Interfacce

- Struttura e sintassi simili a quelli di una classe ma i metodi **NON** sono implementati
- vengono usate per modellare i comportamenti di una classe

```
public interface I {
        public type_0  m_1 (...);
                ⋮
        public  t_1    m_2 (...);
                ⋮
        public  t_n    m_n (...);
}
```

! nuovo datatype

Le interfacce non dichiarano l'implementazione dei metodi ma solamente la signature

↓

descrivono **COSA** può fare un oggetto, non come!

NON è possibile instanziare oggetti da interfacce

Implementazione

Una classe può estendere un'unica classe, ma può implementare più interfacce

- Una classe C può implementare più comportamenti (interfacce):
  - Iterabile
  - Pesabile
  - Clonabile

può implementare più di un interfaccia

```
public class C implements I {...}
```

La classe C deve necessariamente fornire un'implementazione dei metodi definiti nell'interfaccia I, ma può anche implementare anche altri metodi

es: Ascensore

```
public class Ascensore {
        private int  pesoContenuto;
        private static int  PESO_LIMITE;
        private Vector<Pesabile>  contenuto;
```

* const = final
  static = condiviso da tutti

```
public void accogli (Pesabile obj) {
        pesoContenuto += obj.peso();
        //TODO controlli PESO_LIMITE
        contenuto.add(obj);
        }
}
```

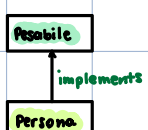
```
public interface Pesabile {
        public double peso();
        }

public class Persona implements Pesabile {
    private final String nome;
    private final String cognome;
    private final double peso;

    public Persona (String nome, String cognome, double peso){
        this.nome = nome;
        this.cognome = cognome;
        this.peso = peso;
        }

    @Override
    public double peso (){
        return peso;
        }
```

Principio di sostituibilità

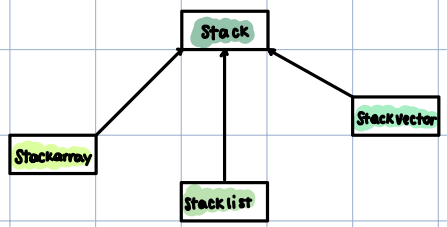


Persona è sottotipo di Pesabile per il principio di sostitutibilità

```
Pesabile p = new Persona(nome, cognome, peso);
p.peso();      ✓ su p posso usare solo i metodi di Pesabile
p.getNome();   ✗ Errore a compile time
   ↙ downcasto
(Persona) p).getNome();
```

Interfaccia Stack

- L'interfaccia Stack descrive la specifica astratta del tipo di dato astratto stack
- Le classi che implementano l'interfaccia denotano tutte le possibili implementazioni di una pila (e.g. tramite array, vector...)




```
public interface Stack {

        public void push (int x);

        public int pop ();

        public boolean isEmpty ();

}
```

```
public class StackVector implements Stack {

        @Override

        public void push (int x) {



        }

        @Override

        public int pop () {



        }

        @Override

        public boolean isEmpty () {



        }
```

```
public class StackArrayFullException extends RuntimeException {

        public StackArrayFullException () {

                super ("Son pieno");

                }
```

```
public class StackArray implements Stack {

        private static final int DIM = 100;

        private int[] elems;

        private int top;



        public StackArray(){

                this.elems = new int [DIM];

                this.top = 0;

        }

        @Override

        public void push (int x) {

                if (top < DIM) {

                        elems[top] = x;
```

```
public class StackArrayEmptyEx extends RuntimeException {

    public StackArrayEmpty () {

        super ("Sou vuoto");

        }
```

```
            top ++; }

        else {

            throw new StackArrayFullException(); }

    @Override

    public int pop ( ) {

        if (isEmpty ()) {

            throw new StackArrayEmptyEx (); }

        else {

            top --;

            return elems [top];

            }

    @Override

    public boolean isEmpty (){

        if (top == 0) {

            return true; }

    }

    public int getElems (int i) {

        return elems [i];

        // check i

    }

public static void main (String [] args){

    Stack S = new StackArray();

    S. push ( 2 );

    S. push ( 3 );
```

```
s.push(4);

System.out.println(s.pop()); // 4

System.out.println(s.pop()); // 3

s.push(5);

System.out.println(s.pop()); // 5
```

Interfaccia CharSequence

descrive i comportamenti di una sequenza di caratteri

Ridefinizione metodo clone

- metodo della classe Object
- effettua una shallow copy di this
- ma potrebbe lanciare un eccezione controllata

Se in una classe voglio usare **clone** devo implementarlo all'interfaccia **CLONABLE**

→ interfaccia vuota che ci dice solo che la classe è clonabile!