Antichi Commerci

Progetto di Programmazione II del 18 Gennaio 2017

Modalità di consegna

Per consegnare:

- all'inizio della prova, compilate il form con i vostri dati;
- per consegnare, aprite un terminale e cambiate la directory corrente (`cd`) raggiungendo la directory che contiene i vostri sorgenti; potete verificarlo dando il comando `ls` ; poi, date il comando

  ```
  /mnt/pboldi00/consegna *.java
  ```

- potete effettuare più consegne, ma verrà corretta solo l'ultima avvenuta con successo.

Vi ricordo che la documentazione delle classi standard è disponibile puntando il browser su

```
file:///usr/share/javadoc/java/index.html
```

Introduzione

In questo esame vi chiederemo di scrivere un insieme di classi e interfacce per la simulazione dell'evoluzione di civilità.

Il progetto verrà valutato prima di tutto in base al suo corretto funzionamento rispetto ai requisiti qui descritti; quindi, in base alla pulizia del codice ed alla sua documentazione (si consiglia di commentare il codice, in particolare anche scrivendo i commenti `javadoc`).

Si consiglia di completare in maniera corretta la **Parte 1** prima di iniziare la **Parte 2**, e di completare correttamente entrambe prima di iniziare la **Parte 3**.

Classi e interfacce: parte 1

Descriviamo ora nei dettagli come realizzare le classi necessarie al progetto. Se lo ritenete, potete apportare cambiamenti alle segnature dei metodi proposti ed anche – se lo ritenete – alla struttura delle classi, restando ovviamente all'interno dei requisiti esposti nel paragrafo introduttivo.

N.B.: Oltre ai metodi qui indicati, ricordatevi di aggiungere ad ogni classe un valido metodo `toString()`

e, dove lo ritenete, anche un metodo `equals` e un metodo `hashCode` (con le segnature appropriate e che soddisfino i contratti previsti).

Classe Risorsa

Una classe che rappresenta una singola unità di una risorsa presente nel terreno. Contiene i campi `nome` (la stringa rappresentante il nome) e `prezzo` (un numero intero che indica a quanto viene venduta un'unità di questa risorsa); per esempio, "Petrolio" potrebbe avere un prezzo di 100, mentre "Carbone" di 30.

Questa classe deve implementare l'interfaccia `Comparable<Risorsa>` con questa semantica: una risorsa è maggiore di un'altra se e solo se il suo prezzo è maggiore e (a parità di prezzo) se il suo nome segue, nell'ordine alfabetico, il nome dell'altra.

Classe astratta Citta

Rappresenta una città. Una città avrà un nome e un metodo astratto `produce(Civilta c)` che restituisce `void` ma che attribuisce alla civiltà `c` i "prodotti" di questa città, che verranno specificati dalle sue sottoclassi.

Classe CittaIndustriale

È una città che possiede una `Risorsa` (contenuta nel campo `risorsa` di questa classe). Il costruttore ha segnatura `CittaIndustriale(String nome, Risorsa r)`.

Il metodo `produce` aggiungerà alla civiltà in questione la `risorsa`, richiamando il metodo `aggiungiRisorsa` della classe `Civilta`.

Classe CittaEconomica

È una città che aggiunge alla civiltà 1000 unità di denaro ogni volta che viene chiamato il metodo `produce`. Il costruttore ha come argomento solo il nome.

Classe Civilta

Rappresenta una civiltà, attraverso il suo `nome` (p.e. "Impero Inglese") e una lista di `Citta`. Avrà un campo intero `tesoro` che indica la quantità di denaro attualmente posseduta da questa civiltà. Avrà anche una lista (o un array) di `Risorsa` chiamato `stock`, contenente le risorse prodotte nelle città di questa civiltà.

Il costruttore chiederà solo il nome. Una civiltà appena costruita non avrà nè città, nè risorse, nè tesoro.

