Absurd difficulty

- Cosa interessante: posso insegnare cose estremamente difficili partendo da cose facili e costruendoci sopra.
- Esempio: Generics at the docs
  - Vediamo il professore che insegna generici in Java, ma non mostriamo la lezione vera e propria
  - I protagonisti della nostra storia vanno al porto a mangiare fish and chips e a studiare generici.
  - Vedono un camion della Pepsi

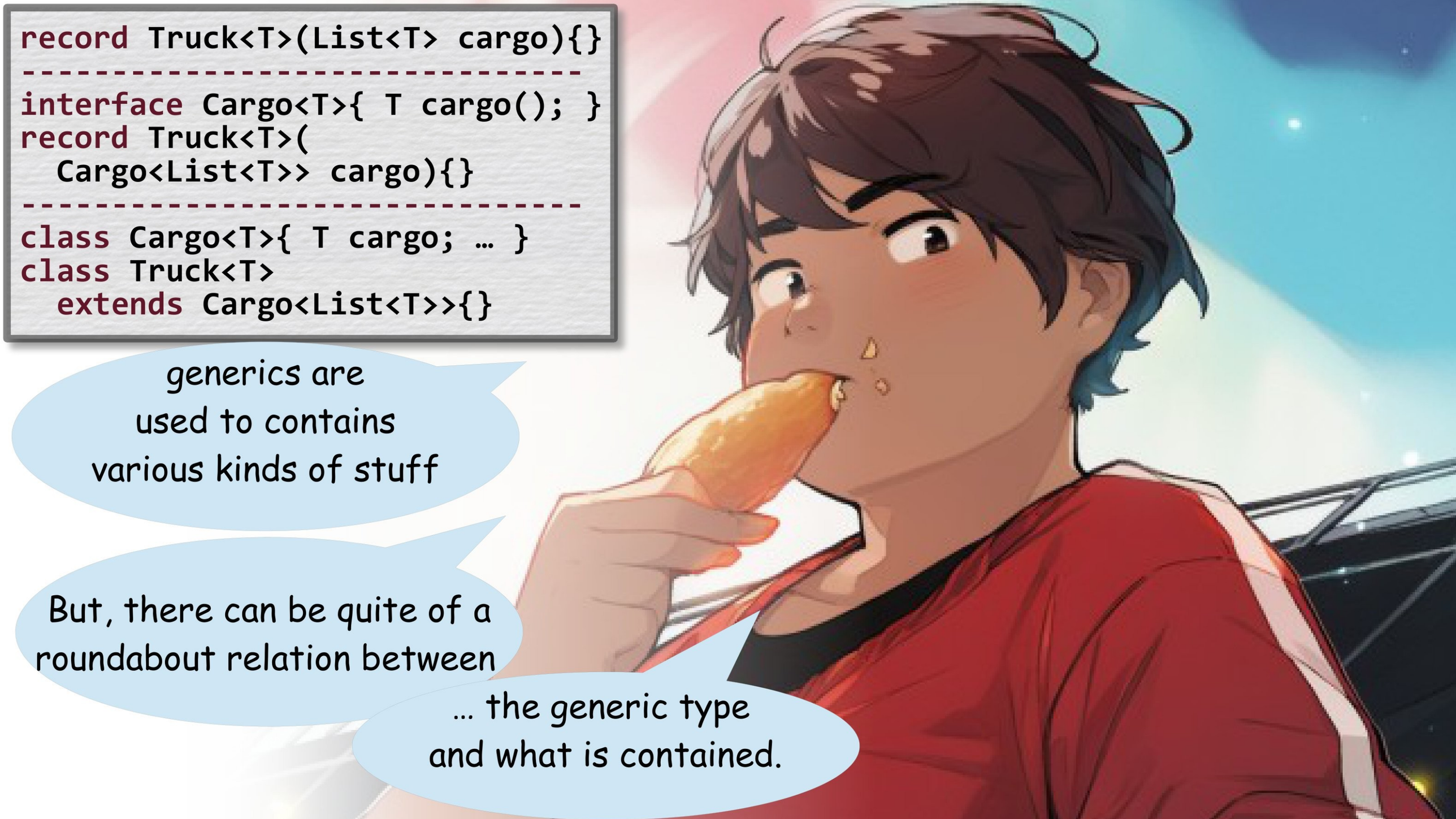

Ah, food!
Nothing beats food.
Let's keep our eyes open, ...
we may find some good inspiration to understand generics better!


