



# Programmazione II

---

## 8. Esempi di Tipi: IntSet e Poly (PDJ 5.1-5.7)

# Esempi di Tipi di dati

## Esempi PDJ IntSet e Poly

I prossimi esempi sono da PDJ. Sono utili perchè danno un'idea di progettazione di tipi ben noti

■ **IntSet** è un tipo mutabile che modella un insieme di interi

- Astrazione: insieme di numeri interi, non ripetuti, senza distinzione di ordine
- Comportamenti: metodi di osservazione e mutazionali necessari per lavorare su insiemi di interi  
aggiunta, rimozione, restituzione, presenza, cardinalità

■ **Poly** è un tipo immutabile che modella i polinomi

- Astrazione: espressioni costituite da somme di monomi (su incognita singola  $x$ )
- (un monomio è un prodotto tra un intero  $c$  e un'incognita  $x$  elevata a una potenza  $e$  intera positiva)
- es:  $5x^4 - 2x^2 + 7x + 2$
- Comportamenti: metodi di produzione e osservazione per costruire e operare su polinomi  
aggiunta, sottrazione, moltiplicazione, cardinalità, coefficiente per un dato grado

# IntSet

## Specifica IntSet

```
1 public class IntSet {
2 //OVERVIEW: IntSets are mutable, unbounded sets of integers. e.g., {x_1, . . . , x_n}.
3
4 //constructors
5 public IntSet()
6 //MODIFIES: this
7 //EFFECTS: Initializes this (empty)
8
9 //methods
10 public void insert(int x)
11 //MODIFIES: this
12 //EFFECTS: Adds x to this, i.e., this_post = this + x
13
14 public void remove(int x)
15 //MODIFIES: this
16 //EFFECTS: removes x from this, i.e., this_post = this - { x }
17
18 public boolean contains(int x)
19 //EFFECTS: If x in this return true, else return false
20
21 public int size()
22 //EFFECTS: return size of this
23
24 public int choose() throws NoSuchElementException
25 //EFFECTS: If this is empty throws NoSuchElementException, else returns element of this
26 }
```

## Rappresentazione IntSet

- Deve essere sufficientemente espressiva ma il meno ridondante possibile
  - array di int o List di Integer (facciamo array per esercitarci, esempio List dopo)
  - il libro di testo usa un Vector (variante obsoleta di List)

```
1 private int[] els; //assumiamo rappresentazione come array
```

## Costruttore

- Da specifica inizializza insieme vuoto
  - inizializza la List, non servono parametri

```
1 public IntSet() {  
2 //MODIFIES: this  
3 //EFFECTS: Initializes this (empty)  
4     this.els = {};  
5 }
```

- Oppure inizializzo list nella rep e ometto il costruttore (esplicito)

```
1 private int[] els = {}; //si puo' assegnare subito -> non serve il costruttore (esplicito)
```

## Implementazione insert()

- caso speciale 1: se primo elemento lo aggiungo col literal
- caso speciale 2: se elemento presente non faccio nulla
- caso principale: creo array nuovo più grande, copio il vecchio e aggiungo x in coda

```
1 public void insert(int x) {
2     //MODIFIES: this
3     //EFFECTS: Adds x to this, i.e., this_post = this + x
4     if(els.length == 0) {
5         els = new int[]{x}; //se non ci sono elementi inizializzo col primo
6     else if(!this.contains(x)) { //se non c'e'
7         int[] tmp = new int[els.length+1]; //nuovo array esteso di 1
8
9         for(int i=0; i < els.length; i++) //copio tutto
10             tmp[i] = els[i];
11
12         tmp[els.length] = x; //x come ultimo valore
13         this.els = tmp; //rimetto in els
14     }
15
16     assert repOk();
17 }
```

