Absurd difficulty

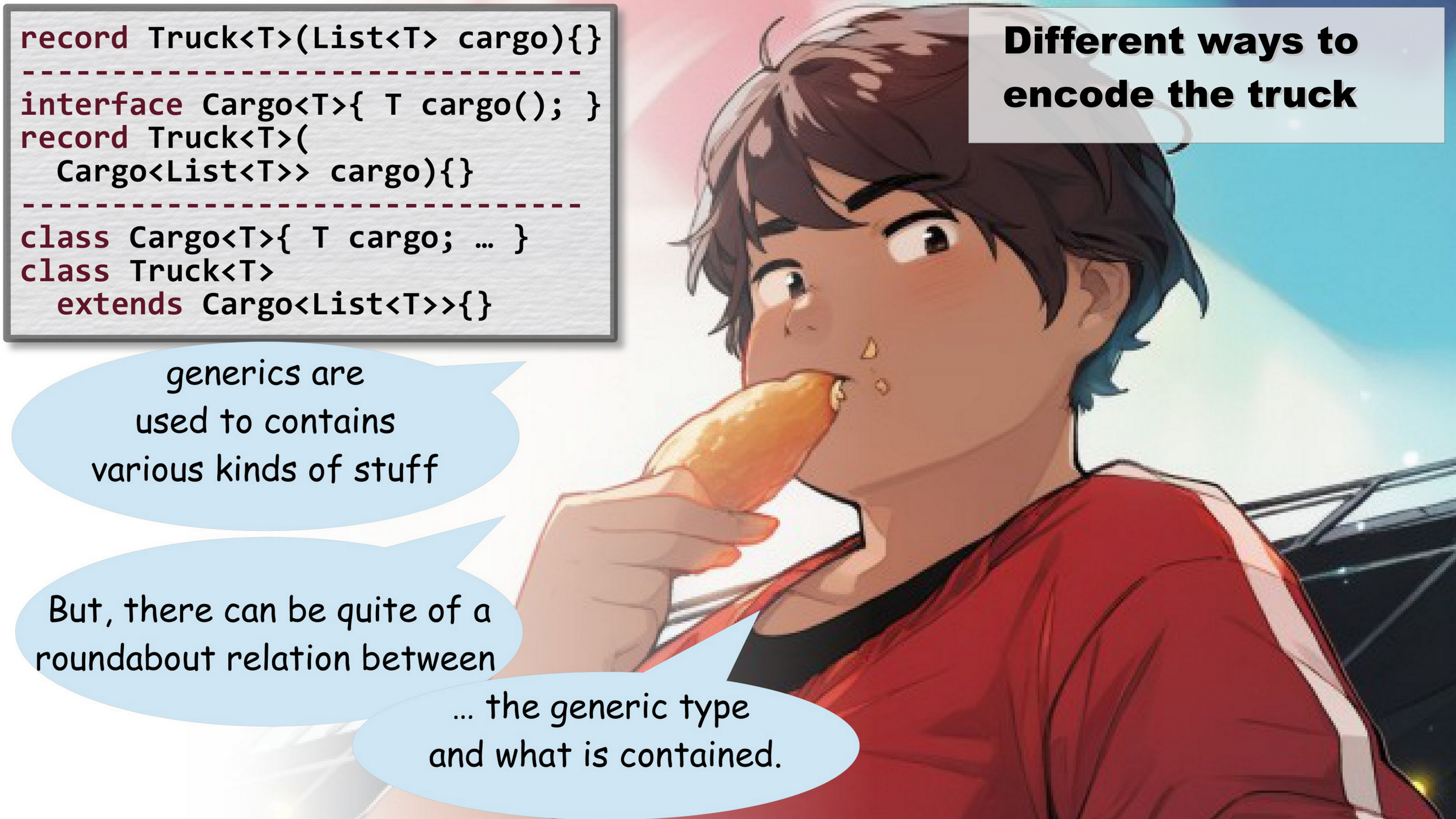
- I try to teach extremely hard concepts starting from mundane things and building on top of them.
- Example: Generics at the docs
  - In the comic-book we see the lecture about generics from far away, without showing the lecture in the details.
  - Our protagonists go to the docks to each fish and chips and to study Java generics.
  - We now see a Pepsi Truck.