Deve contenere i seguenti metodi:

- un metodo `Citta fondaCitta(String nome, char tipo)` che crea (aggiunge alla lista delle città e restituisce) una città con nome `nome`, di tipo `CittaEconomica` se `tipo` è `'e'` o di tipo `CittaIndustriale` se tipo è `'i'`. Se viene creata una `CittaIndustriale`, dovrete anche creare una risorsa: per farlo,
  - potete scegliere casualmente una risorsa da una lista di nomi come `new String[] { "Petrolio", "Oro", "Grano", "Riso" }` e assegnargli un prezzo casuale.
  - Oppure, potete seguire le indicazioni contenute nella **Parte 2** di questo tema.
- un metodo `aggiungiRisorsa(Risorsa r)` che aggiunge `r` allo `stock`
- un metodo `aggiungiDenaro(int d)` che aggiunge la quantità `d` di denaro (che può anche essere negativa) al `tesoro`
- un metodo `faiProdurre()` che richiama il metodo `produce` di ciascuna città, passando come argomento questa civiltà
- un metodo `boolean vendiRisorseA(Civilta altra)`, che consente a questa civiltà di vendere risorse ad un'altra civiltà `c` in questo modo:
  - crea una nuova lista locale `nuovoStock` che alla fine dell'esecuzione del metodo conterrà il nuovo stock di questa civiltà, e che verrà assegnata all'attributo `stock` al termine, prima di uscire dal metodo
  - scorre tutte le risorse dello `stock`
  - Per ogni risorsa `r`, controlla se questa civiltà possiede `r` *in almeno due copie* (ovvero, `r` deve essere presente due volte con stesso nome e prezzo nell'array `stock`)
  - Quindi, controlla se la civiltà `altra` possiede la risorsa `r`
  - Se non la possiede, questa civilità vende a `altra` la risorsa: cioè, ne elimina una copia dal proprio `stock`, la aggiunge a quello di `altra`, e sposta denaro pari al prezzo di `r` dalla civiltà `altra` a questa. Il denaro di una civiltà può anche andare in negativo.
  - Se invece la possiede, oppure se la risorsa non è presente in due copie, la risorsa `r` viene messa nel `nuovoStock` (in quanto non è stata venduta).
  - Il metodo restituisce `true` sse vi è stato commercio tra le due civiltà.

  Se lo ritenete, potete scomporre questo metodo in più sottometodi (per esempio, un metodo `possiede` e un metodo `possiedeDoppio`).

Classe Storia

Ha un costruttore che accetta un array di `Civilta`, che sarà un campo della classe.

Ha un metodo `Civilta commercia(int n)` che, per `n` volte, fa quanto segue:

- chiama il metodo `faiProdurre` di ogni civiltà.

- Per ogni civiltà `c`, chiama il metodo `vendiRisorseA` ogni altra civiltà `k`. Se vi è effettivamente un commercio da `c` a `k`, `c` non venderà risorse a nessun altro.
- Alla fine, il metodo restituisce la civiltà con più denaro.

Contiene un metodo `main(String[] args)` che interagendo con l'utente crea una sequenza di civiltà (fino all'inserimento della stringa vuota come nome di civiltà). Per ogni civilità, deve chiedere all'utente quante nuove città deve creare; per ogni città, deve chiederne il nome e il tipo, in questo modo:

```
Inserisci il nome di una nuova civiltà:
Italia
Inserisci il nome di una nuova città, o "stop" per non inserire altre città.
Milano
Di che tipo dev'essere? [i/e]
i
Italia ha fondato Milano, una città di tipo Industriale, che controlla la risorsa
Zafferano (prezzo 100).
```

Dopo la fase di creazione, deve chiedere all'utente un numero `n` e chiamare il metodo `commercia` con questo intero. Alla fine, stampate il risultato del metodo `commercia`.

