Consegna 4
progettazione soluzione/algoritmo

Lorenzo Dentis, lorenzo.dentis@edu.unito.it

20 aprile 2023

1 Domande

- Leggete l'articolo Programming Patterns and Design Patterns in the Introductory Computer Science Course e scrivete un breve resoconto sull'approccio proposto:siete d'accordo con l'approccio proposto? lo adottereste? immaginate di proporlo in una classe in cui insegnate: quali sono le potenziali difficoltà?
- riportate il testo del problema analizzato per l'attività 2) Progettazione soluzione/algoritmo e la soluzione che avete progettato: suddivisione in sottoproblemi,pattern algoritmici, elementari e ruoli delle variabili.
- descrivete anche eventuali critiche (es. sulla formulazione del problema), problemi e considerazioni emerse durante l'attività

2 Risposte

2.1 Commento sull'articolo

In generale, concordo molto con il metodo presentato nell'articolo [1], anche perché è simile al modo in cui io stesso preferisco imparare nuovi argomenti. Nella sezione 4.2, subito dopo *Intent and Motivation*, viene presentata la sezione *Problem Examples*, in cui sono mostrati alcuni esempi di utilizzo del pattern. Trovo molto utile vedere prima un esempio di soluzione in atto quando si affronta un nuovo problema.

In futuro, gli studenti potranno far riferimento a questi pattern quando si troveranno di fronte a problemi simili, conoscendo già una possibile soluzione. Inoltre, questo metodo favorisce il riuso. Personalmente, durante il mio secondo anno di università, ho dovuto analizzare un file CSV molto complesso in Java e ho scritto del codice che ancora oggi utilizzo ogni volta che mi trovo ad avere in input un file con tale estensione.

Tutti i pattern presentati nell'articolo sono molto comuni, tanto che, prima di averli letti, li avevo già inseriti nella mia prima bozza dell'attività 2. In modo inconscio, ho scritto una domanda che richiede di risolvere due dei problemi

presentati nell'articolo, volendo presentare un problema "semplice" e "comune".

In aula, una collega ha sostenuto che questo metodo potrebbe ostacolare l'apprendimento del *problem solving* e, in generale, concordo con questa affermazione. Cercare di categorizzare ogni problema e fornire la soluzione agli studenti rende la soluzione molto "meccanica" e non aiuta a sviluppare la capacità di "cercare" un modo di risolvere il problema.
Gli autori cercano di guidare gli studenti per evitare la situazione "I do not even know where to start", ma spesso capire da dove partire per risolvere un problema è una skill a sé, un'abilità che va sviluppata e che risulta utile nella vita quotidiana di uno sviluppatore.

Tuttavia, l'articolo non propone questo metodo come una panacea per insegnare a tutti gli sviluppatori, ma, citando ancora la sezione 4.2, *"The patterns are written to guide a novice with little or no programming experience"*. Quindi, si cerca di insegnare a programmare a persone con scarsa esperienza, che non hanno ancora le basi per iniziare a sviluppare la capacità di "cercare da soli una soluzione". In questo senso, credo che questo metodo sia eccellente.

2.2 Attività 2

2.2.1 Problema posto da Dentis

`Lo Chef Tony vuole che la sua cucina sia sempre in ordine e soprattutto sapere quali ingredienti ha a disposizione.`
`Quindi ha suddiviso gli ingredienti in 5 categorie:` *`pasta,sugo,carne,verdure e spezie.`*
`Quando arriva un nuovo carico di materie prime (il numero di ingredienti in un carico può variare) Tony vuole che queste vengano separati automaticamente nelle 4 categorie e gli venga restituito in output quanti elementi di ogni categoria ci sono nel carico appena giunto.`

Il problema può essere complicato come nella seguente variante: `Il programma deve tenere conto che Tony non è l'unico Chef di Cat&Ring, nella dispensa potrebbero esserci ingredienti rimasti, quindi deve chiedere allo Chef la quantità rimasta e comportarsi di conseguenza.`

