Ingegneria del Software
Esercitazione

30 Ottobre 2023

Davide Yi Xian Hu

Email: davideyi.hu@polimi.it

POLITECNICO
MILANO 1863

Esercizio 1 / Cron

Si progetti un package che offra un "demone temporale" simile a cron di Unix.

- 👉 Definire l'interfaccia **Action** con metodo **execute( )**.
  Definire la classe **PrintAction** con metodo **execute( )** che stampa a schermo.

- 👉 L'utente del package deve poter creare un **demone** (in inglese daemon), registrare presso di lui una serie di coppie <orario, azione da compiere>.

- 👉 Il demone temporale, una volta avviato, deve eseguire le azioni registrate all'orario prestabilito.

- 👉 Si supponga che **non si possano registrare più di 10 azioni**, che ogni azione debba venir eseguita una volta soltanto e che una volta eseguite tutte le azioni cron termini la sua esecuzione.

- 👉 Si può interpretare l'orario di esecuzione come "orario indicativo": viene garantito che l'azione viene eseguita *dopo* l'orario specificato.

Esercizio 2 / Runtime Type Checking

```
class Father {
  int x;
  public Father(int x) {
    this.x = x;
  }
  public int m(Father f) {
    return (f.x - this.x);
  }
}

class Son extends Father {
  int y;
  public Son(int x, int y) {
    super(x);
    this.y = y;
  }
  public int m(Father f) {
    return 100;
  }
  public int m(Son s) {
    return super.m(s) + (s.y - this.y);
  }
}
```

```
public class MainClass {
  public static void main(String args[]) {
    Father f1, f2; Son s1, s2;
    f1 = new Father(3);
    f2 = new Son(3,10);
    System.out.println(f1.m(f2));            /*1*/
    System.out.println(f2.m(f1));            /*2*/
    s1 = new Son(4,21);
    System.out.println(s1.m(f1) + s1.m(f2)); /*3*/
    System.out.println(f1.m(s1) + f2.m(s1)); /*4*/
    s2 = new Son(5,22);
    System.out.println(s1.m(s2));            /*5*/
  }
}
```

Recap / Exceptions

Compile-time Exceptions (Checked Exceptions)

- 👉 Esempi: IOException, FileNotFoundException, ClassNotFoundException.
- 👉 Gestione: Con **try-catch** e dichiarati nella firma del metodo.

Runtime-time Exceptions (Unchecked Exceptions)

- 👉 Esempi: NullPointerException, ArithmeticException, IndexOutOfBoundsException.
- 👉 Gestione: Non e' necessario catturarli e possono essere propagati.

Esercizio 3 / Exceptional Bank

Si progetti un applicazione che simula le operazioni di una banca.

- 👉 Definire la classe **BankAccount** caratterizzato da un identificativo **id** e un bilancio **balance**.
- 👉 Definire i metodi **withdraw(double amount)** per prelevare denaro e **deposit(double amount)** per depositare denaro.
- 👉 Definire l'eccezione **NegativeAmountException** se qualcuno cerca di prelevare e depositare quantità' negative di denaro.
- 👉 Definire l'eccezione **InsufficientFundsException** se qualcuno cerca di prelevare più' denaro di quanto possiede.

Esercizio 4 / SafeStack

Si progetti la classe SafeStack.

- 👉 Definire il metodo **SafePush** che gestisce correttamente i casi limite dell'inserimento di nuovi elementi.
- 👉 Definire il metodo **SafePop** che gestisce correttamente i casi limite dell'estrazione degli elementi nello stack.
- 👉 Rendere la classe Stack **tipizzabile** con i Generics
- 👉 Rendere **iterabile** la classe Stack (con Iterator<T>)

Esercizio 5 / Database

Si progetti la classe Database.

- 👉 Definire la astratta classe **Entity** caratterizzato da un identificativo **id** e un metodo **displayDetails( )**.
- 👉 Definire **Person** and **Product** che estendono Entity. Person ha attributi nome e cognome, mentre Product ha attributi nome e prezzo. Fare override del metodo displayDetails( ).
- 👉 Definire la classe **Database<T extends Entity>** che salva una lista di Entity. Definire i metodi **add**, **remove**, **find (id)**, e **displayAll**. Fare overloading del metodo add per accettare anche una lista.
- 👉 Rendere **iterabile** la classe Database (con Iterator<T>).
- 👉 Rendere **clonabile** la classe Database con (Cloneable).

Esercizio 5+ / Database+

(Extra) Si progetti la classe Database+.

- 👉 Introdurre il metodo **update(id, newEntity)** che permette di aggiornare un elemento del database.
- 👉 Gestire i casi limite con le eccezioni (ad esempio se si aggiunge una entità' con lo stesso id due volte nel database)
- 👉 Implementare un iteratore custom per filtrare e ordinare le entita' mentre si itera.