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


Truck
pepsi
pepsi

Different ways to encode the truck

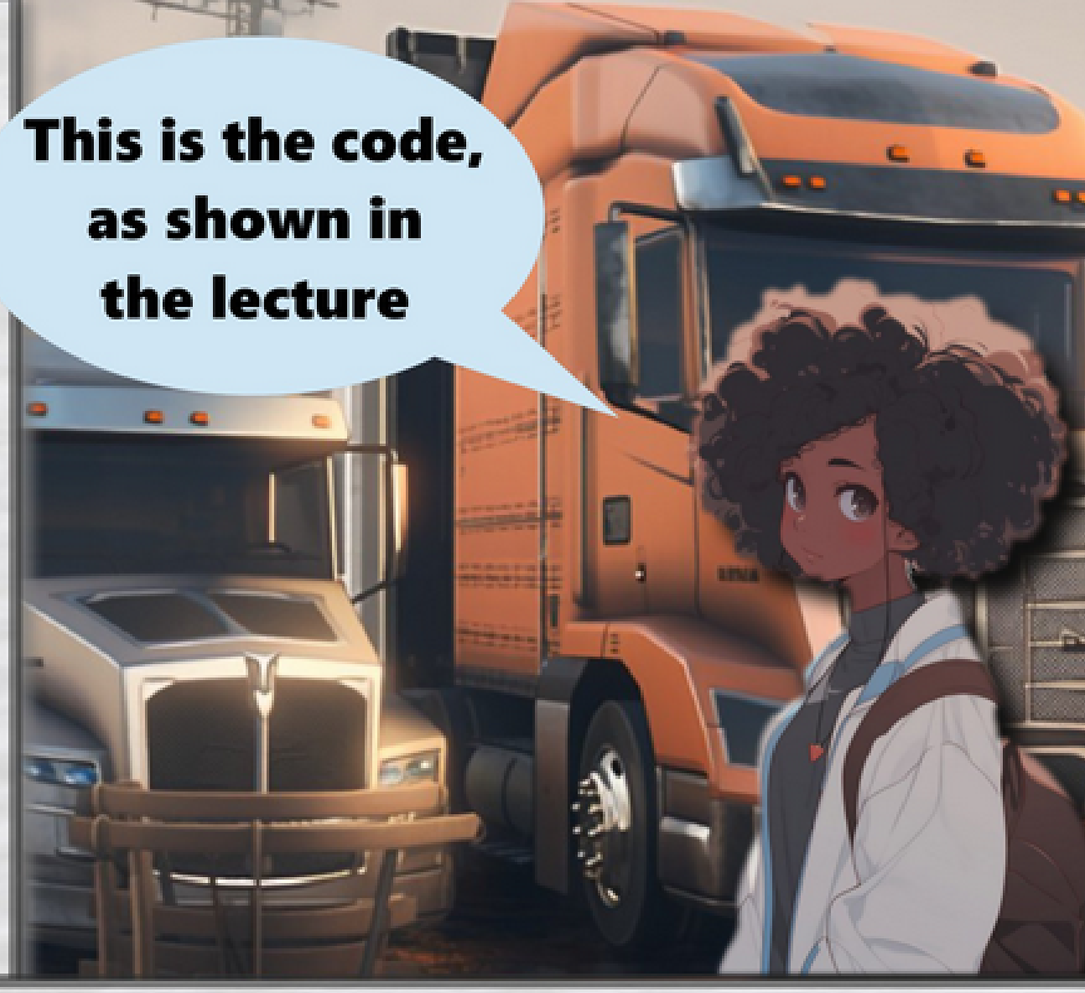
```
record Truck<T>(List<T> cargo){}
-------------------------------
interface Cargo<T>{ T cargo(); }
record Truck<T>(
  Cargo<List<T>> cargo){}
-------------------------------
class Cargo<T>{ T cargo; … }
class Truck<T>
  extends Cargo<List<T>>{}
```




Hey, boys,
I'm Sophia.
You are from our
class, right?
You are talking "generics".
Can we join you?
More students are
joining the discussion

```
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;
  }
}
```



