
# Marco Servetto

Victoria University of Wellington

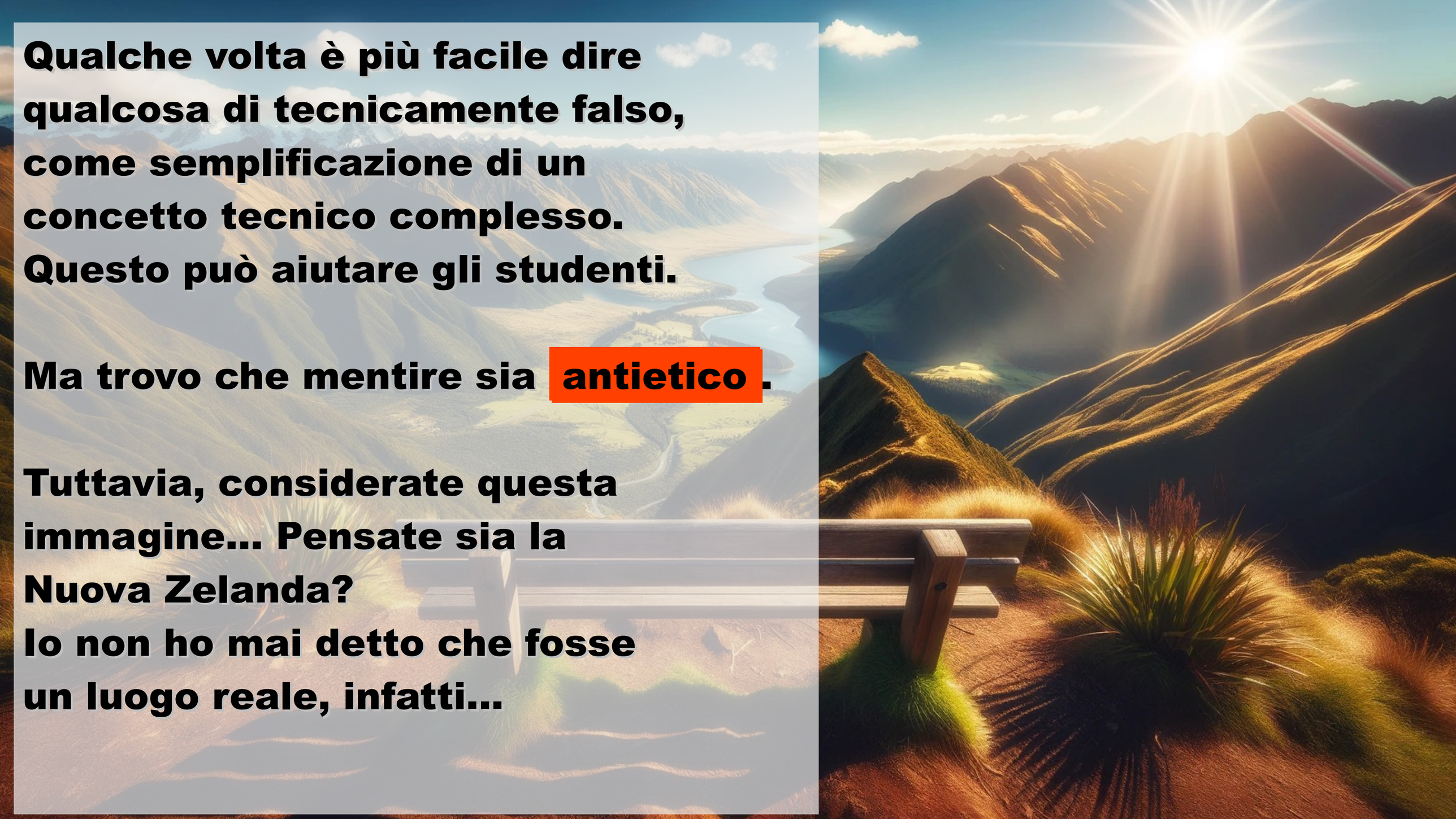
New Zealand


# How To Get The Benefits Of Lying Without Ever Lying

Qualche volta è più facile dire qualcosa di tecnicamente falso, come semplificazione di un concetto tecnico complesso. Questo può aiutare gli studenti.

Ma trovo che mentire sia **antietico**.

Tuttavia, considerate questa immagine... Pensate sia la Nuova Zelanda?
Io non ho mai detto che fosse un luogo reale, infatti...



**Ho semplicemente scritto a ChatGPT:**

**“Draw a beautiful new zealand landscape as seen from POW sitting on a wooden bench on top of a mountain in a sunny day 16:9. Make it look as a real photo”**

**Presentazione in due parti:**

1) **Come essere memorabili senza mentire**
2) **Come mentire senza mentire ed essere ancora piu’ chiari**

# Qui qualche risultato alternativo:


Quando ho cominciato ad insegnare
Le mie lezioni erano noiose e (quasi) standard...
Poi ho imparato ad usare immagini
(Esempio nelle prossime slides)


Java

# Java is now an old language(1996)

# But, recently…..

(after training in secret for 20 years)

# Java: now a modern language!

Now with

- Records,
- Interfaces with default methods
- Lambdas

- We will use those features very intensively in this course!
- We will use ‘class’ more rarely.

