Generics (Java)

Meccanismo di astrazione linguistica che consente la parametrizzazione di classi e metodi rispetto ai tipi che utilizzano. È supportata la specifica di un limite superiore sulla posizione nella gerarchia (`<S extends T & I & ...>`) dei tipi con cui si possono istanziare i parametri, in modo da consentire l'utilizzo di operazioni valide per oggetti che rispettano il vincolo (e lower bound con `super`).

Alternativa all'utilizzo di `Object` con cast espliciti e controlli tramite `instanceof`, che permette verifica statica della correttezza dei tipi.

Sottotipo

La relazione di sottotipo è invariante rispetto ai parametri generici: `A<S>` e `A<T>` non sono in relazione di sottotipo (né covariante né invariante) a prescindere da `S` e `T`.

Se invece il tipo concreto U con cui è istanziato il parametro è lo stesso, allora vale la seguente regola:

$$\frac{\texttt{S<E> extends T<E>} \vee \texttt{S<E> implements T<E>}}{\texttt{S<}U\texttt{>} <: \texttt{T<}U\texttt{>}}$$

Il motivo per cui non è covariante è legato alla mutabilità:

```
List<Dogs> dogs = new ArrayList<Dog>(); // List<E> <: ArrayList<E>
List<Animal> animals = dogs;
animals.add(new Cat);            // un List<Dogs> contiene un Cat!
```

Le liste in OCaml sono covarianti rispetto al tipo degli elementi perché sono immutabili.

PECS

Producer Extends, Consumer Super: `extends` in lettura e `super` in scrittura.

```
    <T> void copy(List<? super T> dst,
                  List<? extends T> src);
```

(`?` è un wildcard, una variabile di tipo anonima usata quando un parametro serve esattamente una volta).

Type erasure

A runtime tutti i tipi generici sono trasformati in Object (type erasure) per compatibilità con le vecchie versioni della JVM:

```
    List<String> lst1 = new ArrayList<String>();
    List<Integer> lst2 = new ArrayList<Integer>();
    lst1.getClass() == lst2.getClass()  // => true
```

Il metodo:

```
public static ArrayList<Integer> magic(ArrayList<?> a) {
    return (ArrayList<Integer>) a;
}
```

effettua un cast *non controllato a runtime* visto che le informazioni per farlo non ci sono. Il compilatore emette soltanto un warning (unchecked cast).

Esempio

```
interface Map<K, V> { ... }

class A<T> implements I<T> {
    List<T> lst;

    <S extends Number> public void foo(T bar, S baz) {
        ...
    }
}
```