Il Linguaggio Java

Le interfacce

La classe BankAccount

```
public class BankAccount {
        public BankAccount() { balance = 0; }
        public BankAccount(double initialBalance) {
                balance = initialBalance; }
        public void deposit(double amount) {
                balance += amount;}
        public void  withdraw(double amount) {
                balance -= amount;}
        public double  getBalance() {
                return balance; }
        private double balance;
}
```

La classe SavingsAccount

Un **BankAccount** che frutta un tasso di interesse fisso (**interestRate**) sui depositi

```
public class SavingsAccount extends BankAccount {
    private double interestRate;
    public SavingsAccount(double rate){
     interestRate = rate; }
    public void addInterest() {
     double interest = getBalance() * interestRate / 100;
          deposit(interest); }
}
```

Ordinamento dei conti

- **Problema**
  si vogliono ordinare i libretti di risparmio (**SavingsAccount**) in base al loro tasso di interesse
- **Requisito**
  La classe deve essere ***ordinabile***, cioè gli oggetti della classe devono essere ***confrontabili*** tra di loro

Ordinamento dei conti

Un approccio

1. Definire una classe **Comparable** molto "generica" che definisce un criterio naturale di confronto tra **Object**

```
public class Comparable {
        public int compareTo(Object other) {
          // default body
        }
    }
```

2. Utilizzare il metodo statico **java.utils.Arrays.sort** che ordina un array di **Object** secondo il loro **ordine naturale** (**sort** usa **compareTo**)

Ordinamento dei conti

3. Definire la classe **SavingsAccount** derivando da **Comparable** e sovrascrivendo il metodo **compareTo** in modo da definire un metodo naturale di ordinamento per libretti di risparmio.

```
public class SavingsAccount extends BankAccount, Comparable {
    public int compareTo(Object other) {
        // ridefinizione del metodo
    }
    // rest of the body}
```

Questo approccio non funziona perché in Java l'ereditarietà multipla non è permessa

Ordinamento dei conti

Questo approccio non funziona perché in Java l'ereditarietà multipla non è permessa

Interfacce

- Un'interfaccia definisce un ***protocollo di comportamento*** che può essere ***implementato*** da una qualunque classe nella gerarchia delle classi
- Un'interfaccia definisce un insieme di metodi ma non li implementa
- Una classe che implementa l'interfaccia *deve* implementare ***tutti*** i metodi facendo proprio il comportamento definito dall'interfaccia

Differenze tra classe astratta ed interfaccia

- Un'interfaccia non può avere variabili istanza
- Tutti i metodi di un interfaccia sono astratti
- Tutti i metodi dell'interfaccia sono automaticamente pubblici
- Un'interfaccia non si estende ma si *implementa*
- Una classe può estendere solo una superclasse ma può implementare molte interfacce
- Un'interfaccia non fa parte della gerarchia delle classi (*classi scorrelate possono implementare la stessa interfaccia*)

L'interfaccia Comparable

```
public interface Comparable {
    int compareTo(Object other); // public di default
}
```

- L'interfaccia **Comparable** (**java.lang**) permette di imporre un ***ordine naturale*** tra gli oggetti di una classe
- L'unico metodo **compareTo** viene detto il ***metodo naturale di confronto***
- Il metodo **compareTo** confronta **questo** oggetto con l'oggetto **other** e ritorna (i) un valore negativo, (i) il valore zero o (iii) un valore positivo se questo oggetto è minore, uguale o maggiore di **other**

Ordinamento dei conti

```
public class SavingsAccount  extends BankAccount
                             implements Comparable {
    public int compareTo(Object other) {
        SavingsAccount o = (SavingsAccount)other;
        if ( interestRate < o.interestRate ) return –1;
        if ( interestRate > o.interestRate ) return 1;
        return 0;
    }
    // …
}
```

Il metodo sort di Arrays

- Il metodo **public static void sort(Object[] a)** ordina l'array **a[]** in ordine crescente, *secondo l'ordine naturale degli elementi*. Cioè,
- gli elementi dell'array **a[]** devono
  - implementare l'interfaccia **Comparable**
  - essere mutuamente confrontabili

Ordinamento dei conti (II)

```
import java.util.Arrays;
import java.util.Random
//...
Random rand = new Random();
SavingsAccount[] s = new SavingsAccount[rand.nextInt(100)];

// inizializzazione di s[i]
for (int i = 0; i < s.length; i++)
   s[i] = new SavingsAccount(rand.nextInt(10));

// Ordinamento dei conti
Arrays.sort(s);
```

Proprietà delle interfacce (I)