Truck<
pepsi
pepsi

Dopo qualche discussione

- Ricky mostra un riassunto del codice discusso, facendo vedere vari modi di codificare un Truck<T> contenente una lista di T come cargo

```
record Truck<T>(List<T> cargo){}
---------------------------------
interface Cargo<T>{ T cargo(); }
record Truck<T>(
  Cargo<List<T>> cargo){}
---------------------------------
class Cargo<T>{ T cargo; … }
class Truck<T>
  extends Cargo<List<T>>{}
```

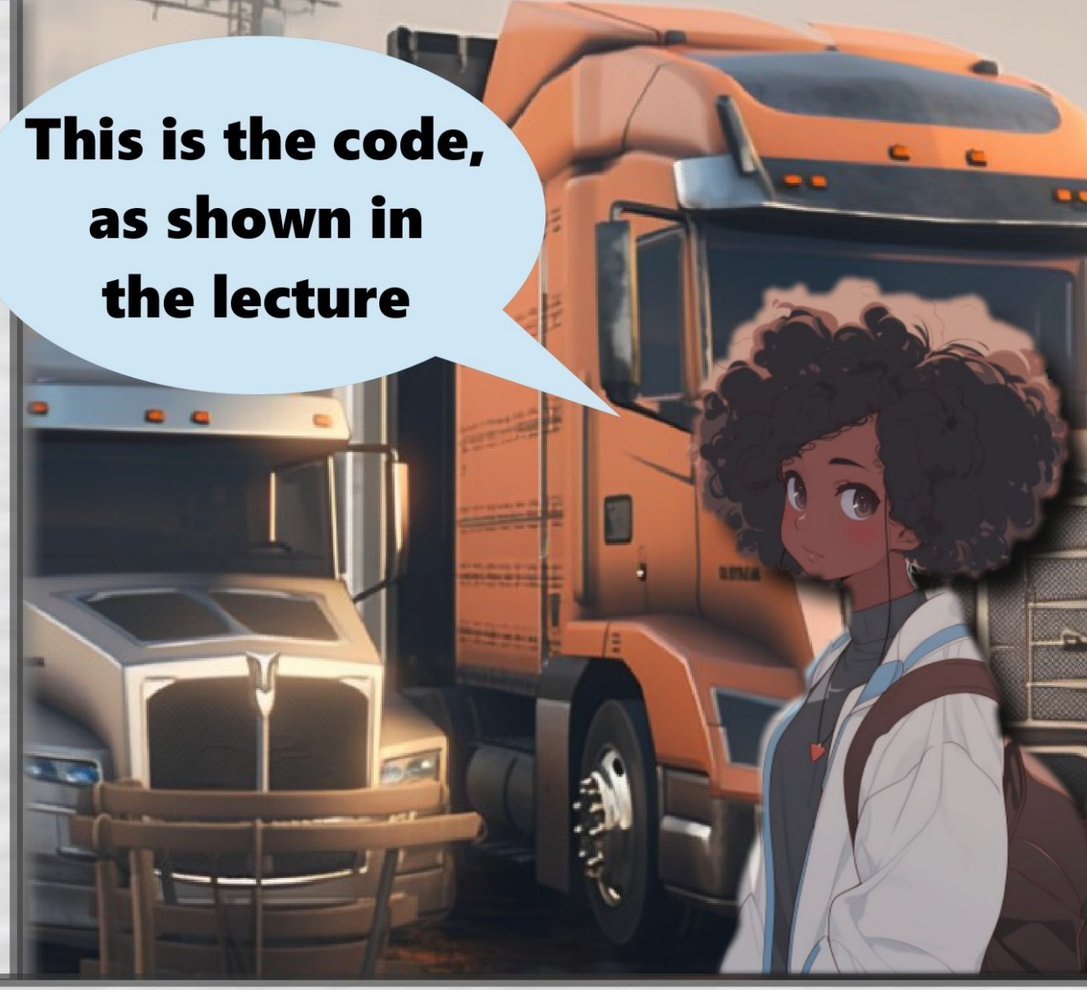



Hey, boys,
I'm Sophia.
You are from our
class, right?
You are talking "generics".
Can we join you?
Altri studenti
si uniscono
alla discussione