# Hint!

When searching for Java examples, Java documentation  or Java help

- Never search for “Java”
  you will get the old stuff.

- Search for “Java 20”, “Java 21”
  or “Java 2023”  to get good stuff!

# Hint!

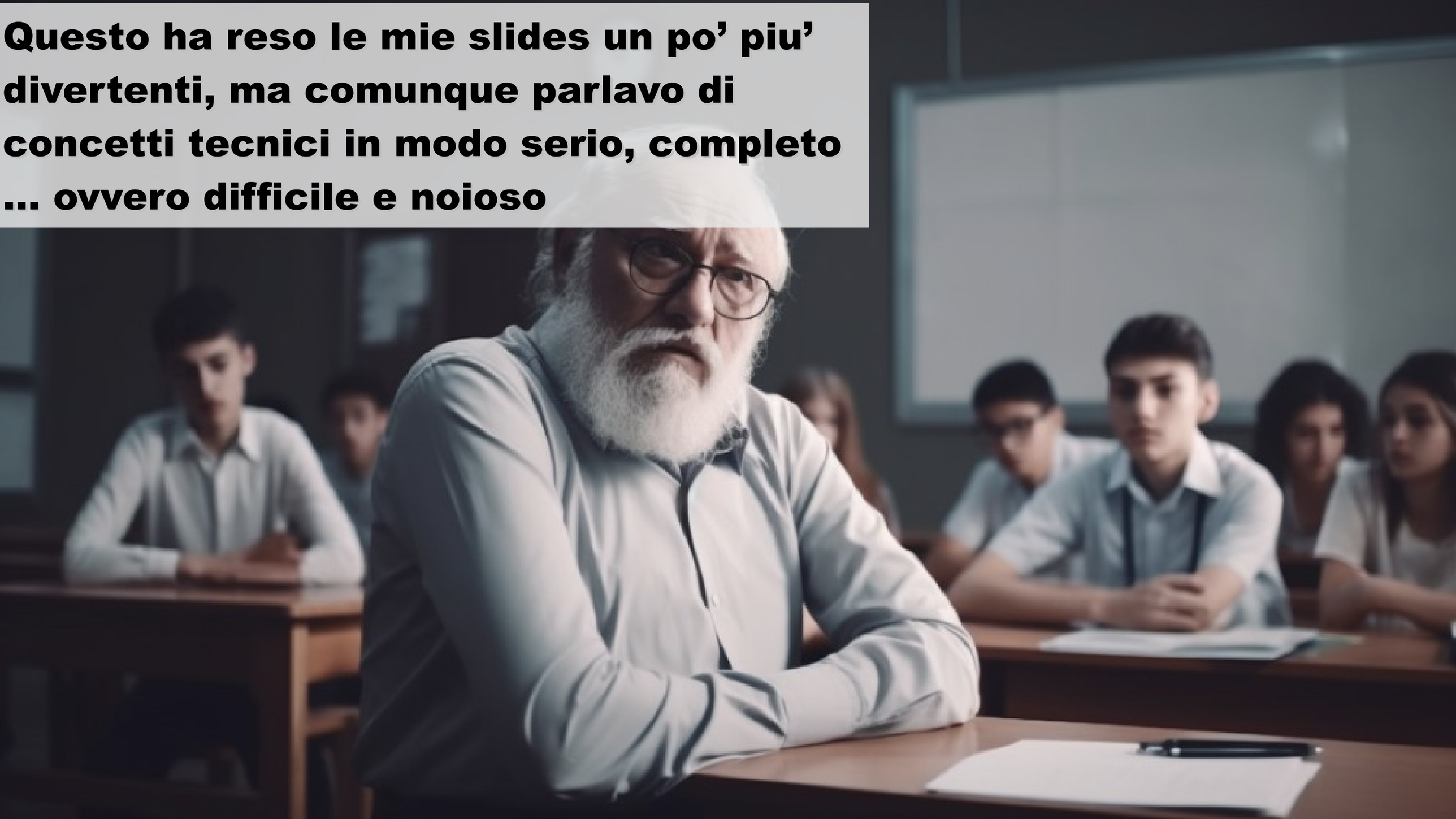
When searching for Java examples, Java documentation or Java help

- Never search for “Java”
  you will get the old stuff.

- Search for “Java 20”, “Java 21”
  or “Java 2023” to get good stuff!