2.2.2 Soluzione del problema posto da Dentis

Algorithm 2.1: La cucina di Chef Tony (pt.1)

```
1   input: int n, list carico
2   output: int pasta, int sugo, int carne, int verdure, int spezie
3   begin
4     i ← 0
5     pasta ← 0
6     sugo ← 0
7     carne ← 0
8     verdure ← 0
9     spezie ← 0
10    while i ≤ n
11       switch (categoria del prodotto i-esimo presente nel carico)
12         case pasta:
13              pasta ← pasta + 1
14         case sugo:
```

```
15              sugo ← sugo + 1
16          case carne:
17              carne ← carne + 1
18          case verdure:
19              verdure ← verdure + 1
20          case spezie:
21              spezie ← spezie + 1
22      i ← i + 1
23    end
24    return pasta, sugo, carne, verdure, spezie
25  end
```

RIVEDERE LEZIONE 24.10

2.2.3 Problema posto da D'Angelo

Tra meno di 30 giorni, si svolgerà a Liverpool la sessantasettesima edizione dell'Eurovision Song Contest!
Vista una serie di problemi avvenuti nelle votazioni dello scorso anno Martin Österdahl, produttore esecutivo della manifestazione, ha deciso di tenere sempre una copia della classifica in un pc separato e inattaccabile da hackers.
Il tuo compito è, dato il singolo paese, vedere i punteggi che quella nazione assegna ad ogni altro partecipante in gara e restituire la classifica completa.
I punteggi che possono essere assegnati sono:

- 0 (default)
- 1 <= punteggio <= 8
- 10
- 12

2.2.4 Soluzione del problema posto da D'Angelo

Algorithm 2.1: Calcolo punteggi

```
1       input: lista_voti (che paese ha votato ogni paese).
2       output: classifica
3       begin:
4       num_paesi ← lunghezza(lista_voti)
5       classifica ← vettore[num_paesi] //vettore con tutti i paesi ed i voti ricevuti
6       foreach voto in lista_voti
7               foreach paese in classifica
8                       if voto uguale paese
9                               classifica[paese] ← 12 + classifica[paese]
10                              break
11                      end
12              end
13      end
14      return classifica
```

Pattern elementari

- Guarded-Action: riga 8
- Process-All-item: riga 6, scorriamo tutta la lista dataci in input

- process-Items-Unitl-Done: riga 10. Mentre si sta scorrendo il vettore della classifica se si trova il paese a cui bisogna assegnare il voto si può interrompere la ricerca.

Pattern Algoritmici

- 3.7 Traversal Pattern. In particolare *simple linear traversal* dato che stiamo scorrendo due vettori
- 3.5 Repetition Patterns. Dato che abbiamo due loop annidiati.
- 3.8 Cumulative Result Patterns. usiamo una composizione di 3 pattern, due repetition pattern (i loop annidiati) ed un accumulatore (il vettore classifica che tiene conto dei punteggi). citando il documento [1] *he goal is to traverse some collection of data, collecting partial information into some accumulator entity and presenting the composite result at the end.*

Ruoli delle variabili

- Valore fissato: la variabile *num_paese* serve solo in fase di inizializzazione per fissare la dimensione della classifica. In realtà se ne sarebbe anche potuto fare a meno.
- Contatore o Indice: la variabile *paese* svolge anche il ruolo di indice del vettore $classifica$
- Accumulatore: la variabile $classifica$ è un vettore di accumulatori.

2.3 Considerazioni emerse

La versione originale del problema prevedeva una struttura dati differente: una collection di oggetti (paesi) contenenti una lista di voti. Ogni paese quindi avrebbe avuto una sua lista di punteggi assegnati ad ogni altro paese.

Il problema è che la struttura dati necessaria a rappresentare tale collezione è un po' troppo complicata per il target, dato che necessiterebbe la nozione di *object* oppure l'uso di una matrice bidimensionale.

Riferimenti bibliografici

[1] Viera K. Proulx. *Programming Patterns and Design Patterns in the Introductory Computer Science Course*. URL: `http://www.ccs.northeastern.edu/home/vkp/Papers/Patterns-sigcse2000.pdf`.