2023-11-28 - Traits

Traits

I trait sono un meccanismo per aggirare la limitazione dell'*ereditarietà singola* di Java: sono delle interfacce con **stato** e possono avere **metodi sia concreti che astratti**.
Come le interfacce di Java, ogni classe ne può **estendere più di uno**, **non possono essere istanziati** e possono **essere generici**.

```
trait Speak {
    val language: String
    def hello: String
    def sayLanguage: String = "I'm " + language
}

class Person(name: String) extends Speak {
    override val language = "English"
    override def hello = "Hello, I'm " + name
}
```

Una classe può estendere una sola classe o trait, dal secondo in poi viene utilizzata la **keyword** `with`.
Un trait può essere esteso direttamente alla **creazione di un oggetto**, senza dover definire una classe.

```
class Person(name: String)

trait Speak {
    val language: String
    def hello: String
    def sayLanguage: String = "I'm " + language
}

class EnglishPerson(name: String) extends Person(name: String) with Speak {
    override val language = "English"
    override def hello = "Hello, I'm " + name
}
val john = new EnglishPerson("John")

val giovanni = new Person("Giovanni") with Speak {
    override val language = "Italian"
    override def hello = "Ciao, sono Giovanni"
}
```

L'**ordine** con cui vengono estesi traits è **importante**, stabilisce quale metodo viene ereditato dall'oggetto finale in caso ci siano **conflitti**.
Anche in caso non ci siano conflitti, l'ordine stabilisce l'**ordine di esecuzione del codice** (ogni volta che viene **legato** un trait, il codice associato ad esso viene **eseguito**).

```
trait Greeter {
  def hello: String
}

trait English extends Greeter {
  override def hello = "Hello"
}

trait Italian extends Greeter {
  override def hello = "Ciao"
}

val person1 = new English with Italian
val person2 = new Italian with English
```

```
person1.hello // Ciao
person2.hello // Hello
```

I trait sono **trasversali** alle classi, ovvero **non appartengono** alla gerarchia. Per questo motivo la parola chiave `super` non chiamerà mai un metodo di un trait ma della classe a cui viene legato un trait.