**Cerco di usare immagini ‘memorabili’**
**E poi le riuso quando devo richiamare il concetto.. ma ero limitato dalle immagini che trovavo in rete. Adesso..**

**Questo ha reso le mie slides un po’ piu’ divertenti, ma comunque parlavo di concetti tecnici in modo serio, completo … ovvero difficile e noioso**




Ho Cercato di usare immagini che
catturino l’attenzione
in modo comico ma accurato,
possibilmente appoggiandomi
a personaggi famosi e memorabili...
Purtroppo io sono un vecchio italiano
che insegna in nuova zelanda

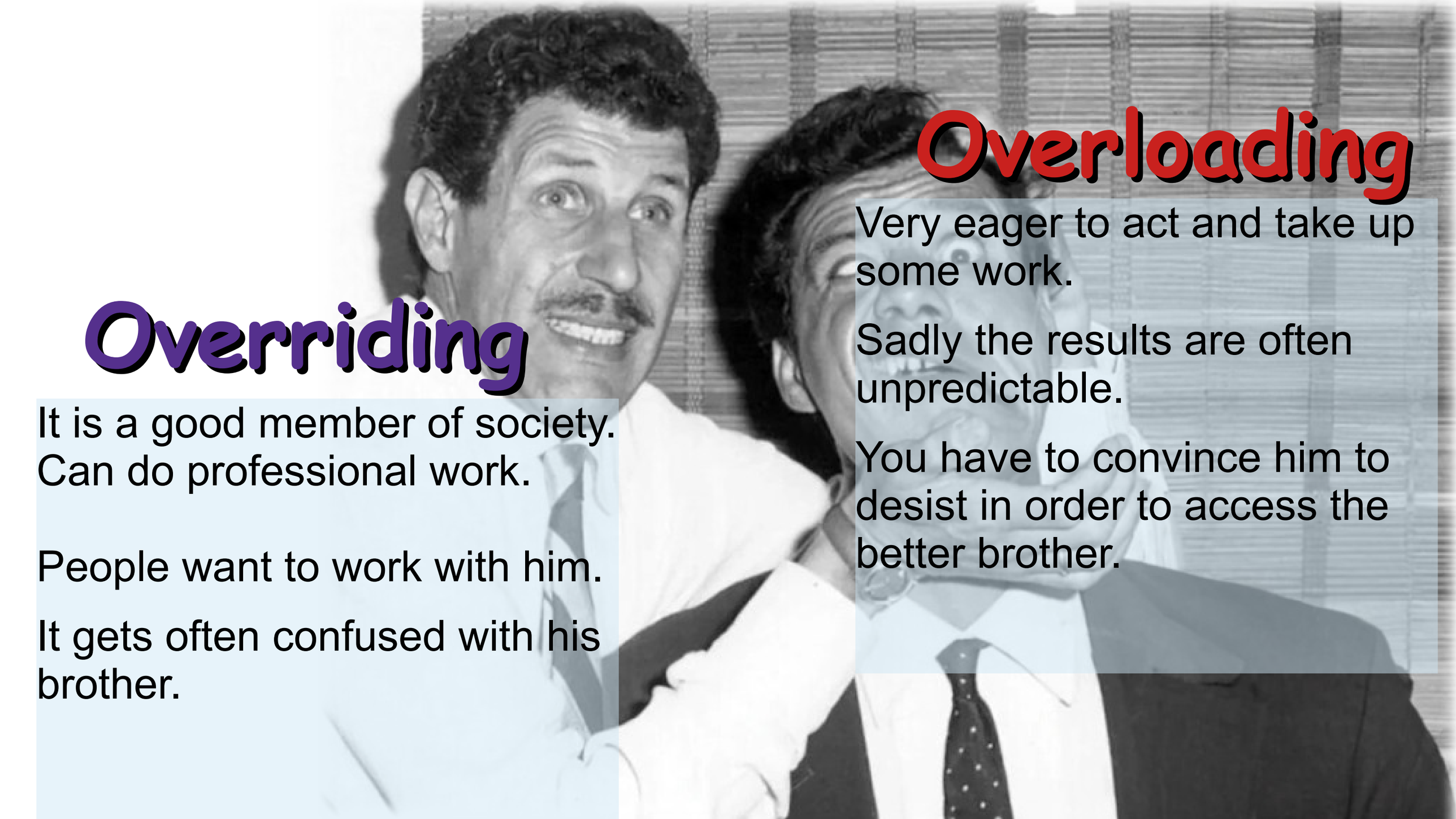

Overloading
Overriding
The two brothers


# Overriding

It is a good member of society.
Can do professional work.

People want to work with him.

It gets often confused with his brother.



# Overriding

It is a good member of society.
Can do professional work.

People want to work with him.

It gets often confused with his brother.

# Overloading

Very eager to act and take up some work.

Sadly the results are often unpredictable.

You have to convince him to desist in order to access the better brother.

It is easy to accidentally assign work to the wrong brother

# Overriding

It is a good member of society. Can do professional work.

People want to work with him.