Usando due tipi di truck spiego connessioni tra inheritance e generici
Here they are!
I’m Luna, by the way

```
abstract class Truck<T>{
  public abstract int aerodynamics();
  public Cargo<List<T>> cargo(){ return cargo; }
  public Point location(){ return location; }
  public void location(Point p){ location=p; }

  void driveToward(Point destination, int duration){
    ...
    Point wayPoint=...;
    location(wayPoint);
  }
  private Cargo<List<T>> cargo;
  private Point location;
  Truck(Point p, Cargo<List<T>> c){
    location=p; cargo=c;
  }
}
```




```
class Bonneted<T> extends Truck<T>{
  Bonneted(Point p, Cargo<List<T>> c){
    super(p,c);
  }
  public int aerodynamics(){ ... }
}
```

```
class CabOver<T> extends Truck<T>{
  CabOver(Point p, Cargo<List<T>> c){
    super(p,c);
  }
  public int aerodynamics(){ ... }
}
```

```
interface Truck<Self,T>{
  int aerodynamics();
  CargoC<List<T>> cargo();
  Point location();
  Self withLocation(Point p);

  default Self driveToward(Point destination, int duration){
    ...
    Point wayPoint=...;
    return withLocation(wayPoint);
  }
}
```

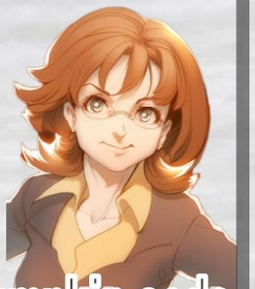
Dopo un interruzione comica

Scopriamo come l'altro professore risolve lo stesso problema

```
record Bonneted<T>
  (Point location, CargoC<List<T>> cargo) implements Truck<Bonneted<T>,T>{
  public Bonneted<T> withLocation(Point p){ return new Bonneted<>(p,cargo); }
  public int aerodynamics(){ ... }
}
record CabOver<T>
 (Point location, CargoC<List<T>> cargo) implements Truck<CabOver<T>,T>{
  public CabOver<T> withLocation(Point p){ return new CabOver<>(p,cargo); }
  public int aerodynamics(){ ... }
}
```

```
interface Truck<Self,T>{
  int aerodynamics();
  CargoC<List<T>> cargo();
  Point location();
  Self withLocation(Point p);

  default Self driveToward(Point destination, int duration){
    ...
    Point wayPoint=...;
    return withLocation(wayPoint);
  }
}
```



Pumpkin code

```
abstract class Truck<T>{
  public abstract int aerodynamics();
  public CargoC<List<T>> cargo(){ return cargo; }
  public Point location(){ return location; }
  public void location(Point p){ location=p; }

  void driveToward(Point destination, int duration){
    ...
    Point wayPoint=...;
    location(wayPoint);
```

Puppon code

```
record Bonne
  (Point loc
  public Bon
  public int
}
record CabOv
 (Point loca
  public Cab
  public int
}
```

Can you understand the difference in these two approaches?
Do you think one is better then the other?

