



Abstract Data Types

PR2 2018-19

1



Astrazioni

Astrazione procedurale:

- Astrarre dai dettagli di implementazione delle procedure (e dei metodi)
- Specifica come strumento dell'astrazione
- Verifica dell'implementazione rispetto alla specifica

Astrazione sui dati:

- Astrarre dai dettagli della rappresentazione dei dati
- Specifica come strumento di astrazione
- Nozione: Abstract Data Type, ADT

PR2 2018-19

2



Motivazioni per ADT

Definire la struttura e la manipolazione dei dati è una attività pervasiva del progetto di software

- Inventare nuovi algoritmi non capita così tanto spesso

Spesso la prima attività del progetto software consiste nel progettare le strutture dati

- Come sono organizzati i dati
- Quali sono le operazioni sui dati messe a disposizione dei clienti del dato.

Problematiche:

- La scelta delle strutture dati può essere effettuata troppo presto
- Complicato modificare le strutture dati essenziali

PR2 2018-19

3



ADT un insieme di operazioni

- ADT astrae dalla organizzazione e dal significato specifico dei dati
- Le scelte di rappresentazione sono nascoste al cliente dei dati (information hiding)

```
class RightTriangle {  
    float base, altitude;  
}
```

```
class RightTriangle {  
    float base, hypot, angle;  
}
```

Idea: pensare ai dati in termini delle operazioni fornite

create, getBase, getAltitude, getBottomAngle, ...

Forzare i clienti ad usare le operazioni per accedere ai dati

PR2 2018-19

4



Una domanda.

Queste due classi descrivono la stessa astrazione?

```
class Point {          class Point {
  public float x;      public float r;
  public float y;      public float theta;
}                      }
```

Diverse: non possono essere messe una al posto dell'altra all'interno di un programma

Identiche: entrambi le classi definiscono la nozione di "2-d point"

La metodologia dei ADT è quella di esprimere la nozione di uniformità della astrazione dei dati:

- Cliente deve usare solo il concetto di "2-d point"



ADT sono importanti

- Si possono ritardare le scelte specifiche relative all'implementazione del tipo di dato
- Si possono risolvere eventuali errori di programmazione modificando solamente l'implementazione del dato
- Si possono modificare gli algoritmi
 - Maggiore performance
 - Adattare l'algoritmo a specifici contesti

Noi parleremo di "*abstraction barrier*"

- Buona cosa averla e non essere in grado di violarla

ADT: 2-d point

```

class Point {
  // A 2-d point, ...
  public float x();
  public float y();
;
  // ... can be created, ...
  public Point(); // new point at (0,0)
  public Point centroid(Set<Point> points);
  // ... can be moved, ...
  public void translate(float delta_x,
                       float delta_y);
  public void scaleAndRotate(float delta_r,
                             float delta_theta);
}
  
```

} Observers

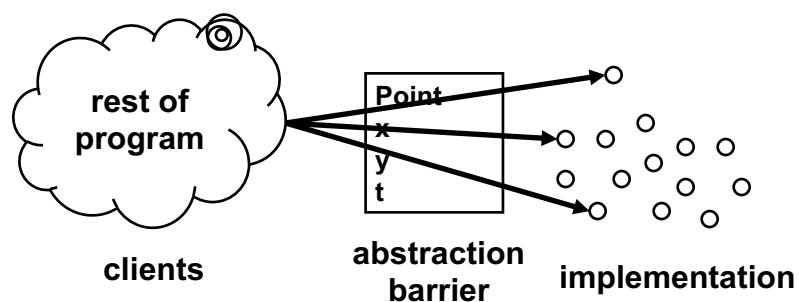
} Creators/
Producers

} Mutators

PR2 2018-19

7

Abstract data type = objects + operations



- Implementazione è nascosta
- L'unico modo per accedere ai dati consiste nell'utilizzare le operazioni dell'astrazione