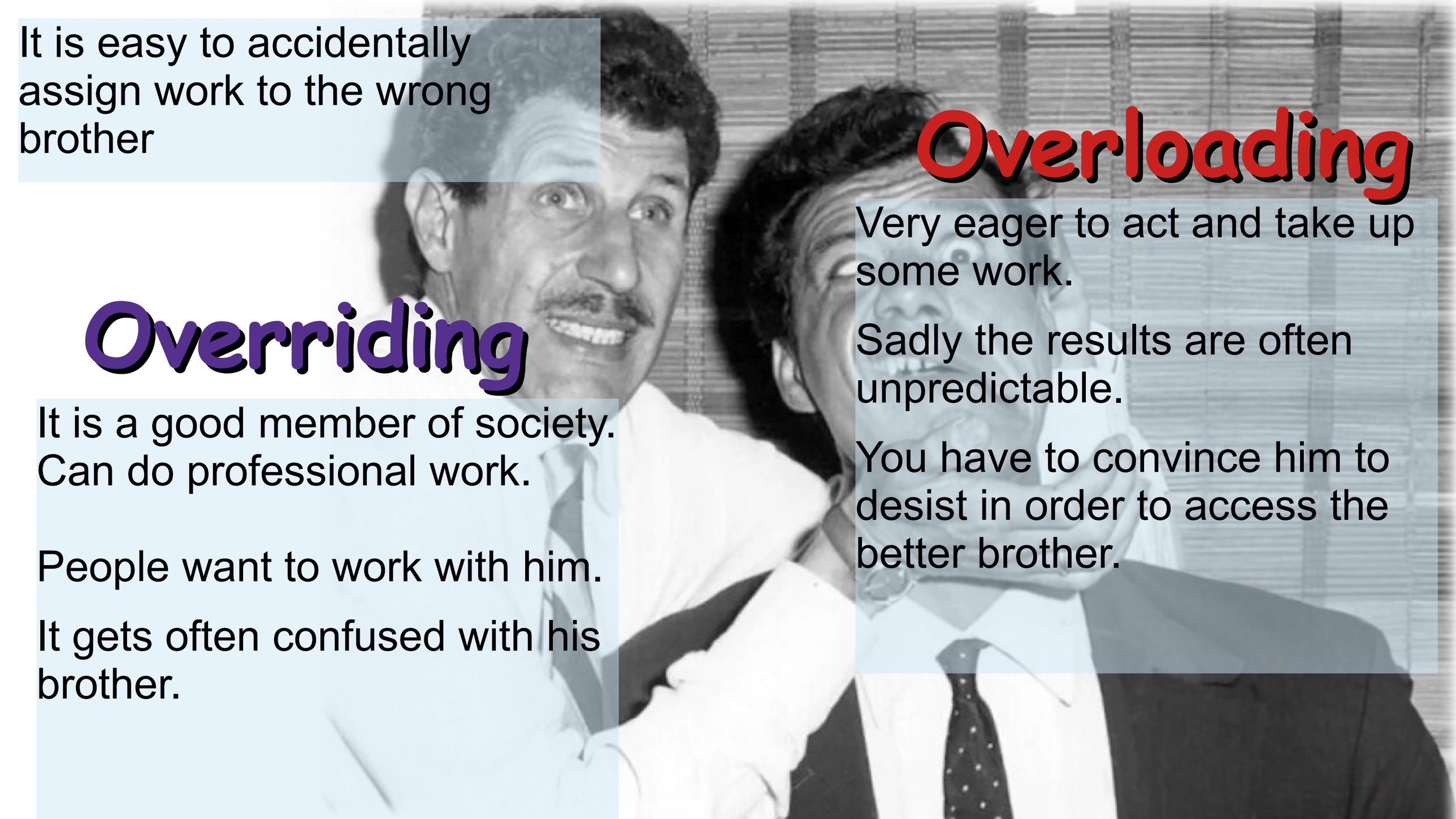
It gets often confused with his brother.

# Overloading

Very eager to act and take up some work.

Sadly the results are often unpredictable.

You have to convince him to desist in order to access the better brother.



**A volte ho grande successo**

# Java exceptions and Norse mythology

# Java exceptions and Norse mythology

- Odin.
- The father of all gods.
- Throwable
- The father of all kinds of exceptions.
- It is very rare to do new Throwable(..).
- Sometime used as a type.
- Quite rare to do catch(Throwable t){..}

# Java exceptions and Norse mythology

- Thor.
- The most iconic.
- Exception
- The most iconic kind of Exception, the first that comes to mind.
- Rare to do
  new Exception(…)
- catch(Exception e){..}
  is often overkill

# Java exceptions and Norse mythology

- Loki.
- The sneaky guy
- Error
- The one whose name causes the most dread.
- Common to do
  new Error(e)
- catch(Error e){..}
  is common in library code

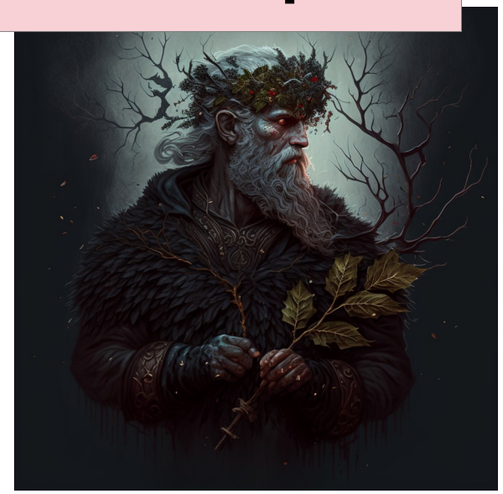
# Java exceptions and Norse mythology

- Hoðr.
- The good sneaky guy with a tragic past: tricked by Loki to kill Baldr, the god of light.
- RuntimeException
- Helps us to solve limitations of Java by tunneling errors around the type system.
- Common to do
  new RuntimeException(e)
- catch(RuntimeException e){..} is common in library code