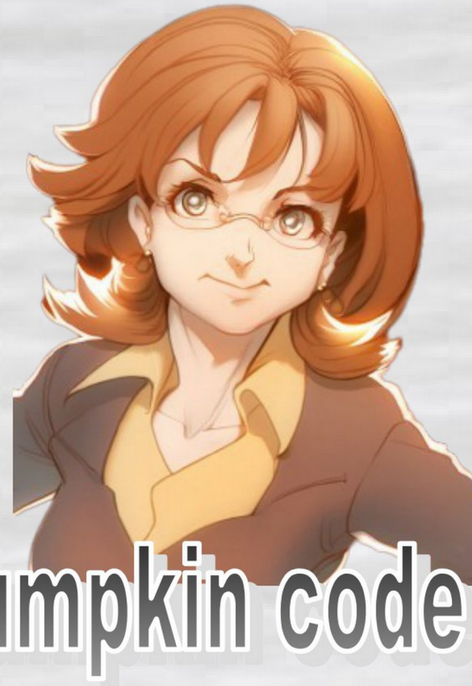

```
class Bonneted<T> extends Truck<T>{
  Bonneted(Point p, Cargo<List<T>> c){
    super(p,c);
  }
  public int aerodynamics(){ ... }
}
```

```
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);
  }
}
```

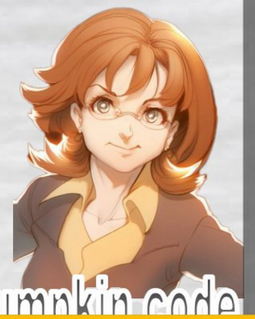
Again, we show another way to solve it




```
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(){ ... }
}
```

```java
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

```java
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

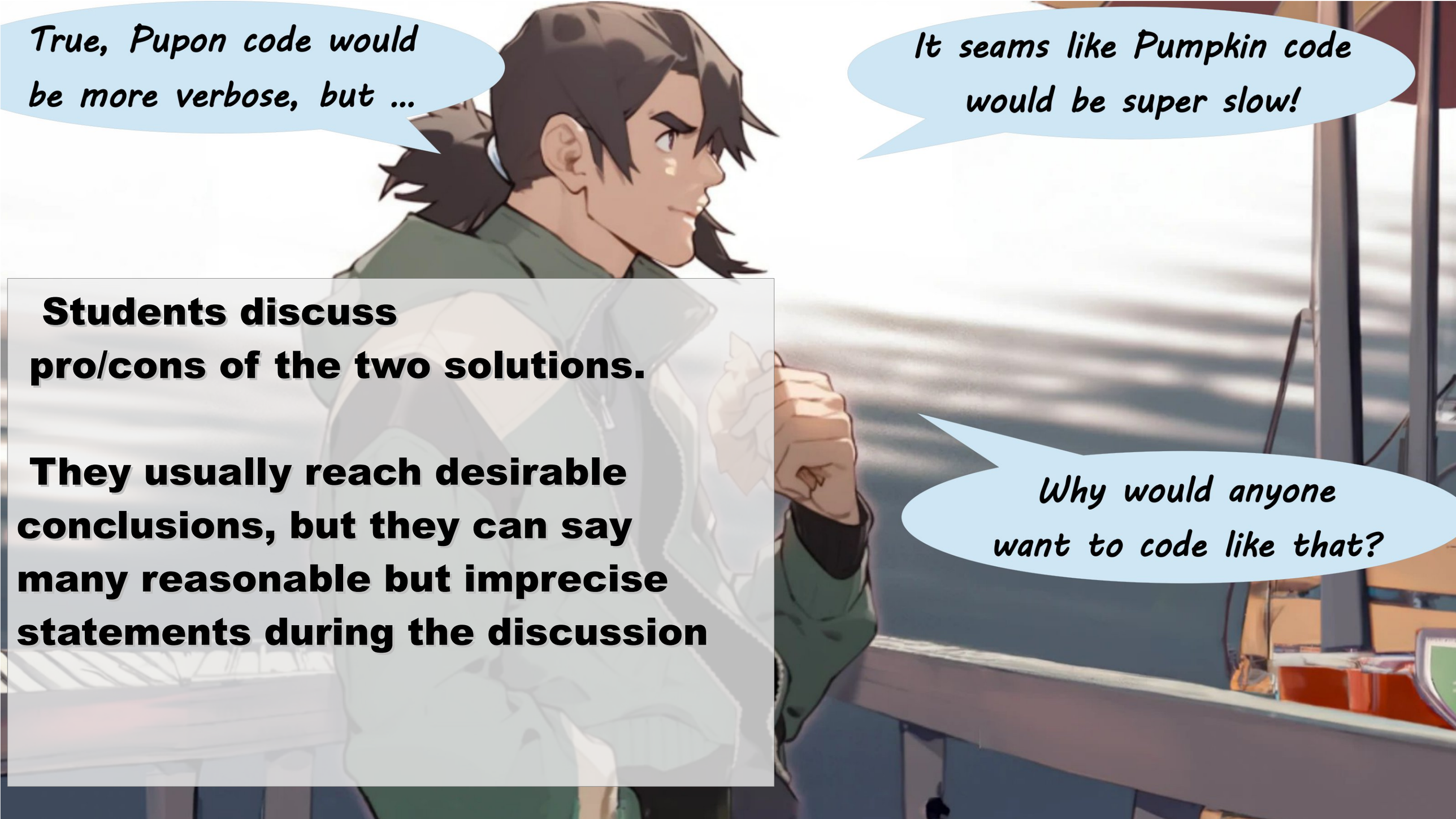
```java
record Bonne
  (Point loc
  public Bon
  public int
}
record CabOv
 (Point loca
  public Cab
  public int
}
```

