Java: Additional Features

Sommario: 3-4 Maggio, 2016

- ADT: Ancora una condizione
- Collection: Vector e LinkedList
- Uguaglianza di valori: ==, equals
- Duplicazione di valori: clone
- Presentazione di valori: toString
- ADT per valori strutturati: elements.
- Enhanced for (o for each)

ADT: Ancora una condizione

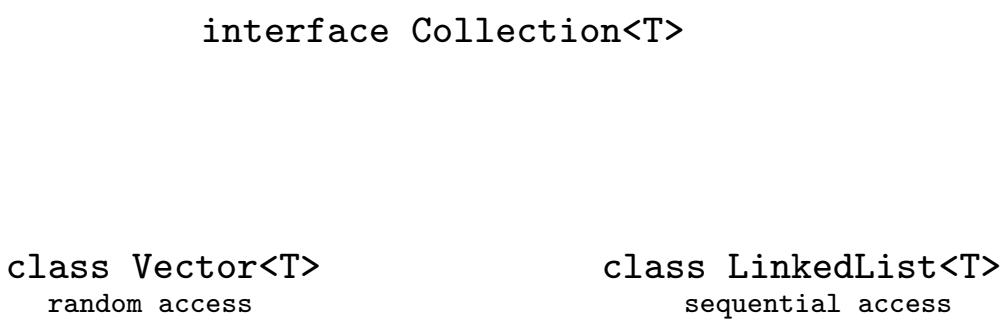
- ADT emulati in Java mediante classi e modificatori
- 3 condizioni:
  - **Stato Privato**
    Implementazione dei valori Inaccessibile
  - **Segnatura Pubblica**
    Uniche operazioni usabili dall'esterno della classe
  - **Esposizione Stato**
    Parametri trasmessi e Valori Calcolati delle operazioni pubbliche non devono mostrare parti dello stato.
    Esempio. Classe muPairADTCheating.java

Definition (Condizione di Non Esposizione dello Stato)

La stato della rappresentazione concreta non deve essere esposto in nessuna parte nè attraverso parametri nè attraverso il valore calcolato di un metodo pubblico. Quando la condizione è soddisfatta, l'ipotesi induttiva I(c) può essere assunta su c prima dell'invocazione di un metodo se provata vera sui costruttori di c.

Dati Strutturati

- **Collection** Interfaccia per gestire gruppi di oggetti
- 2 sottoclassi: Vector<T> e LinkedList<T>



Vector e LinkedList

- Vector<T>: Le operazioni che useremo
- LinkedList<T>: Le operazioni che useremo
- Vector<T>. Usiamo in:
  - Definizioni di tipi concreti: Poly.java
  - Definizioni di ADT: PolyADT.java

Equivalenza di tipi e Assegnamento

- Java è un Linguaggio "fortemente tipato":

Definition (Strongly Typed, ST)

Ad ogni costruzione (espressione o comando) c, di ogni programma (legale) possiamo associare (a compile time) un **tipo unico** (possibilmente, un supertipo del tipo effettivo): $(\exists!\ \mathtt{T})\mathtt{c} : \mathtt{T}$

- I tipi di un Linguaggio ST hanno relazioni di equivalenza:
  - Strutturale e/o;
  - Nominale
- I tipi di ~~un~~ Java hanno relazioni di equivalenza: ...

  `T1 equivalente T2 sse T1 è T2`
- Assegnamento:

  $(\mathtt{x} = \mathtt{e}) : \mathtt{T1}$ `sse` $(\mathtt{x}{:}\mathtt{T1} \wedge \mathtt{e}{:}\mathtt{T2} \wedge \mathtt{T1} > \mathtt{T2})$

Equivalenza di valori: ==, equals

- Categorie di Valori/Oggetti in Java:
  2 in accordo al **comportamento atteso**
  - **Modificabili (Mutable)**
    - `Stato oggetto può cambiare`
  - **NonModificabili (Immutable)**
    - `Stato oggetto non può cambia`
- Per l'equivalenza di valori Java possiede:
  - operatore ==
    - `v1==v2 sse stesso` **reference** `in memoria`
    - Corretto solo per Mutable e valori scalari (int, char, ...)
  - metodo **equals**:
    - Definito in Object ed ereditato da tutte le classi
      `public boolean equals(Object o)`
    - Ereditato: corretto solo per Mutable
    - Overidden: **obbligatoriamente** da tipi Immutable