## Implementazione remove()

- caso speciale: se array vuoto o elemento non presente non faccio nulla
- caso principale: creo array nuovo più piccolo, copio il vecchio ma salto x

```
1 public void remove(int x) {
2     //MODIFIES: this
3     //EFFECTS: removes x from this, i.e., this_post = this - {x }
4     if(els.length == 0 || !contains(x)) //se non elementi o non ha x tra gli elementi
5         return; //non deve fare modifiche
6
7     int[] tmp = new int[els.length-1]; //inizializza un nuovo array piu' piccolo di 1
8
9     for(int i = 0, j = 0; i < tmp.length; i++, j++) { //i su tmp e j su els
10         if(els[i] == x) //se trovo l'elemento da togliere lo salto
11             j++;
12
13         tmp[i] = els[j]; //copio gli altri elementi
14     }
15
16     els = tmp; // riassegno els a tmp
17     assert repOk();
18 }
```

## Implementazione contains()

```
1 public boolean contains(int x) {
2 //EFFECTS: If x in this return true, else return false
3     for(int i : els)
4         if(i == x)
5             return true;
6
7     return false;
8 }
```

## Implementazione size()

```
1 public int size() {
2 //EFFECTS: return size of this.
3     return els.length;
4 }
```

## Implementazione choose()

```
1 public int choose() throws NoSuchElementException {
2 //EFFECTS: If this is empty throws NoSuchElementException, else returns element of this
3     if(els.length == 0)
4         throw new NoSuchElementException("no elements in set");
5
6     return els[0];
7 }
```

## Implementazione repOk()

```
1 public boolean repOk() { //valida sempre tranne quando ci sono duplicati
2     for(int i=0; i < els.length-1; i++)
3         for(int j=i+1; j < els.length; j++)
4             if(els[i] == els[j]) //se per un dato elemento ne esiste uno successivo uguale
5                 return false; //la rep e' errata
6     return true; //altrimenti tutto OK
7 }
```

## Implementazione toString()

```
1 @Override
2 public String toString() { //non essendoci duplicati tutta rep modella l'astrazione
3     String ret = "IntSet: [ "; //segnalare che e' un'istanza di IntSet
4     for(int i : els)
5         ret += i + " "; //riportare gli elementi
6     return ret + "]";
7 }
```

## Implementazione clone()

```
1 @Override
2 public IntSet clone() { //non ci possono essere eccezioni...
3     IntSet t = new IntSet(); //...perche' non serve super.clone()...
4     t.els = this.els.clone(); //...perche' l'unico attributo lo devo clonare a mano
5     return t;
6 }
```

## Implementazione equals()

```
1  @Override
2  public boolean equals(Object o) {
3      if(this == o)
4          return true;
5
6      if(o == null)
7          return false;
8
9      if(!(this.getClass().equals(o.getClass())))
10         return false;
11
12     IntSet t = (IntSet) o;
13
14     if(els.length != t.els.length) //se lunghezza diversa non possono essere uguali
15         return false;
16
17     for(int i=0; i < els.length; i++) { //ogni elemento di this...
18         if(!t.contains(els[i])) //...deve essere in t...
19             return false; //...altrimenti sono diversi
20
21     }
22 }
```

# IntSet

## Implementazione IntSet con rep List

```
1 public class IntSet implements Cloneable {
2     private List<Integer> els = new ArrayList<>();
3
4     public void insert(int x) {
5         if(!this.contains(x))
6             els.add(x);
7
8         assert repOk();
9     }
10
11    public void remove(int x) {
12        els.remove((Integer)x); //serve perche' esiste un metodo remove(int i) dove i e' indice
13
14        assert repOk();
15    }
16
17    public boolean contains(int x) {
18        return els.contains(x);
19    }
20
21    public int size() {
22        return els.size();
23    }
24
25    public int choose() throws NoSuchElementException {
26        if(els.size() == 0)
27            throw new NoSuchElementException("no elements in set");
28
29        return els.get(0);
30    }
31 }
```