Classi e interfacce: parte 2

Ogni volta che viene invocato il costruttore di `Risorsa`, la risorsa creata viene aggiunta ad una lista statica `risorseCreate` all'interno della stessa classe `Risorsa`. La classe `Risorsa` deve quindi avere un metodo `risorsaCasuale()` che restituisce una risorsa casuale tra quelle create.

Ora:

- modificate il metodo `main` della classe `Storia` in modo che, prima di creare le città, chieda all'utente di creare delle risorse:

Per esempio:

```
Inserisci il nome di una nuova risorsa, o "stop" per non inserire altre risorse.
Zafferano
Qual è il suo prezzo?
100
```

- modificate il metodo `fondaCitta` di `Civilta` in modo che, se deve creare una risorsa casuale, usi il metodo `risorsaCasuale()`.

Classi e interfacce: parte 3

Create queste nuove classi:

- `RisorsaDerivata` è una sottoclasse di `Risorsa`. Ha un costruttore con questa segnatura: `RisorsaDerivata(String nome, int prezzo, Risorsa originale)`. La risorsa derivata sarà una risorsa prodotta a partire dalla `Risorsa originale`; per esempio la `"Plastica"` potrebbe essere prodotta con `"Petrolio"` come risorsa originale.
- Modificate il metodo `produce` di `CittaIndustriale` in modo che, se la `Risorsa` della città è del tipo `RisorsaDerivata`, il metodo `produce` dovrà controllare (prima di aggiungere la risorsa derivata alla civiltà) che questa possieda la risorsa `originale`. Se non è così, deve lanciare una `RisorsaMancanteException` (create questo tipo di `RuntimeException`).
- Nel metodo `faiProdurre` di `Civilita`, catturate la `RisorsaMancanteException` e fate stampare un messaggio. Dopodiché, il metodo `faiProdurre` deve continuare come se non fosse accaduto nulla.

Per esempio:

```
La città Gorgonzola non può produrre la risorsa derivata "Formaggio (prezzo 20)" p
erchè non è disponibile la risorsa originale "Latte (prezzo 10)".
```

- Nel metodo `main` di `Storia`, dopo la creazione di una risorsa *c*, viene chiesto all'utente se desidera creare una `RisorsaDerivata` che abbia *c* come risorsa originale.

Per esempio:

```
Inserisci il nome di una nuova risorsa, o "stop" per non inserire altre risorse.
Latte
Qual è il suo prezzo?
10
Vuoi creare una risorsa derivata da Latte? [y/n]
y
Qual è il suo nome?
Formaggio
Qual è il suo prezzo?
20
```

