Il linguaggio Java

Le classi astratte e le classi finali

Problema

- La politica della banca richiede che ciascun tipo di conto possa prevedere una qualche forma di commissione mensile
- Perciò ciascuna classe deve avere il metodo deductFees

```
public class BankAccount {
    public void deductFees() {} // non fa nulla!!!
    // altri metodi
}
```

Cosa accade se il programmatore si dimentica di ridefinire il metodo **deductFees()** nelle classi derivate?

Metodo Astratto

public **abstract** void deductFees();

- Un metodo astratto non ha implementazione
- Non è possibile creare oggetti istanza di una classe che contiene un metodo astratto perciò viene detta *astratta* (altrimenti è detta *concreta*)
- Una classe astratta può essere estesa
- Un metodo astratto può essere sovrascritto altrimenti viene ereditato

Classe Astratta (I)

Una classe che definisce un metodo astratto o che eredita un metodo astratto senza sovrascriverlo deve essere dichiarata **abstract**

```
abstract class A {
    public abstract int f(double d);
    // …
}

abstract class B extends A {
     public abstract int f(double d);
    //...

}
```

```
class C extends A {
    public int f(double d) {
        // corpo di f
    }
    // ...
}

abstract class D extends B {
     public abstract int f(double d);
    // …
}
```

Classe Astratta (II)

È possibile dichiarare astratta una classe priva di metodi astratti

```
public abstract class A {
    public A() { x = 0; }
    public int incr(int i) { return x += i; }
    public int decr(int d) { return x -= d; }
    int x;
}
```

L'obiettivo è quello di impedire ai programmatori di creare oggetti istanza di questa classe ma di consentirgli di creare sottoclassi di questa classe

Riferimenti e conversioni

- Non è possibile creare oggetti istanza di una classe astratta ma è possibile definire riferimenti a tale classe
- Un riferimento a classe astratta può riferire un oggetto di una *sottoclasse concreta*

```
public abstract class A {/* classe astratta */}
public class C extends A {/* classe concreta */}
A a; // OK
a = new A(); // compile-time error
a = new C(); // OK
a = null; // OK
```

Proprietà

- L'obiettivo di una classe astratta è di obbligare i programmatori a definire sottoclassi
- L'obiettivo di un metodo astratto è di evitare
  - la complicazione di metodi predefiniti inutili e che
  - questi metodi siano ereditati per errore
- Le classi astratte differiscono dalle interfacce perché possono avere sia variabili istanza sia metodi concreti

Metodi e classi finali

- Una classe che non può essere estesa è detta ***finale***

```
final class A extends B implements C {/* implementazione */ }
class X extends A {/* impl. */ } // compile-time error
```

- Un metodo che non può essere sovrascritto è detto finale

```
class A {
    public final int f(double d) {/* corpo */}
    // altri metodi
}
class B extend A {
    public int f(double d) {/* corpo */} // errore
}
```

Motivazioni

Efficienza

- Un metodo finale non può essere sovrascritto, per cui il compilatore può tradurre la chiamata con lo static binding che è più efficiente del dynamic binding.

  Addirittura, il compilatore può tradurre un metodo finale semplice con codice *inline*

Affidabilità

- Si impedisce che una sottoclasse ridefinisca il metodo e gli faccia fare una cosa completamente diversa

La classe **String**

La classe **String** è un esempio di classe finale

- Le chiamate ai metodi di **String** sono ottimizzati
- Le stringhe sono oggetti immutabili: nessuno può definire una sottoclasse che altera questo comportamento