Classi Generics

Nel definire una classe (o un'interfaccia) si può far riferimento a classi generiche , cioè che hanno la funzione di “parametro” a cui si fa riferimento in modo generico e che verranno specificate solo al momento della compilazione.

```
/**
 * Permette di incapsulare un dato che sia di una classe generica
 * @author giovanni.ragno
 */
public class Contenitore<E> {
    E dato;

    /**
     * Costruttore
     * @param dato il dato da memorizzare
     */
    public Contenitore(E dato) {
        this.dato = dato;
    }

    /**
     * Getter
     * @return il dato contennto
     */
    public E getDato() {
        return dato;
    }

    /**
     * Setter
     * @param dato il dato da memorizzare
     */
    public void setDato(E dato) {
        this.dato = dato;
    }

    /**
     * Overridde di toString, utile per le verifiche in sviluppo
     * @return
     */
    @Override
    public String toString() {
        return "Contenitore{" + "dato=" + dato + '}';
    }
}
```

Un semplice test d'uso potrà essere:

```
public class TestGeneric {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        Contenitore<String> cs; //memorizza String
        Contenitore<Integer> ci; //memorizza Integer
        Contenitore<Double> cd; //memorizza Double
        cs=new Contenitore<String>("Ciao");//istanza di un
contenitore di String
        System.out.println(cs);//uso del toString
        System.out.println(cs.getDato());//uso del getter
        cs.setDato("Liberi!");//uso del setter
        System.out.println(cs);
        ci=new Contenitore<Integer>(5);//istanza di un contenitore
di Integer
        System.out.println(ci);
        cd= new Contenitore<Double>(3.5*ci.getDato());//istanza di
un contenitore di Double
        System.out.println("cd="+cd);    }
}
```

la suo esecuzione comporta output:

```
Contenitore{dato=Ciao}
Ciao
Contenitore{dato=Liberi!}
Contenitore{dato=5}
cd=Contenitore{dato=17.5}
```

Le classi parametrizzata possono essere più di una, ecco un esempio con due:

```
/**
 * Esempio di classe con due generic
 * @author giovanni.ragno
 */
public class Coppia<E1,E2> {
    E1 primo;
    E2 secondo;

    /**
     * costruttore
     * @param primo
     * @param secondo
     */
    public Coppia(E1 primo, E2 secondo) {
        this.primo = primo;
        this.secondo = secondo;
    }
```

```
    /**
     * getter primo dato
     * @return
     */
    public E1 getPrimo() {
        return primo;
    }

    /**
     * setter primo dato
     * @param primo
     */
    public void setPrimo(E1 primo) {
        this.primo = primo;
    }

    /**
     * getter secondo dato
     * @param primo
     */
    public E2 getSecondo() {
        return secondo;
    }

    /**
     * setter secondo dato
     * @param primo
     */
    public void setSecondo(E2 secondo) {
        this.secondo = secondo;
    }

    /**
     * Overridde di toString, utile per le verifiche in sviluppo
     * @return
     */
    @Override
    public String toString() {
        return "Coppia{" + "primo=" + primo + ", secondo=" +
secondo + '}';
    }

}
```

e un semplice test:

```
    public static void main(String[] args) {
        Coppia<String,String> css; //coppia di String e String
        Coppia<String,Integer> csi; //Coppia di String e Integer
        Coppia<Double,Integer> cdi; //coppia di Double e Integer
```

```
        Coppia<Integer,Contenitore<String>> cics; //coppia di
Integer e Contenitore di String
        Coppia<Integer,Coppia<String,Double>> cicsd; //coppia di
Integer e Coppia di String e Double
        css=new Coppia<String,String>("Ciao","Mondo!");
        System.out.println("css="+css);
        csi=new Coppia<String,Integer>("Test",-3);
        System.out.println("csi="+csi);
        cdi=new Coppia<Double,Integer>(45.2,34);
        System.out.println("cdi="+cdi);
        Contenitore<String> cs=new Contenitore<String>("Ciao");
        cics= new Coppia<Integer,Contenitore<String>>(10,cs);
        System.out.println("cics"+cics);
        cicsd=new Coppia<Integer,Coppia<String,Double>> (12,new
Coppia<String,Double>("Bello, ma inutile!",-32.1e4));
        System.out.println("cicsd="+cicsd);
    }
}
```

la sua esecuzione produce il seguente output:

```
css=Coppia{primo=Ciao, secondo=Mondo!}
csi=Coppia{primo=Test, secondo=-3}
cdi=Coppia{primo=45.2, secondo=34}
cicsCoppia{primo=10, secondo=Contenitore{dato=Ciao}}
cicsd=Coppia{primo=12, secondo=Coppia{primo=Bello, ma inutile!,
secondo=-321000.0}}
```

cc