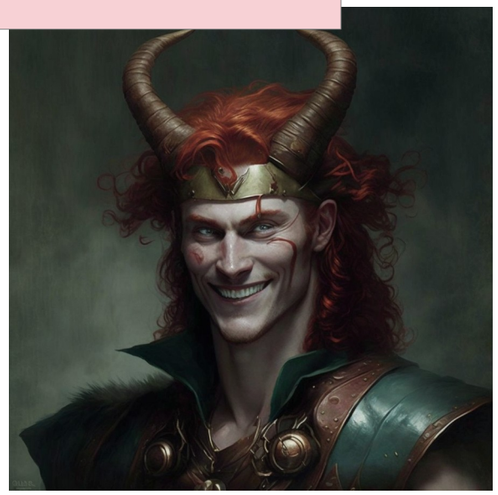
**Throwable**

**Exception**

**RuntimeException**



**Error**



Family relationships:
Throwable(Odin) is the father (supertype) of

- Error (Loki) and
- Exception (Thor). Exception is the father of
  - RuntimeException (Hoðr)

They all spawn their minions

Throwable

Checked

Exception

Checked

Checked

RuntimeException

Unchecked

Unchecked

Error

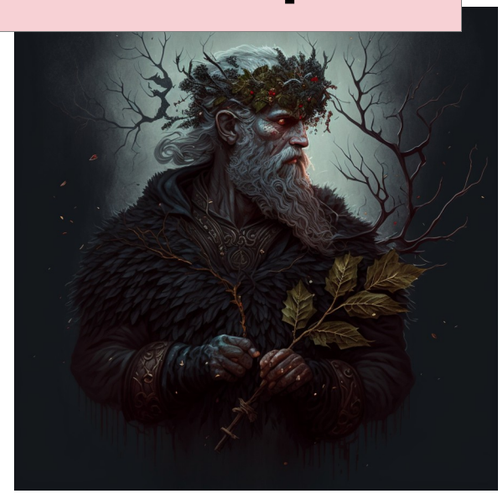
Unchecked

Unchecked

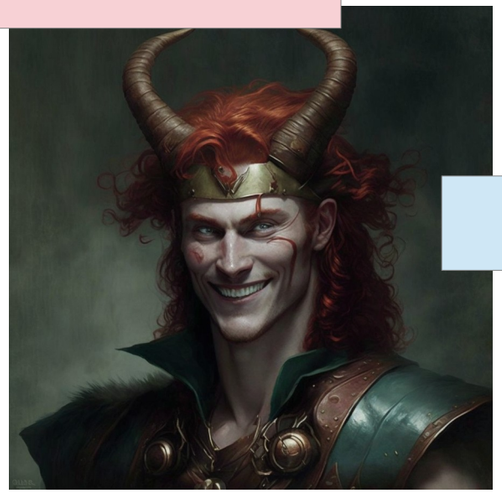
**Throwable**

**Exception**

**RuntimeException**



**Error**



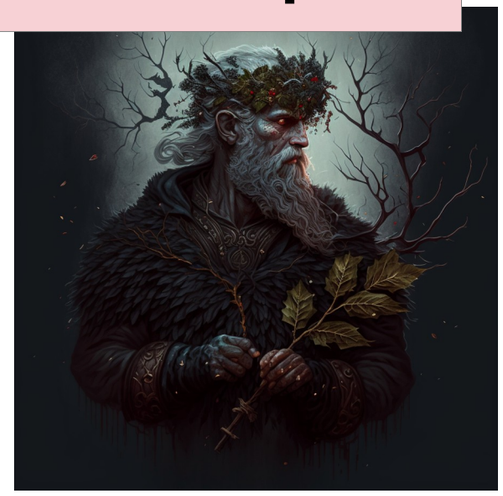
**An Error is a subclass of Throwable**
**that indicates serious problems that a reasonable application should not try to catch. Most such errors are abnormal conditions.**

| | | | |
|---|---|---|---|
| **AssertionError,** | **AWTError,** | **CoderMalfunctionError,** | **IOError,** |
| **LinkageError,** | **ThreadDeath,** | **VirtualMachineError,** | **...** |

## Throwable

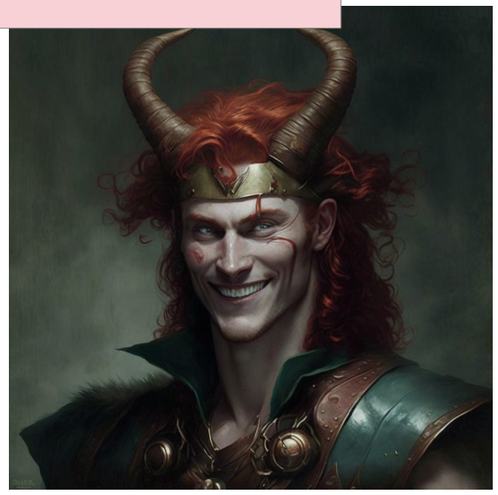
## Exception

## RuntimeException



**The class Exception and its subclasses are a form of Throwable that indicates conditions that a reasonable application might want to catch.**

## Error



**The class Exception and any subclasses that are not also subclasses of RuntimeException are checked exceptions. Checked exceptions need to be declared in a method or constructor's throws clause if they can be thrown by the execution of the method or constructor and propagate outside the boundary.**

