HOMEWORK 3

Java Collection Framework

Esercizio 0

- Chi non avesse concluso la scrittura dei test, lo faccia in questo homework, prima di fare le modifiche al codice

Esercizio 1

- Sostituire tutti gli array utilizzati nelle classi **`Stanza`** e **`Borsa`** con opportune collezioni (**`List`**, **`Set`**, **`Map`**)
  - Assumere che non possano esistere due oggetti **`Attrezzo`** con lo stesso nome in stanze dello stesso **`Labirinto`**
  - Eliminare il vincolo che al max 10 attrezzi possano essere collocati nella borsa (ma mantenere quello sul peso max)
  - Provare ad usare (in alternativa) **`List`** e **`Map`** per implementare la collezione di attrezzi nella borsa (vedi POO-14). Quale risulta più semplice?
- Queste modifiche impattano solo sull'implementazione: dopo averle effettuate confermare con i test sviluppati nei precedenti homework la correttezza del codice dopo le modifiche

Esercizio 2

- Rivisitare il codice delle nuove tipologie di stanze introdotte nel precedente homework:
  - La stanza *buia*: se nella stanza non è presente un attrezzo con un nome particolare (ad esempio "lanterna") il metodo **`getDescrizione()`** di una stanza buia ritorna la stringa "*qui c'è un buio pesto*"
  - La stanza *bloccata*: una delle direzioni della stanza non può essere seguita a meno che nella stanza corrente non sia presente un oggetto di un certo nome (ad es. «piedediporco»)
- Queste classi subiscono modifiche consequenziali ai cambiamenti effettuati nella implementazione di **`Stanza`**?
- Le modifiche che subiscono sono le stesse nelle due versioni di **`Stanza`** (*con* e *senza* campi **`protected`**) ipotizzate nel precedente homework?

Esercizio 2 (continua)

- Ampliare i test di tutte le classi nella gerarchia che ha radice in **`Stanza`**:
  - **`Stanza`**, **`StanzaMagica`**, **`StanzaBuia`** e **`StanzaBloccata`**

- Eliminare dal codice delle classi **`Borsa`**, **`Stanza`**, **`StanzaMagica`**, **`StanzaBuia`** e **`StanzaBloccata`** (estensioni di **`Stanza`**) **ogni** ciclo di ricerca da un collezione (ad es. di un attrezzo per nome, o di una stanza per direzione)

✓ affidarsi invece sempre e solo alle funzionalità offerte dai metodi presenti nelle classi del JCF

Esercizio 3

- Aggiungere alla classe **`Borsa`** dei metodi di interrogazione del suo contenuto:
  - **`List<Attrezzo> getContenutoOrdinatoPerPeso();`**
    restituisce la lista degli attrezzi nella borsa ordinati per peso e quindi per nome a parità di peso
  - **`List<Attrezzo> getContenutoOrdinatoPerNome();`**
    restituisce la lista degli attrezzi nella borsa ordinati per nome
  - **`Map<Integer,Set<Attrezzo>> getContenutoRaggruppatoPerPeso()`**
    restituisce una mappa che associa un intero (rappresentante un peso) con l'insieme (comunque *non* vuoto) degli attrezzi di tale peso: tutti gli attrezzi dell'insieme che figura come valore hanno lo stesso peso pari all'intero che figura come chiave
- Utilizzare questi metodi per migliorare la stampa del contenuto della **`Borsa`** (ad es. comando *guarda* >>)

Esercizio 3 (notazione es.)

- Si utilizzi *piombo:10* per indicare un riferimento ad un oggetto **`Attrezzo`** di nome "piombo" e di peso 10
- Per brevità scriviamo *piombo* al posto di *piombo:10* quando non è utile ripetere il dettaglio sul peso
- Si utilizzi

– `{` *piombo*, *piuma*, *libro*, *ps* `}` per indicare un **`Set`** di attrezzi

– `[` *piuma*, *libro*, *ps*, *piombo* `]` per indicare una **`List`** di attrezzi

– `(` 5 `,` `{` *libro*`,`*ps* `}` `)` per indicare una coppia chiave/valore di una **`Map<Integer,Set<Attrezzi>>`**

Esercizio 3 (esempio)

- Si consideri una **`Borsa`** contenente questo insieme di riferimenti ad oggetti **`Attrezzo`**:

  { *piombo:10*, *ps:5*, *piuma:1*, *libro:5* **}**

- Allora i metodi di cui prima, invocati sullo questa **`Borsa`**:

– **`List<Attrezzo> getContenutoOrdinatoPerPeso();`**

  deve restituire: **[** *piuma*, *libro*, *ps*, *piombo* **]**

– **`List<Attrezzo> getContenutoOrdinatoPerNome();`**

  deve restituire: **[** *libro*, *piombo*, *piuma*, *ps***]**

– **`Map<Integer,Set<Attrezzo>>`**

  `getContenutoRaggruppatoPerPeso()`

  deve restituire una **`Map`** contenente tutte e sole le seguenti coppie chiave/valore: **(**1, **{** *piuma* **}** **)** ; **(**5, **{** *libro*, *ps* **}** **)** ; **(**10, **{** *piombo* **}** **)**

Esercizio 4

- Utilizzando JUnit, scrivere una batteria di test-case *minimali* per verificare la correttezza delle soluzioni prodotte nell'esercizio precedente
  - *minimali*: ovvero facenti utilizzo delle collezioni più semplici possibile utili alla verifica (piccole e con **`Attrezzi`** di nomi/pesi in configurazioni *a loro volta minimali*)

- Solo dopo aver completato il precedente punto, ampliare i test-case di sopra con altri non minimali per migliorarne la copertura

TERMINI E MODALITA' DI CONSEGNA

- La soluzione deve essere inviata al docente entro le 21:00 del 19 maggio 2015 come segue:
  - Svolgere in gruppi di max 2 persone
  - Esportare (con la funzione File->Export di Eclipse) il progetto realizzato nel file **homework3.zip**
  - Inviare il file **homework3.zip** all'indirizzo di posta elettronica poo.roma3@gmail.com
  - Nel corpo del messaggio riportare eventuali malfunzionamenti noti, ma non risolti
  - L'oggetto (subject) deve iniziare con la stringa **[2015-HOMEWORK3]** seguita dalle matricole mittenti
  - Ad es.: **[2015-HOMEWORK3] 412345 454321**