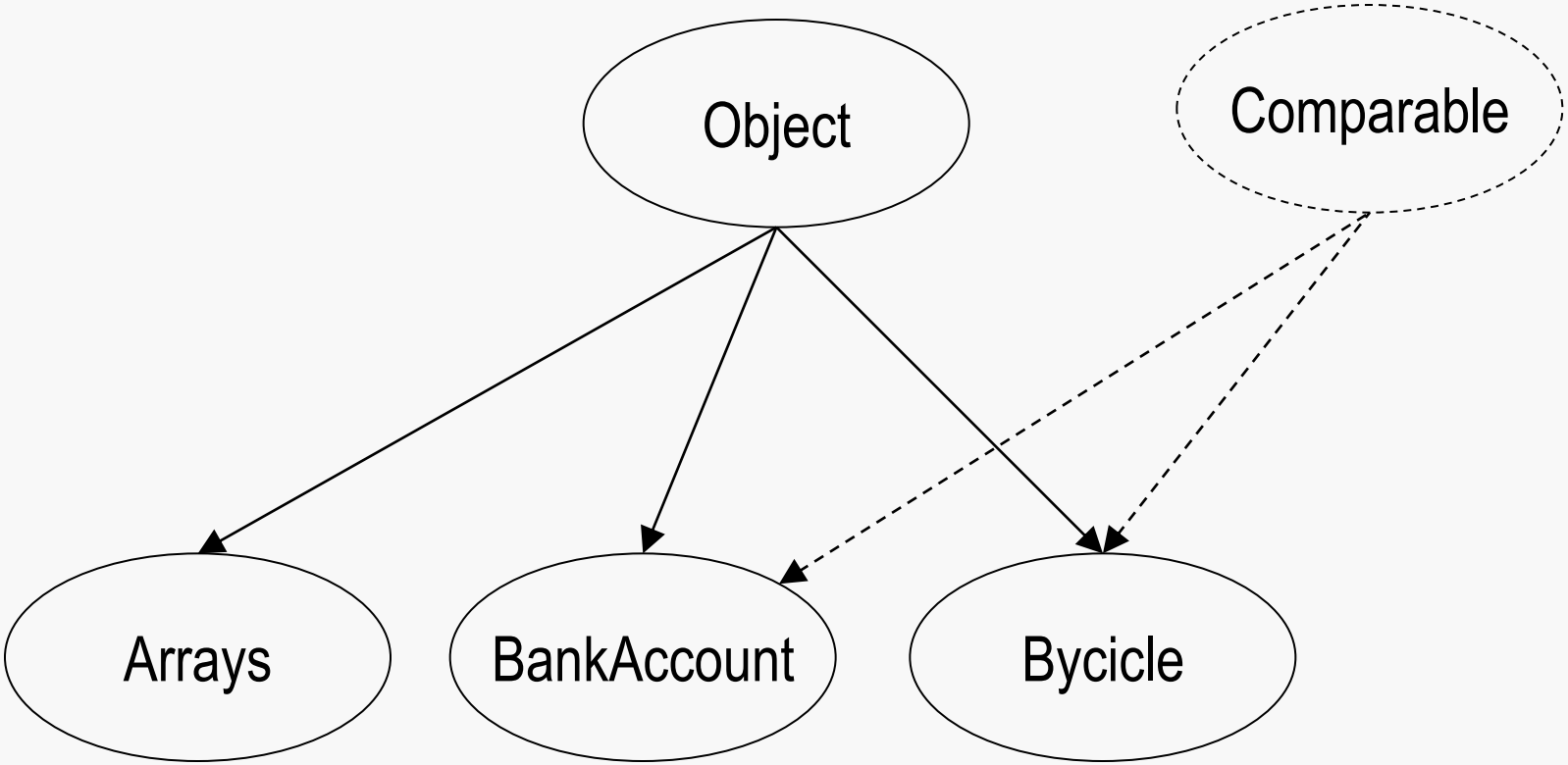
- Una classe può avere una sola superclasse ma può implementare qualsiasi numero di interfacce

```
public class SavingsAccount extends BankAccount
                            implements Comparable, Cloneable {
    // ...
}
```

- Un'interfaccia non è una classe per cui non è possibile crearne oggetti istanza;

```
Comparable x; // Riferimento interfaccia: OK
x = new Comparable(); // Errore di compilazione
```

Proprietà delle interfacce



Le interfacce evitano di forzare relazioni tra classi che logicamente non sono richieste o necessarie

Riferimenti e conversioni

- Sono ammessi riferimenti interfaccia che riferiscono oggetti istanza di classi che implementano l'interfaccia

  Comparable x = new SavingsAccount(10);

- Quando una classe implementa un'interfaccia è possibile convertire un riferimento di classe in un riferimento di interfaccia

  SavingsAccount s = new SavingsAccount();
  Comparable c = s;

- È possibile passare un riferimento di classe a metodi che si aspettano riferimenti di interfaccia

Costanti nelle interfacce

- Un'interfaccia non può avere variabili istanza ma può avere costanti che verranno ereditate da tutte le classi che implementano l'interfaccia

```
public interface MiaInterfaccia {
    int pippo(int a);
    int COST_1 = 1;
    int COST_2 = 2;
}
```

- Le variabili di un interfaccia sono automaticamente **public static final**