Comparable and comparator

Info 2 – 22/3/19

Esempio: Comparable

- Altro esempio di codice generico con l'utilizzo del polimorfismo

- Metodo statico *sort()* della classe *Collections*, in grado di ordinare una lista qualsiasi.

```
List list = ...;
Collections.sort(list);
```

Comparable (1)

- I singoli oggetti della lista possono appartenere a una classe qualsiasi, a patto che implementi l'interfaccia Comparable.

```
public interface Comparable<T>
{
     int compareTo(T other);
}
```

- **La chiamata a *compareTo()* restituisce un valore negativo se l'oggetto invocante precede l'oggetto parametro, zero se i due oggetti sono uguali e un valore positivo**

Comparable (2)

- Come mai tutti gli oggetti devono essere di tipo Comparable?

- Perché l'algoritmo di sort, chiamando compareTo() riesce a decidere gli spostamenti degli oggetti, senza dover conoscere il loro vero tipo.

```
Comparable<...> object1 = ...;
if (object1.compareTo(object2) > 0){
   sposta object1 rispetto a object2
}
```

Comparable (3)

Esempio 1: String realizza Comparable

```
List<String> countries = new ArrayList<String>();
countries.add(“Switzerland);
countries.add(“Belgium”);
countries.add(“Germany”);

Collections.sort(countries);
...
```

Comparable (4)

Esempio 2: Realizzazione di una nuova classe

- Criterio di ordinamento: dimensione della superficie del territorio

```
public class Country implements Comparable<Country> {
    private String fName;
    private double fArea;

    public Country(String name, double area){
        fName = name;
        fArea = area;
    }

    public String getName() {
        return fName;
    }

    public double getArea() {
        return fArea;
    }

    public int compareTo(Country other) {
        if (fArea < other.getArea()){
            return -1;
        }
        if (fArea > other.getArea()){
            return 1;
        }
        return 0;
    }
}
```

Comparable (5)

- Utilizzo della classe Country, con ordinamento secondo la grandezza in km2

```
public class CountryTry {
    public static void main(String[] args) {
        List<Country> countries = new ArrayList<Country>();
        countries.add(new Country("Belgium",77000));
        countries.add(new Country("Switzerland",41000));
        countries.add(new Country("Uruguay",440000));
        ...
        Collections.sort(countries);

        for (Country country : countries){
           System.out.println(country.getName() + " " + country.getArea());
        }
    }
}
```

Interfaccia Comparator (1)

- Se ora volessimo ordinare le stesse nazioni dell'esempio 2 in base al nome, invece che in base alla superficie, dovremmo ridefinire il metodo compareTo().

- Oltre che essere scomodo, ci obbligherebbe a definire sottoclassi unicamente per distinguere diversi metodi compareTo() (modifica del design per scopi che con il design nulla hanno a che vedere)

Interfaccia Comparator (2)

- Soluzione: utilizzo di un overloading di sort(), che accetta come secondo parametro un oggetto Comparator

- Gli oggetti presenti in List vengono ordinati in base all'ordinamento definito in Comparator.

```
class Collections {
    ...
    public static void sort(List<T> list, Comparator<? super T> c) {
      ...
   }
}
```

Interfaccia Comparator (3)

- La lista List può ora contenere oggetti di qualsiasi tipo.
- Non è più necessario che appartengano a classi che implementino un'interfaccia particolare.

```
public class Country {
    private String fName;
    private double fArea;

    public Country(String name, double area){
        fName = name;
        fArea = area;
    }

    public String getName() {
        return fName;
    }

    public double getArea() {
        return fArea;
    }
}
```

Interfaccia Comparator (4)

- Devo però implementare Comparator

```
public class ComparatorByName implements Comparator<Country> {
    public int compare(Country country1, Country country2) {
        String name1 = country1.getName();
        String name2 = country2.getName();

        return name1.compareTo(name2);   //implementato in String
    }
}
```

Interfaccia Comparator (5)

- Se volessi confrontare guardando l'area:

```
class ComparatorByArea implements Comparator<Country>{

@Override
public int compare(Country o1, Country o2) {
        return Double.compare(o1.getArea(), o2.getArea());
}

}
```

Interfaccia Comparator (6)

- **Utilizzo di *ComparatorByName***

```
public class CountryTry2 {
    public static void main(String[] args) {
        List<Country> countries = new ArrayList<Country>();
        countries.add(new Country("Switzerland", 15000));
        countries.add(new Country("Uruguay", 170000));
        countries.add(new Country("Belgium", 30000));
        ...
        Collections.sort(countries, new ComparatorByName());

        for (Country country : countries) {
           System.out.println(country.getName() + "  " + country.getArea());
        }
    }
}
```