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!



Students discuss pro/cons of the two solutions.

They usually reach desirable conclusions, but they can say many reasonable but imprecise statements during the discussion


Wildcards? Give me a break!
Truck<?, Pepsi >
Students can get angry
and have intense discussions
With the wildcard, the
method ‘driveToward’
would just return an Object.
Truck<?, Pepsi > truck = ...;
Object pointless = truck.driveToward(..);
//nothing else would compile


Just what I would expect from UPU
A first year student writing perfectly working code...
at the first attempt, does not even meet the minimum standards!
I can represent stressful situations, showing failure.


My time at UPU ends today
This can actually help Students in difficulty to feel less alone.

Questions and Quizzes

Both in my comicbook and in my lectures,
I insert questions and discuss answers

```
.map(s -> s.name())
.get()
return
.max(
.forE
.find
.redu
.filt
.orElseThrow(()->new Error("Empty student list provided"))
```

```
record Student(String name, double height){}

String tallestStudent(List<Student> ss){
```

Pause the video. Can you solve this one?

Note: not all snippets are needed.

Yes, you can not drag and drop code on a paused youtube video, but you can imagine doing it.
Similar visualization exercises have been shown to be very effective at improving programming.

This message will disappear shortly; you can pause after that.

```
.map(s -> s.name())
```

```
.get()
```

```
return ss.stream()
```

```
record Student(String name, double height){}

String tallestStudent(List<Student> ss){
  ????;
}
```

```
.max(Comparator.comparingDouble(s -> s.height()))
```

```
.forEach(s -> System.out.println(s.name()))
```

```
.findFirst()
```

```
.reduce((s1, s2) -> s1.height() < s2.height() ? s1 : s2)
```

```
.filter(s -> s.height() > 6)
```

```
.orElseThrow(()->new Error("Empty student list provided"))
```

```
.toList();
```

```
record Student(String name, int grade){}
```

```
.findFirst();
```

```
List<String> best(List<Student> ss){
```

```
return
```

```
.map(
```

```
.filt
```

```
int i
```

```
Optio
```

```
.filt
```

```
.sorted(Comparator.comparingInt(s -> -s.grade()))
```

Pause the video. Can you solve this one?

Note: not all snippets are needed.

The method returns the list of the student names
with grades of 80 or more.
Their grade must also be good enough that it could place
them in the best 10% of the class.
The class is composed of all the students in the list.

