Abstract Data Types

Astrazioni

Astrazione procedurale:

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

Astrazione sui dati:

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

Motivazioni per ADT

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

- Inventare nuovi algoritmi non capita cosí 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

ADT un insieme di operazioni

- ADT astrae dalla organizzazione e dal significato specifico dei dati
- Le scelte di rappresentazione sono nascoste al cliente del 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 leoperazioni per accedere ai dati

Una domanda.

Queste due classi descrivono la stessa astrazione?

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

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

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

Identiche: entrabi 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 sclete specifiche relative all'implementazione del tipo di dato
- Si possono risolvere eventuali errori di programamzione modificando solamente l'implementazione del dato
- Si possono mofificare 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

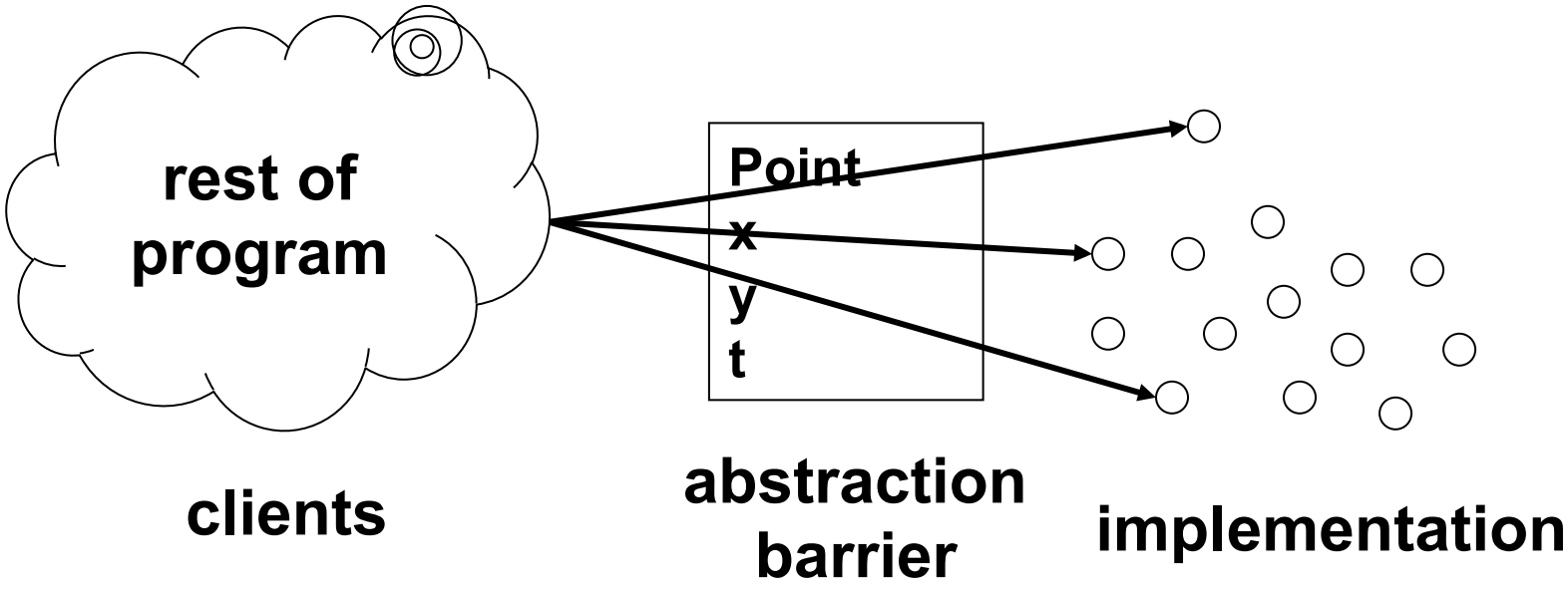
```
class Point {
  // A 2-d point, ...
  public float x();
  public float y();
;
  // ... can be created, ...
  public Point(); // new point at (0,0)
  public Point centroid(Set<Point> points);
  // ... can be moved, ...
  public void translate(float delta_x,
                        float delta_y);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

Observers

Creators/ Producers

Mutators

Abstract data type = objects + operations



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

Specificare una data abstraction

- Una collezzione di astrazioni procedurali
  - *Non una collezione di metodi*
  - *Specifica tramire @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 operazione descritte in termini di “creating”, “observing”, “producing”, or “mutating”
  - Le uniche operazoni sono quelle definite nelle specifica

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 (rare)**
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: Restituioscono infotmazioni sullo stato ADT

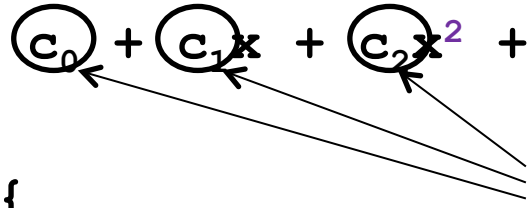
Impementazione ADT

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

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

Poly, immutable ADT

```
/**
 * A Poly is an immutable polynomial with
 * integer coefficients.  A typical Poly is
 *              c0 + c1x + c2x^2 + ...
 **/
class Poly {
```



Clausola Overview:

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

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`**

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)
```

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 ∈ 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()
```

IntSet: mutators


```
// modifies: this
// effects:  this_post = this_pre ∪ {x}
public void add(int x)

// modifies: this
// effects:  this_post = this_pre - {x}
public void remove(int x)
```

Mutators

- Operazioni che moficano l'o stato (**this**)
- Raramente modificano lo stato del cliente
- Di solito non producono valori
  - "Do one thing and do it well"