Fondamenti di Informatica (Elettronici)

Da THINK JULIA – Capitolo 4 (Parte b)

23 ottobre 2020

Caso di studio: Design dell'interfaccia (parte b)[1]

1. Refactoring
2. Piano di sviluppo
3. Docstring
4. Debug
5. Glossario
6. Esercizi

[1] Tratto da https://benlauwens.github.io/ThinkJulia.jl/latest/book.html, disponibile sotto Licenza 'Creative Commons Attribution-NonCommercial 3.0 Unported'.

Section 1

Refactoring

Refactoring del nostro modulo geometrico

Quando ho scritto "`circle`", sono stato in grado di riutilizzare "`polygon`" perché un poligono a molti lati è una buona approssimazione di un cerchio.

Ma "`arc`" non è così cooperativo; non possiamo usare "`polygon`" o "`circle`" per disegnare un arco.

Un'alternativa è iniziare con una copia di `polygon` e trasformarla in `arc`. Il risultato potrebbe essere simile a questo:

```
function arc(t, r, angle)
    arc_len = 2 * pi * r * angle / 360
    n = trunc(arc_len / 3) + 1
    step_len = arc_len / n
    step_angle = angle / n
    for i in 1:n
        forward(t, step_len)
        turn(t, -step_angle)
    end
end
```

Una funzione più generale: "polyline"

La seconda metà di questa funzione è simile a "polygon", ma non possiamo riutilizzare "polygon" senza cambiare l'interfaccia.

Potremmo generalizzare `polygon` per prendere un `angle` come terzo argomento, ma poi `polygon` non sarebbe più un nome appropriato! Chiamiamo invece la funzione più generale "polyline":

```
function polyline(t, n, len, angle)
    for i in 1:n
        forward(t, len)
        turn(t, -angle)
    end
end
```

Riscrivi polygon e arc per usare polyline

Ora possiamo riscrivere polygon e arc per usare polyline:

```
function polygon(t, n, len)
    angle = 360 / n
    polyline(t, n, len, angle)
end
function arc(t, r, angle)
    arc_len = 2 * pi * r * angle / 360
    n = trunc(arc_len / 3) + 1
    step_len = arc_len / n
    step_angle = angle / n
    polyline(t, n, step_len, step_angle)
end
```

Riscrivi `circle` per usare `arc`

Infine, possiamo riscrivere `circle` per usare `arc`:

```
function circle(t, r)
    arc(t, r, 360)
end
```

Alcune conclusioni

Questo processo, ovvero la riorganizzazione di un programma per migliorare le interfacce e facilitare il riutilizzo del codice, è chiamato refactoring.

In questo caso, abbiamo notato che c'era un codice simile in `arc` e `polygon`, quindi lo abbiamo "scomposto" in `polyline`.

Se avessimo pianificato in anticipo, avremmo potuto scrivere prima la polilinea ed evitare il refactoring, ma spesso non ne sai abbastanza all'inizio di un progetto per progettare tutte le interfacce.

Una volta che inizi a programmare, comprendi meglio il problema.

A volte il refactoring è un segno che hai imparato qualcosa.

Section 2

Piano di sviluppo

Incapsulamento e generalizzazione

Un piano di sviluppo è un processo per scrivere programmi. Il processo che abbiamo utilizzato in questo caso di studio è "incapsulamento e generalizzazione".

I passaggi di questo processo sono:

1. Inizia scrivendo un piccolo programma senza definizioni di funzione.

Incapsulamento e generalizzazione

Un piano di sviluppo è un processo per scrivere programmi. Il processo che abbiamo utilizzato in questo caso di studio è "incapsulamento e generalizzazione".

I passaggi di questo processo sono:

1. Inizia scrivendo un piccolo programma senza definizioni di funzione.
2. Una volta che il programma funziona, identificare un pezzo coerente di esso, incapsulare il pezzo in una funzione e dargli un nome.

Incapsulamento e generalizzazione

Un piano di sviluppo è un processo per scrivere programmi. Il processo che abbiamo utilizzato in questo caso di studio è "incapsulamento e generalizzazione".

I passaggi di questo processo sono:

1. Inizia scrivendo un piccolo programma senza definizioni di funzione.
2. Una volta che il programma funziona, identificare un pezzo coerente di esso, incapsulare il pezzo in una funzione e dargli un nome.
3. Generalizzare la funzione aggiungendo parametri appropriati.

Incapsulamento e generalizzazione

Un piano di sviluppo è un processo per scrivere programmi. Il processo che abbiamo utilizzato in questo caso di studio è "incapsulamento e generalizzazione".