**Tons of kids:**

**ExecutionException, FontFormatException, GeneralSecurityException, InterruptedException, IOException, MarshalException, MidiUnavailableException, MimeTypeParseException, ParseException, ReflectiveOperationException, SQLException, TimeoutException, TooManyListenersException, XMLParseException, ...**

## Throwable

## Exception

## RuntimeException

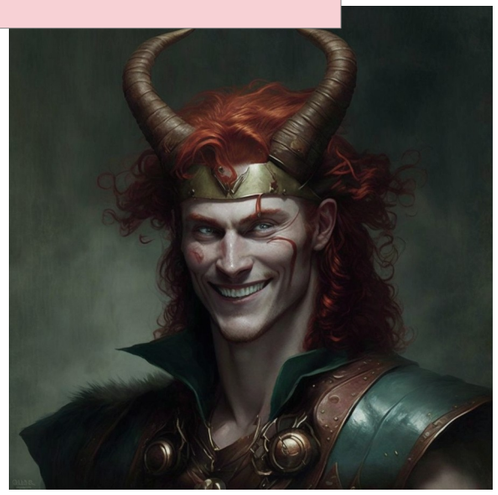
**RuntimeException is the superclass of those exceptions that can be thrown during the normal operation of the Java Virtual Machine.**

**RuntimeException and its subclasses are unchecked exceptions. Unchecked exceptions do not need to be declared in a method or constructor's throws clause**

**Tons of kids. Those are actually useful/relevant**

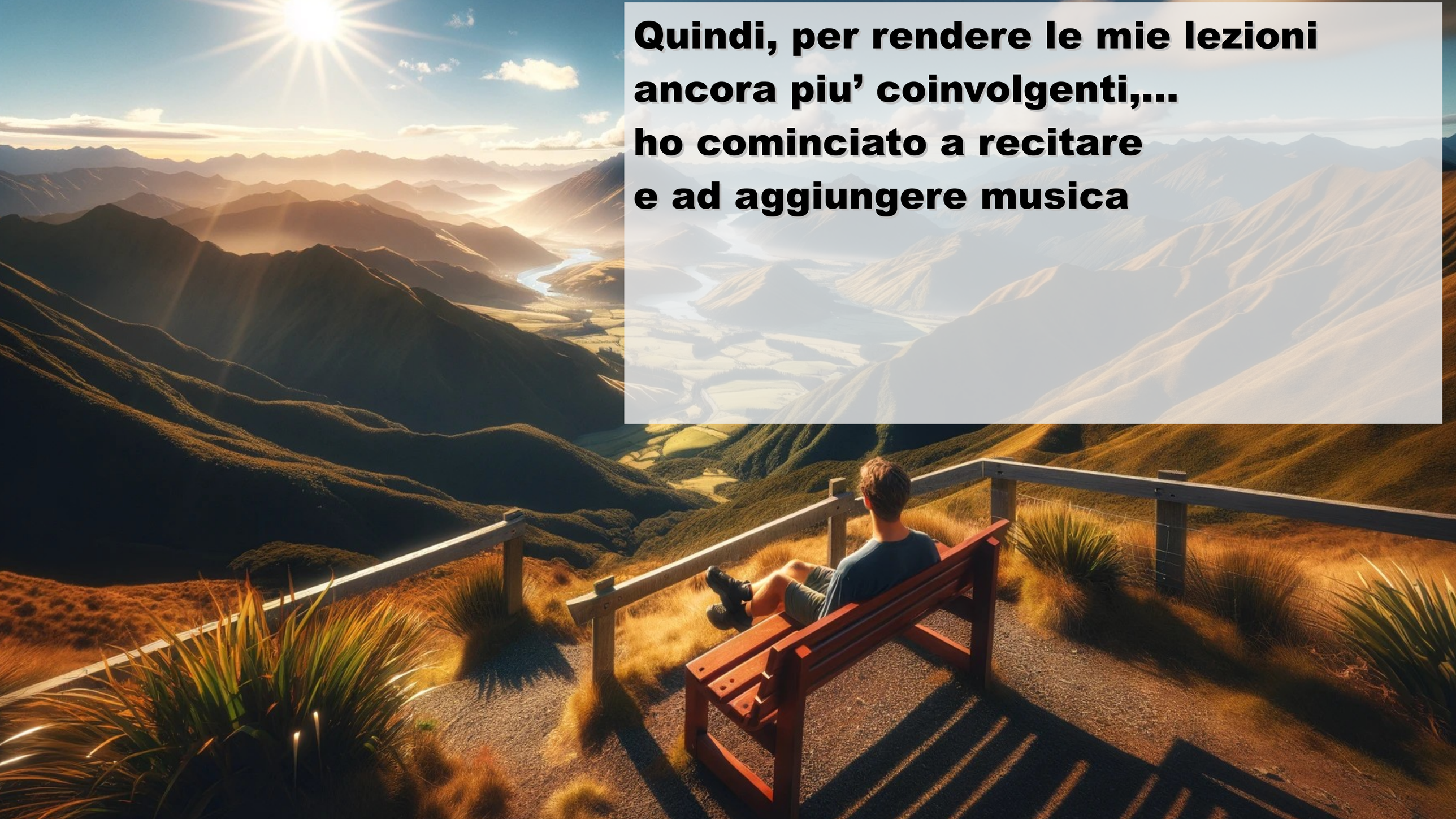
**ArithmeticException, ArrayStoreException,ClassCastException, CompletionException, ConcurrentModificationException, EnumConstantNotPresentException, IllegalArgumentException, IndexOutOfBoundsException, NegativeArraySizeException, NoSuchElementException, NullPointerException, SecurityException, UnmodifiableSetException, UnsupportedOperationException,**