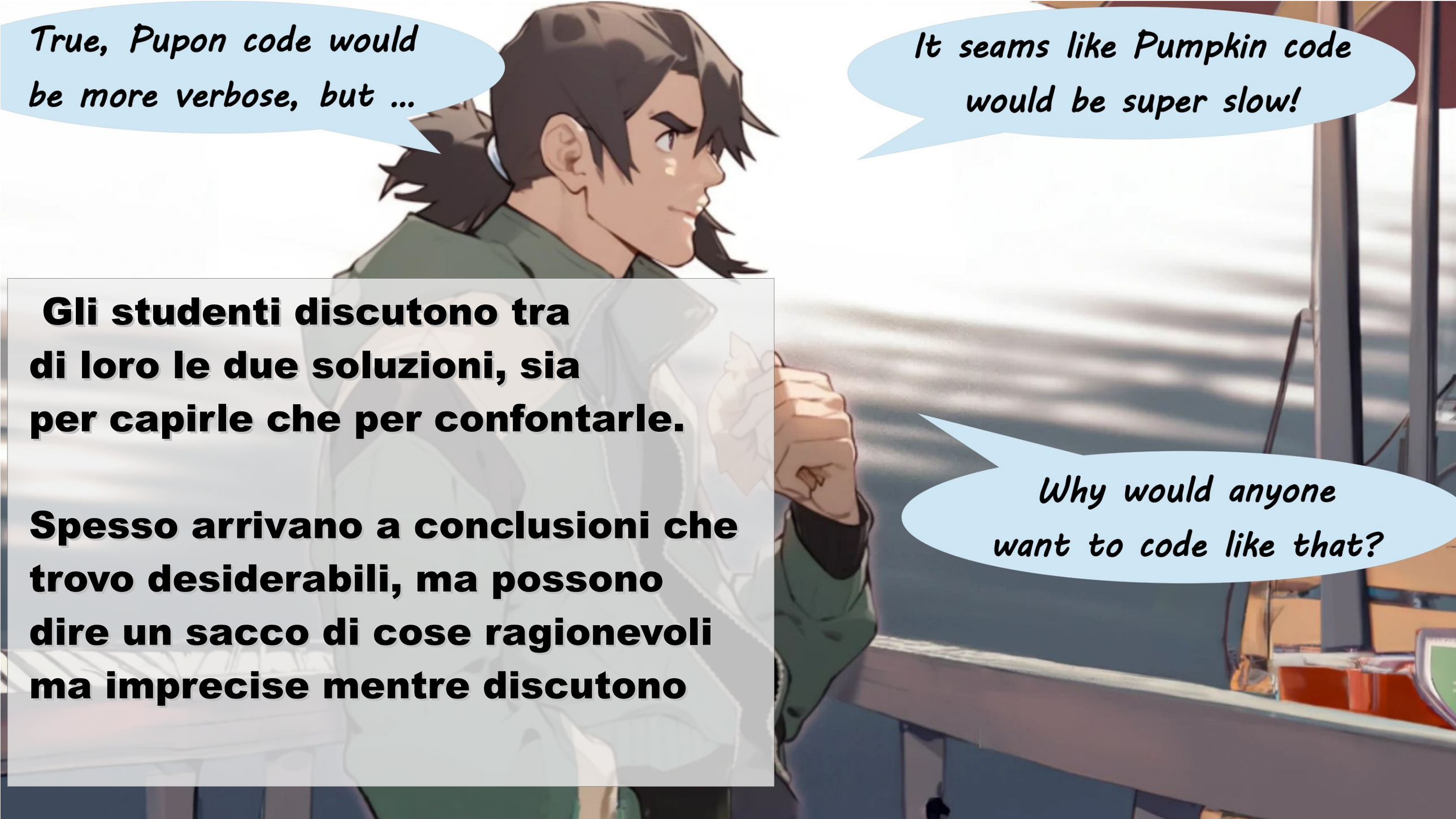
One of these two is not as usable as it should.
Which one? Why?

Wow, this is unexpected

But... how can it be?
They are modeling exactly the same concept!

Ah si.. ci sono i super poteri nella mia storia

Ah si.. ci sono i super poteri nella mia storia



Gli studenti discutono tra di loro le due soluzioni, sia per capirle che per confontarle.