Esempio di esecuzione

Di seguito un esempio completo di input e output, relativo ad un progetto completo delle parti 1, 2 e 3.

```
Inserisci il nome di una nuova risorsa, o "stop" per non inserire altre risorse.
Grano
Qual è il suo prezzo?
50
Vuoi creare una risorsa derivata da Grano (prezzo 50)? [y/n]
```

```
y
Qual è il suo nome?
Pasta
Qual è il suo prezzo?
100
Inserisci il nome di una nuova risorsa, o "stop" per non inserire altre risorse.
Carbone
Qual è il suo prezzo?
250
Vuoi creare una risorsa derivata da Carbone (prezzo 250)? [y/n]
n
Inserisci il nome di una nuova risorsa, o "stop" per non inserire altre risorse.
Petrolio
Qual è il suo prezzo?
500
Vuoi creare una risorsa derivata da Petrolio (prezzo 500)? [y/n]
n
Inserisci il nome di una nuova risorsa, o "stop" per non inserire altre risorse.
stop
Inserisci il nome di una nuova civiltà:
Italia
Inserisci il nome di una nuova città, o "stop" per non inserire altre città.
Roma
Di che tipo dev'essere? [i/e]
i
Italia ha fondato Roma, una città di tipo Industriale, che controlla la risorsa Ca
rbone (prezzo 250)
Inserisci il nome di una nuova città, o "stop" per non inserire altre città.
Milano
Di che tipo dev'essere? [i/e]
e
Italia ha fondato Milano, una città di tipo Economico
Inserisci il nome di una nuova città, o "stop" per non inserire altre città.
Torino
Di che tipo dev'essere? [i/e]
i
Italia ha fondato Torino, una città di tipo Industriale, che controlla la risorsa
Grano (prezzo 50)
Inserisci il nome di una nuova città, o "stop" per non inserire altre città.
Napoli
Di che tipo dev'essere? [i/e]
i
Italia ha fondato Napoli, una città di tipo Industriale, che controlla la risorsa
Pasta (prezzo 100)
Inserisci il nome di una nuova città, o "stop" per non inserire altre città.
Bologna
Di che tipo dev'essere? [i/e]
i
```

```
Italia ha fondato Bologna, una città di tipo Industriale, che controlla la risorsa
 Grano (prezzo 50)
Inserisci il nome di una nuova città, o "stop" per non inserire altre città.
stop
Inserisci il nome di una nuova civiltà:
Svizzera
Inserisci il nome di una nuova città, o "stop" per non inserire altre città.
Ginevra
Di che tipo dev'essere? [i/e]
e
Svizzera ha fondato Ginevra, una città di tipo Economico
Inserisci il nome di una nuova città, o "stop" per non inserire altre città.
Zurigo
Di che tipo dev'essere? [i/e]
e
Svizzera ha fondato Zurigo, una città di tipo Economico
Inserisci il nome di una nuova città, o "stop" per non inserire altre città.
stop
Inserisci il nome di una nuova civiltà:
Grecia
Inserisci il nome di una nuova città, o "stop" per non inserire altre città.
Atene
Di che tipo dev'essere? [i/e]
i
Grecia ha fondato Atene, una città di tipo Industriale, che controlla la risorsa C
arbone (prezzo 250)
Inserisci il nome di una nuova città, o "stop" per non inserire altre città.
Inserisci il nome di una nuova civiltà:
Quanti cicli di commercio andranno simulati?
Italia possiede 0 $.
Svizzera possiede 0 $.
Grecia possiede 0 $.
Ciclo #0...
Italia (1050 $) ha venduto Grano (prezzo 50) a Svizzera (1950 $)
Italia possiede 1050 $.
Svizzera possiede 1950 $.
Grecia possiede 0 $.
Ciclo #1...
Italia (2300 $) ha venduto Carbone (prezzo 250) a Svizzera (3700 $)
Italia (2400 $) ha venduto Pasta (prezzo 100) a Svizzera (3600 $)
Italia possiede 2400 $.
Svizzera possiede 3600 $.
Grecia possiede 0 $.
Ciclo #2...
Italia (3450 $) ha venduto Grano (prezzo 50) a Grecia (-50 $)
Italia (3550 $) ha venduto Pasta (prezzo 100) a Grecia (-150 $)
Italia possiede 3550 $.
Svizzera possiede 5600 $.
```

```
Grecia possiede -150 $.
Ciclo #3...
Italia possiede 4550 $.
Svizzera possiede 7600 $.
Grecia possiede -150 $.
Ciclo #4...
Italia possiede 5550 $.
Svizzera possiede 9600 $.
Grecia possiede -150 $.
La civiltà più ricca è Svizzera (9600 $)
```

Esempio di input

Di seguito l'input che ha generato l'esecuzione precedente. Potete impiegarlo per testare il vostro programma senza dover riscrivere ogni volta tutto, sfruttando la *pipe* di Unix (`|`).

```
Grano
50
y
Pasta
100
Carbone
250
n
Petrolio
500
n
stop
Italia
Roma
i
Milano
e
Torino
i
Napoli
i
Bologna
i
stop
Svizzera
Ginevra
e
Zurigo
e
stop
Grecia
Atene
i
stop

15
```