Java 1.5

Le liste e gli insiemi sono collezioni di variabili `Object`

Si possono definire collezioni di tipi particolari

Esempio: lista di `Point`, insieme di `Studente`, ecc.

Esempio

Lista di studenti:

```
import java.util.*;

class ListStud {
  public static void main(String args[]) {
    LinkedList<Studente> l;
    l=new LinkedList<Studente>();

    l.add(new Studente("Ciccio", 932222));
    l.add(new Studente("Franco", 234325));

    // errore
    // l.add(new Point(20,30));

    Studente s=l.get(1);
  }
}
```

Osservazioni

Si può specificare che una lista contiene solo elementi di un certo tipo:

```
LinkedList<Studente> l;
l=new LinkedList<Studente>();
```

Una volta fatto questo, è come se la lista fosse una sequenza di variabili `Studente`, invece che una lista di variabili `Object`:

1. si possono solo inserire oggetti `Studente` (o sottoclassi): `l.add(new Point(20,30));` è un errore;
2. il valore di ritorno di `get` è di tipo `Studente`

Cosa cambia

1. si mette `<Tipo>` nella dichiarazione di variabile e nel costruttore;
2. al posto fdi `Object`, si usa sempre `Tipo` in tutti i metodi (`get`, `add`, ecc.)

Sottoclassi

Tutto come al solito: dove si può mettere `Studente` si può mettere `Borsista`:

```
import java.util.*;

class ListStud {
  public static void main(String args[]) {
    LinkedList<Studente> l=new LinkedList<Studente>();

    l.add(new Studente("Ciccio", 932222));
    l.add(new Studente("Franco", 234325));

    l.add(new Borsista("Io", 23433, 2300));

    Studente s=l.get(1);

    // errore
    // Borsista b=l.get(2);

    System.out.println(l);
  }
}
```

Metodi

Si possono anche realizzare metodi con liste parametrizzate:

```
import java.util.*;

class Metodo {
  static int somma(LinkedList<Integer> l) {
    int a=0;

    for(int i=0; i<l.size(); i++) {
      Integer x=l.get(i);
      a=a+x.intValue();
    }

    return a;
  }

  public static void main(String args[]) {
    LinkedList<Integer> l;
    l=new LinkedList<Integer>();

    l.add(new Integer(12));
    l.add(new Integer(1));
    l.add(new Integer(3));

    System.out.println(somma(l));
  }
}
```

Autobox

È una facilitazione introdotta per semplificare programmi che usano i tipi `Integer` ecc.

L'idea è usare gli oggetti come fossero numeri e viceversa:

```
  Integer i=12;

  int x=i;
```

Esempio

```
import java.util.*;

class Metodo {
  static int somma(LinkedList<Integer> l) {
    int a=0;

    for(int i=0; i<l.size(); i++) {
      a=a+l.get(i);
    }

    return a;
  }

  public static void main(String args[]) {
    LinkedList<Integer> l;
    l=new LinkedList<Integer>();

    l.add(12);
    l.add(1);
    l.add(3);

    System.out.println(somma(l));
  }
}
```

Attenzione!

Il compilatore automaticamente converte:

```
Integer i=12;    ---> Integer i=new Integer(12);

int x=i;         ---> int x=i.intValue();
```

In ogni caso, `i` resta una variabile oggetto e `x` resta una variabile intera.