I passaggi di questo processo sono:

1. Inizia scrivendo un piccolo programma senza definizioni di funzione.
2. Una volta che il programma funziona, identificare un pezzo coerente di esso, incapsulare il pezzo in una funzione e dargli un nome.
3. Generalizzare la funzione aggiungendo parametri appropriati.
4. Ripetere i passaggi 1–3 finché non si dispone di una serie di funzioni operative. Copia e incolla il codice funzionante per evitare di ridigitare (e rieseguire il debug).

Incapsulamento e generalizzazione

Un piano di sviluppo è un processo per scrivere programmi. Il processo che abbiamo utilizzato in questo caso di studio è "incapsulamento e generalizzazione".

I passaggi di questo processo sono:

1. Inizia scrivendo un piccolo programma senza definizioni di funzione.
2. Una volta che il programma funziona, identificare un pezzo coerente di esso, incapsulare il pezzo in una funzione e dargli un nome.
3. Generalizzare la funzione aggiungendo parametri appropriati.
4. Ripetere i passaggi 1–3 finché non si dispone di una serie di funzioni operative. Copia e incolla il codice funzionante per evitare di ridigitare (e rieseguire il debug).
5. Cercare opportunità per migliorare il refactoring del programma. Ad esempio, se hai un codice simile in più punti, considera di scomporlo in una funzione adeguatamente generale.

Metodo di progettazione del software

Progettazione tramite refactoring

Questo processo ha alcuni inconvenienti, vedremo alternative più avanti, ma può essere utile se non sai in anticipo come dividere il programma in funzioni.

Questo approccio ti consente di progettare mentre procedi.

Section 3

Docstring

Esempio di stringa di documentazione (docstring)

Una `docstring` è una stringa prima di una funzione che spiega l'interfaccia ("doc" è breve per "documentation"). Esempio:

```
"""
polyline(t, n, len, angle)
Draws n line segments with the given length and angle
(in degrees) between them. t is a turtle.
"""
function polyline(t, n, len, angle)
    for i in 1:n
        forward(t, len)
        turn(t, -angle)
    end
end
```

Accesso alla documentazione

È possibile accedere alla documentazione in REPL o in un notebook digitando ? seguito dal nome di una funzione o macro, e premendo ENTER:

```
help?> polyline

search:
polyline(t, n, len, angle)
Draws n line segments with the given length and angle
(in degrees) between them. t is a Turtle.
```

Le docstring sono spesso stringhe tra virgolette, note anche come stringhe su più righe perché le virgolette triple consentono alla stringa di occupare più di una riga.

Una `docstring` contiene le informazioni essenziali di cui qualcuno avrebbe bisogno per usare questa funzione.

Spiega in modo conciso cosa fa la funzione (senza entrare nei dettagli di come lo fa).

Spiega quale effetto ha ogni parametro sul comportamento della funzione e che tipo dovrebbe essere ogni parametro (se non è ovvio).

Consigli

SUGGERIMENTI

- Scrivere questo tipo di documentazione è una parte importante del design dell'interfaccia.

Consigli

SUGGERIMENTI

- Scrivere questo tipo di documentazione è una parte importante del design dell'interfaccia.
- Un'interfaccia ben progettata dovrebbe essere semplice da spiegare;

Consigli

SUGGERIMENTI

- Scrivere questo tipo di documentazione è una parte importante del design dell'interfaccia.
- Un'interfaccia ben progettata dovrebbe essere semplice da spiegare;
- se hai difficoltà a spiegare una delle tue funzioni, forse l'interfaccia potrebbe essere migliorata.

Section 4

Debug

Un'interfaccia è un contratto tra una funzione e un chiamante

Il chiamante accetta di fornire* alcuni parametri e la funzione accetta di fare un certo lavoro.

Per esempio:

```
polyline(t, n, len, angle)
```

`polyline` richiede quattro argomenti: `t` deve essere una tartaruga; "`n`" deve essere un numero intero; "`len`" dovrebbe essere un numero positivo; e "angle" deve essere un numero, che si intende in gradi.

Presupposti e postcondizioni

Questi requisiti sono chiamati precondizioni perché dovrebbero essere veri prima che la funzione inizi l'esecuzione.

Al contrario, le condizioni alla fine della funzione sono postcondizioni.

Le postcondizioni includono l'effetto voluto della funzione (come disegnare segmenti di linea) e qualsiasi effetto collaterale (come spostare la tartaruga o apportare altre modifiche).

```
vfill  pause
```

Le condizioni preliminari sono responsabilità del chiamante.