## Specifica Poly

```
1 public class Poly {
2 //OVERVIEW; Polys are immutable polynomials with int coefficients. e.g., C_0 + C_1x + ...
3
4 public Poly(int c, int n) throws NegativeExponentException
5 //EFFECTS: initializes this to be  $cx^n$ . omit if  $c == 0$ 
6 //          If  $n < 0$  throws NegativeExponentException
7
8 public int deg()
9 //EFFECTS: Returns largest exp with non-zero coeff, or 0 if this is the zero Poly
10
11 public int coeff(int d)
12 //EFFECTS: Returns the coefficient of the term of this whose exponent is d
13
14 public Poly add(Poly q) throws NullPointerException
15 //EFFECTS: returns the Poly this + q
16 //          if q is null throws NullPointerException
17
18 public Poly mul(Poly q) throws NullPointerException
19 //EFFECTS: returns the Poly this * q
20 //          if q is null throws NullPointerException
21
22 public Poly sub(Poly q) throws NullPointerException
23 //EFFECTS: returns the Poly this - q
24 //          if q is null throws NullPointerException
25 }
```

## Rappresentazione Poly

### ■ Diverse possibilità:

- (denso) array o List di int dove indice = grado e valore = costante (esempio nel libro di testo)
- (sparso) array di gradi e array di costanti allineati per indice
- (sparso) array di Record con grado e costante
- (sparso) HashMap con chiave = grado e valore = costante (facciamo questo, esempio denso dopo)

```
1 private final HashMap<Integer,Integer> m = new HashMap<>();
```

## Costruttore

```
1 public Poly(int c, int n) throws ArithmeticException {
2 //EFFECTS: initializes this to be  $cx^n$ . omit if  $c == 0$ 
3 //          If  $n < 0$  throws NegativeExponentException
4     if(n < 0)
5         throw new ArithmeticException("n < 0");
6
7     if(c != 0)
8         this.m.put(n,c);
9
10    assert repOk();
11 }
```

## Implementazione coeff()

```

1 public int coeff(int d) {
2 //EFFECTS: Returns the coefficient of the term of this whose exponent is d
3     return this.m.getDefault(d, 0);
4 }

```

## Implementazione add()

■ devo copiare i termini in this e in q, sommando quelli presenti in entrambi

• es:  $A = 3x^2 + 2x$ ;  $B = 4x^3 - x$ ;  $A + B = 4x^3 + 3x^2 + x$

```

1 public Poly add(Poly q) throws NullPointerException {
2 //EFFECTS: returns the Poly this + q
3 //     if q is null throws NullPointerException
4     if(q == null)
5         throw new NullPointerException("q null");
6
7     Poly ret = new Poly(0,0);
8
9     for(int i = 0; i <= Math.max(this.deg(),q.deg()); i++) //per tutti i gradi in q o this
10         if(this.coeff(i) + q.coeff(i) != 0) //se somma coefficienti != 0
11             ret.m.put(i, this.coeff(i) + q.coeff(i)); //aggiungi a ret
12
13     assert ret.repOk(); //anche se astrazione immutabile, la rep non lo e', devo controllare
14     return ret;
15 }

```

## Implementazione mul()

- devo moltiplicare ciascun termine del primo polinomio con quelli del secondo

• es:  $A = 2x^2 + 3x$ ;  $B = 3x^2 - x$ ;  $A * B = 6x^4 + 7x^3 - 3x^2$

```

1 public Poly mul(Poly q) throws NullPointerException {
2 //EFFECTS: returns the Poly this * q
3 //      if q is null throws NullPointerException
4     if(q == null)
5         throw new NullPointerException("q null");
6
7     Poly ret = new Poly(0,0);
8
9     for(int i = 0; i <= this.deg(); i++) //moltiplico tutti i termini di this
10        for(int j = 0; j <= q.deg(); j++) //per tutti i termini di q
11            ret = ret.add(new Poly(this.coeff(i) * q.coeff(j),i+j)); //aggiungo a ret
12
13     return ret;
14 }