## Error



# More metaphors: catch something

- If the hero is too weak, the monster will win and the hero will be defeated.
- However, with the ‘finally’ block, the hero can say their last words!
- **No matter if they win or lose**, they will say those words before the stack unwinds away from them, sending them back into oblivion.



**Quindi, per rendere le mie lezioni ancora piu’ coinvolgenti,...**
**ho cominciato a recitare**
**e ad aggiungere musica**

# Automated testing -- Why is good

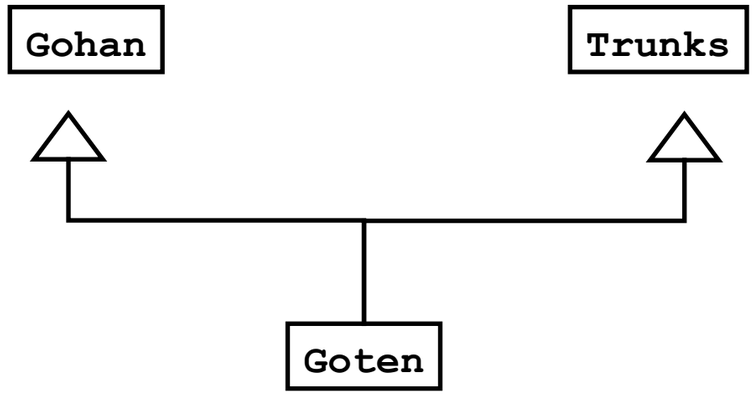
- Don Knuth: this is the story of my life


[...]
Birds don't just fly
They fall down and get up
**Nobody** learns without getting it won
[...]
I wanna try everything
I wanna try even though I could **fail**
I'll keep on making those **new** mistakes
I'll keep on making them **every day**
Those **new mistakes**
Oh oh, try everything

Shakira - Try Everything
https://www.youtube.com/watch?v=c6rP-YP4c5I

# Multiple inheritance

https://www.youtube.com/watch?v=8cpyejx506o




- From Java8, this is now possible!
  - A class cannot have more than one superclass
    - Other languages (e.g. C++) support this
  - But, a class can implement **more than one interface**
  - Interfaces can now have reusable implementation!

```
class Goten extends Gohan, Trunks { … }
```

```
class Goten implements Gohan, Trunks { … }
```

- Risk of multiple inheritance: we may end up with too much stuff,
- A class that is too fat.
  (violating the single
  responsibility principle)

- To avoid this, we need to make the interfaces used to modularize our behaviour as slim as possible

- To avoid this, we need to make the interfaces used to modularize our behaviour as slim as possible

- We can have many different interfaces, sometimes with just a single default method (and the private methods that it uses internally)

- The other methods will be abstract and defined in other interfaces.