PR2 2018-19

8



Specificare una data abstraction

- Una collezione di astrazioni procedurali
 - *Non una collezione di metodi*
 - *Specifica tramite @requires, @effects,*
- Uno stato astratto
 - Non è la rappresentazione concreta dei dati, ...
 - Stato astratto permette di specificare le astrazioni procedurali
 - Stato concreto non fa parte della specifica
- Le operazioni descritte in termini di “creating”, “observing”, “producing”, or “mutating”
 - Le uniche operazioni sono quelle definite nelle specifica

PR2 2018-19

9



Specificare ADT

Immutable

1. **overview**
2. **abstract state**
3. **creators**
4. **observers**
5. **producers**
- ~~6. **mutators**~~

Mutable

1. **overview**
2. **abstract state**
3. **creators**
4. **observers**
5. **producers (rari)**
6. **mutators**

- Creators: restituiscono un nuovo oggetto del ADT (Java constructors)
- Producers: Operazioni (dell'ADT) che restituiscono valori
- Mutators: Modificano lo stato concreto ADT
- Observers: Restituiscono informazioni sullo stato ADT

PR2 2018-19

10



Implementazione ADT

Definire le strutture di implementazione concrete e implementare i metodi (tramite una opportuna codifica con gli strumenti linguistici messi a disposizione da Java)

- Lo vedremo in seguito
- *La rappresentazione concreta non deve emergere dalla specifica*

PR2 2018-19

11



Poly, immutable ADT

```
/**  
 * A Poly is an immutable polynomial with  
 * integer coefficients. A typical Poly is  
 *  $C_0 + C_1x + C_2x^2 + \dots$   
 **/
```

```
class Poly {
```



Abstract state

Clausola Overview:

- Definisce se il nuovo ADT è mutable o immutable
- Definisce il modello astratto da utilizzare nella specifica del tipo di dato astratto
 - Difficile! ... ma essenziale
 - Bene usare notazioni matematiche non ambigue

PR2 2018-19

12



Poly: creators

```
// @effects: makes a new Poly = 0
public Poly()

// @effects: makes a new Poly = cx^n
// @throws: NegExponent if n < 0
public Poly(int c, int n)
```

Creators

- Nuovi oggetti: solo **effects**

PR2 2018-19

13



Poly: observers

```
// @returns: the degree of this,
//   the largest exponent with a
//   non-zero coefficient.
//   Returns 0 if this = 0.
public int degree()

// @returns: the coefficient of the term
//   of this whose exponent is d
// @throws: NegExponent if d < 0
public int coeff(int d)
```

PR2 2018-19

14



Observers

Observers

- Utilizzati per reperire informazione
- Non modificano lo stato astratti
- Utilizzano la rappresentazione astratta dell'oggetto

```
Poly x = new Poly(4, 3);  
int c = x.coeff(3);  
System.out.println(c); // prints 4
```



Poly: producers

```
// returns: this + q  
//(visto come un obj di tipo Poly)  
public Poly add(Poly q)  
  
// returns: the Poly equal to this * q  
public Poly mul(Poly q)  
  
// returns: -this  
public Poly negate()
```




Producers

- Operazioni che creano nuovi dati
- Operazioni tipiche in Java: `java.lang.String`
 - `String substring(int offset, int len)`
- Nessun effetto laterale
 - Non modificano lo stato astratto



IntSet: mutable datatype:

```
// Overview: An IntSet is a mutable,  
// unbounded set of integers. A typical  
// IntSet is { x1, ..., xn }.  
class IntSet {  
  
    // effects: makes a new IntSet = {}  
    public IntSet()  
  
}
```



IntSet: observers

```
// returns: true if and only if  $x \in$  this
public boolean contains(int x)

// returns: the cardinality of this
public int size()

// returns: some element of this
// throws: EmptyException when size() == 0
public int choose()
```

PR2 2018-19

19



IntSet: mutators

```
// modifies: this
// effects:  $\text{this}_{\text{post}} = \text{this}_{\text{pre}} \cup \{x\}$ 
public void add(int x)

// modifies: this
// effects:  $\text{this}_{\text{post}} = \text{this}_{\text{pre}} - \{x\}$ 
public void remove(int x)
```

PR2 2018-19

20



Mutators

- Operazioni che modificano lo stato (**this**)
- Raramente modificano lo stato del cliente
- Di solito non producono valori
 - “Do one thing and do it well”