```

## Implementazione sub()

```

1 public Poly sub(Poly q) throws NullPointerException {
2 //EFFECTS: returns the Poly this - q
3 //      if q is null throws NullPointerException
4     if(q == null)
5         throw new NullPointerException("q null");
6
7     return this.add(q.mul(new Poly(-1,0)));
8 }

```

## Implementazione deg()

```
1 public int deg() {  
2 //EFFECTS: Returns largest exp with non-zero coeff, or 0 if this is the zero Poly  
3     int max = 0;  
4  
5     for(int k: this.m.keySet())  
6         if(k > max)  
7             max = k;  
8  
9     return max;  
10 }
```

■ Per non rifare questo calcolo ogni volta posso fare il caching del grado

- aggiungo alla rep una variabile 'private int deg = 0'
- nel costruttore imposto 'deg = n;' dove  $n$  è l'esponente del monomio
- in add() imposto deg al grado massimo tra i monomi non nulli risultanti
- deg() diventa solamente 'return deg;'
- esempio di effetto collaterale benevolo

## Implementazione equals()

```
1  @Override
2  public boolean equals(Object o) {
3      if(this == o)
4          return true;
5
6      if(o == null)
7          return false;
8
9      if(!(this.getClass().equals(o.getClass())))
10         return false;
11
12     Poly t = (Poly) o;
13
14     if(this.deg() != t.deg())
15         return false;
16
17     for(int i = 0; i <= this.deg(); i++)
18         if(this.coeff(i) != t.coeff(i))
19             return false;
20
21     return true;
22 }
```

## Implementazione hashCode()

```
1  @Override
2  public int hashCode() {
3      return Objects.hash(m);
4  }
```

## Implementazione clone()

```
1  @Override
2  public Poly clone() {
3      return this.add(new Poly(0,0));
4  }
```

## Implementazione repOk()

```
1 public boolean repOk() {
2     int tdeg = 0;
3
4     for(Integer k: this.m.keySet())
5         if(k == null || k < 0 || this.m.get(k) == null)
6             return false;
7         else if(this.m.get(k) > tdeg)
8             tdeg = this.m.get(k);
9
10    if(tdeg != deg()) //uso deg(), va bene con o senza caching
11        return false;
12
13    return true;
14 }
```

## Implementazione toString()

```
1 @Override
2 public String toString() {
3     String ret = "";
4
5     for(int i = this.deg(); i >= 0; i--) {
6         if(this.coeff(i) != 0) {
7             if(!ret.equals(""))
8                 ret += " + ";
9
10            ret += this.coeff(i) + "x^" + i;
11        }
12    }
13
14    return "Poly (deg " + deg() + "): " + ret;
15 }
```

## Implementazione Poly (dense) con rep List

```
1 public class Poly implements Cloneable {
2     private List<Integer> m = new ArrayList<>();
3
4     public Poly(int c, int n) throws ArithmeticException {
5         if(n < 0)
6             throw new ArithmeticException("n < 0");
7
8         if(c != 0)
9             for(int i = 0; i <= n; i++)
10                 i == n ? this.m.add(c) : this.m.add(0);
11
12         assert repOk();
13     }
14
15     public int deg() {
16         return this.m.size()-1;
17     }
18
19     public int coeff(int d) {
20         return d > this.deg() ? 0 : this.m.get(d);
21     }
22
23     public Poly add(Poly q) throws NullPointerException {
24         if(q == null)
25             throw new NullPointerException("q null");
26
27         Poly ret = new Poly(0,0);
28         for(int i = 0; i <= Math.max(this.deg(),q.deg()); i++)
29             ret.m.add(this.coeff(i) + q.coeff(i));
30
31         assert ret.repOk();
32         return ret;
33     }
34 }
```

# Programmazione II

---

## 8. Esempi di Tipi: IntSet e Poly (PDJ 5.1-5.7)

Dragan Ahmetovic