Esercizi 9 e 10 dicembre 2019

Programmazione Avanzata

a.a. 2019-20

1

Esercizio 1

- Immaginiamo che un pacco venga inviato all'ufficio postale: il pacco puo` essere ordinato, poi spedito all'ufficio postale e quindi ricevuto dal destinatario.
- Vogliamo scrivere il suo stato ogni volta che questo cambia: lo stato iniziale e` ordinato. La classe Pacco ha il metodo _succ per passare allo stato successivo e _pred per passare a quello precedente. Lo stato ordinato non ha stati che lo precedono; lo stato ricevuto non ha stati che vengono dopo di esso.
- L'approccio piu` semplice sarebbe di aggiungere dei flag booleani e applicare dei semplici statement if/else all'interno di ciascun metodo. Cio` complicherebbe il nostro codice quando abbiamo piu` stati da considerare negli if/else.
- Inoltre, la logica per tutti gli stati sarebbe dissemintata tra tutti i metodi . Usiamo qundi l'approccio state-specific

- File: esercizio_27_5_2019.py

2

Esercizio 2

- Si consideri lo scenario in cui si ha una lista di numeri che devono essere processati in base all'intervallo a cui appartengono. Gli intervalli considerati sono [1-10], [11,20], [21,30]

- Usare lo schema nel file schemaChain.py per

1. Create un oggetto cliente
2. Creare le richieste da processare
3. Inviare le richieste, una alla volta, agli handler come essi appaiono nella sequenza definita nella classe Client

File: schemaChain.py

3

Esercizio 2

- Si consideri lo scenario in cui si ha una lista di numeri che devono essere processati in base all'intervallo a cui appartengono. Gli intervalli considerati sono [1-10], [11,20], [21,30]

- Usare coroutine per implementare la stessa catena (e` possibile utilizzare Client dello schema di prima aggiungendo un'istruzione per chiudere ...).

4

Esercizio 4

- Scrivere un programma concorrente (facendo prima uso di Futures e Multiprocessing e poi di joinable queue) che prende in input una lista L di stringhe ed un itero n e crea, in modo concorrente, per ciascuna delle stringhe s in L una lista. La lista creata per la stringa i-esima di L deve contenere $n//(10^i)$ occorrenze della stringa. Le liste devono essere stampate non appena vengono create.
- Ad esempio, se n=10 e L=["anna", "mario"]
- vengono stampate le liste ["anna", anna", "anna", anna", "anna", "anna", "anna", "anna", anna", "anna", anna"] ["mario"]

5

Esercizio

- Scrivere una classe tale che ciascuna istanza della classe ha solo tre attributi: nome , cognome ed eta . Non deve essere possibile aggiungere altri attributi.
- Scrivere una classe tale che ogni istanza ha lo stesso stato

6

Esercizio 5

- Scrivere una classe FW_factory che produce istanze di oggetti FW che condividono parte dello stato con altri oggetti
- FW_factory ha un metodo che get_fw prende in input uno stato condiviso sotto forma di lista e restituisce un oggetto FW per quello stato, se gia` esiste o ne crea uno nuovo,
- ed un metodo list_FWs che stampa prima il numero di oggetti FW creati fino a quel momento e poi stampa gli stati relativi agli oggetti FW creati.
- La classe FW ha un costruttore che prende in input lo stato condiviso e lo assegna ad una variabile di istanza shared_state dell'oggetto FW creato
- Inoltre FW ha un metodo op(self, itsOwnState: list,tipo: type, file) che crea un'istanza di tipo con stato formato dalla concatenazione delle liste shared_state e itsOwnState. Il metodo aggiunge al file una linea che contiene lo stato dell'oggetto creato e stampa il suddetto stato restituendolo poi in output.
- file: FW.py

7

Esercizio 6

- Scrivere una classe LaureaT_Student che puo` essere osservata e che ha i seguenti attributi che ne determinano lo stato:
- total_cfu : numero cfu acquisiti
- english_r: booleano settato a False (valore di default) se e solo se lo studente non ha superato la prova di inglese
- grades: dizionario degli esami sostenuti con elementi con chiave uguale al nome dell'esame e valore uguale al voto (exam name, grade)
- exam e` una tupla del tipo definito in basso
  - Exam=collections.namedtuple("Exam", "name cfu")
- Gli attributi total_cfu e english_r sono accessibili con il loro nome e modificabili con '=' mentre grades e` modificabile con il metodo add_grades che prende in input come primo argomento un oggetto Exam e come secondo argomento un int che rappresenta il voto

8

Esercizio 6

Scrivere inoltre i due observer HystoryView e LiveView:

- HistoryView mantiene una lista di triple della forma (dizionario degli esami sostenuti, booleano che indica se inglese superato, data cambio stato) . Ciascuna tripla e` creata quando l'oggetto LaureaT_Student cambia stato.

- LiveView esegue le seguenti stampe:

```
print("Cambio stato: lo studente ha appena superato la prova di Inglese\n")
```

se il cambio di stato e` dovuto al superamento della prova di inglese

```
print("Cambio stato: lo studente ha superato un nuovo esame")
print("Cambio stato: il numero di CFU e` : " , student.total_cfu,"\n")
```

se il cambio di stato e` dovuto al superamento di un nuovo esame

file: observedstudents.py

9