamo simulare una azione "parallela"