**A volte una sola immagine, piccola e mai ripetuta e' meglio e distrai meno**

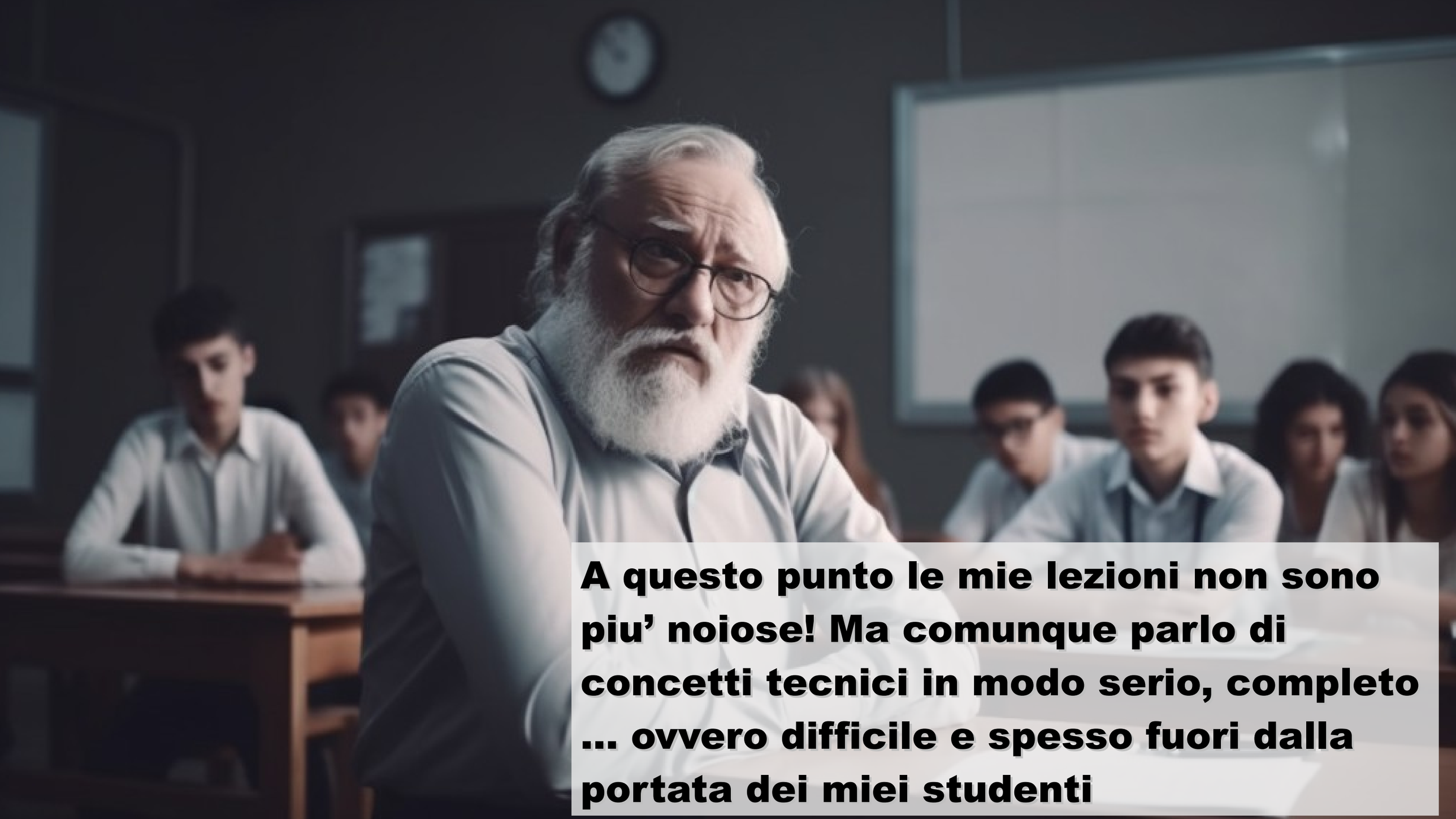
# Property testing

Instead of just testing that an operation on a specific input does exactly what we expect, we can generate some input and check for general properties, for example with minIndex we could generate random sequences of numbers and test that the number at the resulting input is indeed smaller than all the other ones.

Watch: https://www.youtube.com/watch?v=IYzDFHx6QPY

“The lazy programmer's guide to writing thousands of tests”
- Scott Wlaschin

This talk is also known as “The enterprise developer from hell”


A questo punto le mie lezioni non sono piu’ noiose! Ma comunque parlo di concetti tecnici in modo serio, completo … ovvero difficile e spesso fuori dalla portata dei miei studenti

# Zone of Proximal Development (ZPD) and Constructivism

- In Nuova Zelanda ci fanno seguire dei seminari di didattica. L'apprendimento più efficace e' spesso facilitato da coetanei più esperti, quelli appena 'oltre il limite'. Gran parte del nostro apprendimento avviene attraverso l'interazione sociale, dove gli studenti costruiscono nuove comprensioni e conoscenze interagendo con coetanei impegnati nelle stesse o leggermente più avanzate fasi di apprendimento. Questi compagni di classe possono offrire intuizioni e spiegazioni uniche, particolarmente accessibili ad altri studenti.
- Ascoltando queste teorie, e riconoscendole come veritiere sulla base della mia esperienza, mi sono sentito inutile come insegnante. Se gli studenti imparano meglio da altri studenti, quale e' il mio ruolo?

# Idea! Fingo di essere un altro studente quando insegno!

- “Questi compagni di classe possono offrire intuizioni e spiegazioni uniche, particolarmente accessibili ad altri studenti.”
- Perché piu’ accessibili? Perché sono **FALSE!**
- Queste spiegazioni contengono abbastanza verità da permettere agli studenti di completare esercizi ed attività, ma non sono sempre perfettamente corrette. Sono semplificate, e spesso piene di incomprensioni che, nel lungo periodo, potrebbero rivelarsi controproducenti. Tuttavia, nel breve periodo, queste spiegazioni 'imperfette' funzionano.
- Se uno è pronto a raffinare e correggere la propria conoscenza man mano che procede, allora imparare la realtà per approssimazioni successive potrebbe non essere una cattiva idea. **Forse?**


I just.. can not do it.
I can not lie.
I will not lie.
Even if I have to fail
all my students,
(and be fired for it)
I will not lie!



**Ultimate Programming University**

# So, I stated writing a comicbook!

- Quando una cosa e’ facile la spiego direttamente
- Quando e’ difficile uso il fumetto.
- La storia evolve cosi che il concetto sia rilevante.
  I miei personaggi scoprono il concetto e non lo capiscono bene.
  Si aiutano tra di loro, discutono, fanno esercizi, esami etc.

- Neanche I miei personaggi mentono, si limitano a dire cose ragionevoli e ‘in character’ a seconda del livello di competenza che hanno in quel momento nella storia.
- Cosi posso usare spiegazioni imperfette senza mentire.