Abstract Data Types: API, ADT

Sommario: 6 aprile, 2016

- Tipi Astratti: Struttura, API e ADT
- Un ADT per IntStack
- ADT: Modifichiamo l'implementazione
- Un API tante ADT
- ADT con polimorfismo: Definiamo Seq(t)
- Moduli

Tipi Astratti: Principi

- **Tipo di Dato Astratto**: Una Collezione di Valori definita dall'insieme delle sole operazioni che possono essere usate per introdurli e operare con essi.
  - **Caratteristiche Distintive** sono le operazioni con cui sono completamente definiti e che si dividono in:
    - **Costruttori**: operazioni per introdurre tali valori (come Valori Esprimibili);
    - **Produttori**: operazioni che calcolano tali valori (come Valori Calcolabili);
    - **Modificatori**: operazioni che modificano (componenti di) tali valori (solo per Tipi Astratti modificabili e strutturati);
    - **Osservatori**: operazioni che accedono i componenti di tali valori (solo per Tipi Astratti strutturati);

Tipi Astratti: Principi/2

- **Tipo di Dato Astratto**: Una Collezione di Valori definita dall'insieme delle sole operazioni che possono essere usate per introdurli e operare con essi.
  - **Caratteristiche Distintive** sono le operazioni con cui sono completamente definiti e che si dividono in:...
  - **Rappresentazione** di tali valori **non è visibile**, e **nemmeno utile** per programmare con tali valori
  - **Implementazione** delle operazioni **non è visibile**[1], e **nemmeno utile** per programmare con tali valori
  - **Rappresentazione e Implementazione** possono essere cambiate senza che il comportamento del programma cambi
  - **Unità di Programmazione per Dati** della Programmazione Strutturata: Introducono nuove collezioni di valori **nascondendo dettagli implementativi che restano però localizzati** nella definizione del tipo astratto.

[1] all'esterno della definizione

Tipi Astratti: API & ADT

- **Tipo di Dato Astratto**: Una Collezione di Valori definita dall'insieme delle sole operazioni che possono essere usate per introdurli e operare con essi.

  - Un tipo astratto è una definizione di tipo con la seguente struttura generale:
    - typeName: che fornisce il nome del nuovo tipo
    - signature: che contiene la segnatura di tutte le operazioni visibili e quindi utilizzabili per il nuovo tipo
    - implementazione: che contiene l'intera implementazione (rappresentazione e implementazione op.)

  - In una sintassi concreta ispirata a Standard ML:

```
abstype typeName{
      private section:  Definizioni di tipo
                per la rappresentazione
   signature
      public section:  Nomi e tipi delle
              operazioni utilizzabili all'esterno
   operations
      private section:  Definizioni delle operazioni,
              incluse le ausiliarie
}
```

  - Applicative Programming Interface = typeName + signature

Scriviamo una API per STACK.

1. STACK di interi? contiene? records?

Possiamo generalizzare —

Definiamo STACK di valori di tipo generico t

⇓

STACK (t)

2. Definiamo la SIGNATURE per STACK (T)

+ EMPTY : () → STACK (T) //COSTRUTTORE

+ PUSH : T × STACK (T) → STACK (T) //PRODUTTORE

+ POP : STACK (T) → STACK (T) //PRODUTTORE

+ TOP : STACK (T) → T //OSSERVATORE

3. Come le API possiamo:

STACK (int) x, // Variabile per STACK di interi

```
Stack(Stack(int)) z; // ...
x = Empty(); // ...
x = Push(3, x); // ...
x = Push(Top(x)*5, x); // ...
y = Push(Empty(), Push(x, Empty())); // ...
```

+ Si commentino gli statement precedenti in particolare ci dica cosa calcolano

+ In ricezione lunedì 11/4/16