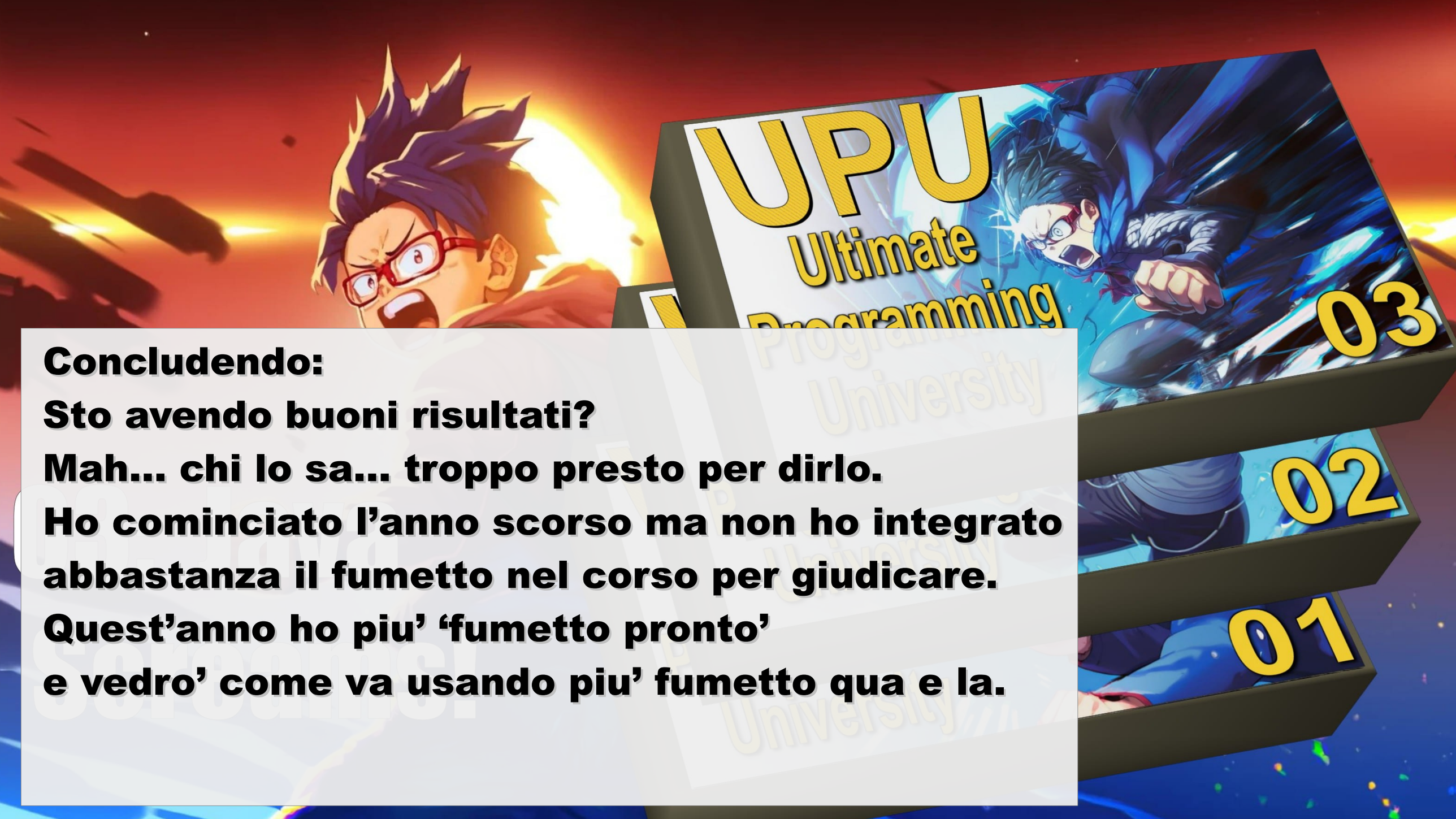
Spesso arrivano a conclusioni che trovo desiderabili, ma possono dire un sacco di cose ragionevoli ma imprecise mentre discutono


Wildcards? Give me a break!
Truck<?, Pepsi >
Si arrabbiano e possono avere discussioni molto vivaci
With the wildcard, the method ‘driveToward’ would just return an Object.
A volte la discussione finisce con un ‘ci siamo quasi’ ma non do mai la soluzione ‘giusta’
Truck<?, Pepsi > truck = ...;
Object pointless = truck.driveToward(..);
//nothing else would compile

Inoltre, posso rappresentare situazioni di stress o anche di fallimento, facendo sentire gli studenti in difficolta'...
meno soli.


My time at UPU ends today
Forse ogni tanto vado un po troppo in la...


UPU
Ultimate
Programming
University
03
University
02
University
01
03: Java
Screams!



Concludendo:

Sto avendo buoni risultati?

Mah... chi lo sa... troppo presto per dirlo.

Ho cominciato l’anno scorso ma non ho integrato abbastanza il fumetto nel corso per giudicare.

Quest’anno ho piu’ ‘fumetto pronto’

e vedro’ come va usando piu’ fumetto qua e la.