Se il chiamante viola una precondizione (adeguatamente documentata!) e la funzione non funziona correttamente, il bug è nel chiamante, non nella funzione. Se le precondizioni sono soddisfatte e le postcondizioni non lo sono, il bug è nella funzione. Se le tue pre- e postcondizioni sono chiare, possono aiutarti con il debug.

Section 5

Glossario

Glossario I

modulo Un file che contiene una raccolta di funzioni correlate e altre definizioni.

Glossario I

modulo Un file che contiene una raccolta di funzioni correlate e altre definizioni.

pacchetto Una libreria esterna con funzionalità aggiuntive.

Glossario I

modulo Un file che contiene una raccolta di funzioni correlate e altre definizioni.

pacchetto Una libreria esterna con funzionalità aggiuntive.

utilizzando l'istruzione Un'istruzione che legge un file di modulo e crea un oggetto modulo.

Glossario I

modulo Un file che contiene una raccolta di funzioni correlate e altre definizioni.

pacchetto Una libreria esterna con funzionalità aggiuntive.

utilizzando l'istruzione Un'istruzione che legge un file di modulo e crea un oggetto modulo.

ciclo continuo Una parte di un programma che può essere eseguita ripetutamente.

Glossario I

modulo Un file che contiene una raccolta di funzioni correlate e altre definizioni.

pacchetto Una libreria esterna con funzionalità aggiuntive.

utilizzando l'istruzione Un'istruzione che legge un file di modulo e crea un oggetto modulo.

ciclo continuo Una parte di un programma che può essere eseguita ripetutamente.

incapsulamento Il processo di trasformazione di una sequenza di istruzioni in una definizione di funzione.

Glossario I

modulo Un file che contiene una raccolta di funzioni correlate e altre definizioni.

pacchetto Una libreria esterna con funzionalità aggiuntive.

utilizzando l'istruzione Un'istruzione che legge un file di modulo e crea un oggetto modulo.

ciclo continuo Una parte di un programma che può essere eseguita ripetutamente.

incapsulamento Il processo di trasformazione di una sequenza di istruzioni in una definizione di funzione.

generalizzazione Il processo di sostituzione di qualcosa di inutilmente specifico (come un numero) con qualcosa di adeguatamente generale (come una variabile o un parametro).

Glossario I

modulo Un file che contiene una raccolta di funzioni correlate e altre definizioni.

pacchetto Una libreria esterna con funzionalità aggiuntive.

utilizzando l'istruzione Un'istruzione che legge un file di modulo e crea un oggetto modulo.

ciclo continuo Una parte di un programma che può essere eseguita ripetutamente.

incapsulamento Il processo di trasformazione di una sequenza di istruzioni in una definizione di funzione.

generalizzazione Il processo di sostituzione di qualcosa di inutilmente specifico (come un numero) con qualcosa di adeguatamente generale (come una variabile o un parametro).

interfaccia Una descrizione di come utilizzare una funzione, incluso il nome e le descrizioni degli argomenti e del valore restituito.

Glossario II

I

refactoring Il processo di modifica di un programma funzionante per migliorare le interfacce delle funzioni e altre qualità del codice.

Glossario II

I

refactoring Il processo di modifica di un programma funzionante per migliorare le interfacce delle funzioni e altre qualità del codice.

piano di sviluppo Un processo per scrivere programmi.

Glossario II

I

refactoring Il processo di modifica di un programma funzionante per migliorare le interfacce delle funzioni e altre qualità del codice.

piano di sviluppo Un processo per scrivere programmi.

docstring Una stringa che viene visualizzata nella parte superiore della definizione di una funzione per documentare l'interfaccia della funzione.

Glossario II

I

refactoring Il processo di modifica di un programma funzionante per migliorare le interfacce delle funzioni e altre qualità del codice.

piano di sviluppo Un processo per scrivere programmi.

docstring Una stringa che viene visualizzata nella parte superiore della definizione di una funzione per documentare l'interfaccia della funzione.

precondizione Requisito che deve essere soddisfatto dal chiamante prima dell'avvio di una funzione.

Glossario II

I

refactoring Il processo di modifica di un programma funzionante per migliorare le interfacce delle funzioni e altre qualità del codice.

piano di sviluppo Un processo per scrivere programmi.

docstring Una stringa che viene visualizzata nella parte superiore della definizione di una funzione per documentare l'interfaccia della funzione.

precondizione Requisito che deve essere soddisfatto dal chiamante prima dell'avvio di una funzione.

postcondizione Requisito che deve essere soddisfatto dalla funzione prima che termini.

Section 6

Esercizi

aaaa

aaaa