ImPairADT con equals

```
public class ImPairADTX <A,B> implements Cloneable{
    private final A left;//da nascondere
    private final B right;//da nascondere
    public ImPairADTX (A x, B y) {
        left = x;
        right = y;
    }
    public A getLeft(){...}
    public B getRight(){...}
    public boolean equals(Object o){//override equals
        ImPairADTX<?,?> ok;
        try{ok = (ImPairADTX<?,?>)o;}
        catch(Exception e){return false;}
        return (left.equals(ok.left) && right.equals(ok.right));
    }
```

Duplicazione di Oggetti: clone

- metodo **clone**:
  - Definito in Object per tutte le classi (Cloneable)

```
protected Object clone()
                  throws CloneNotSupportedException
```

  - Crea una differente copia dell'oggetto: `x.clone()!=x`
  - Overidden: **obbligatoriamente** da tutti i tipi Cloneable

ImPairADT con equals e clone

```
public class ImPairADTX <A,B> implements Cloneable{
    private final A left;//da nascondere
    private final B right;//da nascondere
    public ImPairADTX (A x, B y) {
        left = x;
        right = y;
    }
    public A getLeft(){...}
    public B getRight(){...}
    public boolean equals(Object o){...}
    protected ImPairADTX<A,B> clone() throws CloneNotSupportedException{
        return (new ImPairADTX<A,B>(left,right));
    }
    public String toString(){...}
}
```

Presentazione dei valori: toString

- metodo **toString**:
  - Definito in Object per tutte le classi
    ```
    public String toString()
    ```
  - Crea una stringa che rappresenta l'oggetto in modo testuale
  - Overidden: per fornire una presentazione dei valori

ImPairADT con equals, clone, toString

```
public class ImPairADTX <A,B> implements Cloneable{
    private final A left;//da nascondere
    private final B right;//da nascondere
    public ImPairADTX (A x, B y) {
        left = x;
        right = y;
    }
    public A getLeft(){...}
    public B getRight(){...}
    public boolean equals(Object o){...}
    protected ImPairADTX<A,B> clone() throws CloneNotSupportedException{...}
    public String toString(){
        return ("("+left.toString()+","+right.toString()+")");
    }
}
```

ImPairADTX con equals, clone, toString e Caso di uso

```
public class ImPairADTX <A,B> implements Cloneable{
    private final A left;//da nascondere
    private final B right;//da nascondere
    public ImPairADTX (A x, B y) {
        left = x;
        right = y;
    }
    public A getLeft(){...}
    public B getRight(){...}
    public boolean equals(Object o){//override equals
        ImPairADTX<?,?> ok;
        try{ok = (ImPairADTX<?,?>)o;}
        catch(Exception e){return false;}
        return (left.equals(ok.left) && right.equals(ok.right));
    }
    protected ImPairADTX<A,B> clone() throws CloneNotSupportedException{
        return (new ImPairADTX<A,B>(left,right));
    }
    public String toString(){
        return ("("+left.toString()+","+right.toString()+")");
    }
}

/.../

/* (c) */
class main{
    public static void main(String args[])throws CloneNotSupportedException{
        ImPairADTX <Integer,String> myPlayCard = new ImPairADTX<Integer,String>(3,"fiori");
        System.out.println("il valore della carta è " + myPlayCard.getLeft());
        ImPairADTX <Integer,String> myPlayCard2 = new ImPairADTX<Integer,String>(3,"fiori");
        System.out.println("una copia della stessa carta? " + myPlayCard.equals(myPlayCard2));
        ImPairADTX <Integer,String> myPlayCard3 = myPlayCard.clone();
        System.out.println("una copia della stessa carta? " + myPlayCard.equals(myPlayCard3));
        System.out.println("una presentazione della carta è " + myPlayCard.toString());
    }
}
```