This message will disappear shortly; you can pause after that.

```
.toList();
```

```
.findFirst();
```

```
return ss.stream()
```

```
.map(s -> s.name())
```

```
.filter(s -> s.grade()>=80)
```

```
int indexAt10pc = (ss.size()-1)/10;
```

```
Optional<Student> threshold = ss.stream()
```

```
.filter(s -> s.grade()>=threshold.get().grade())
```

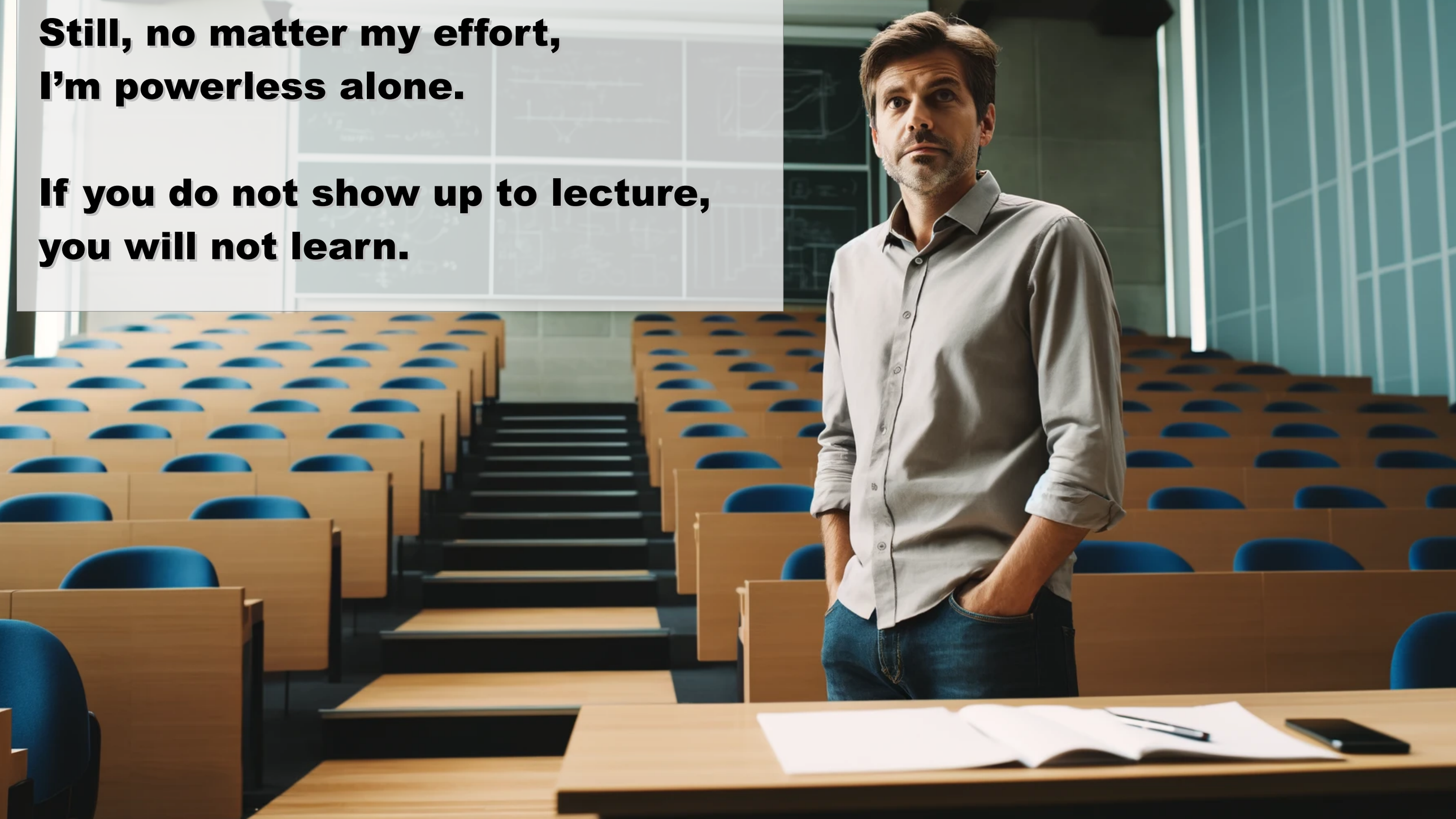
```
.sorted(Comparator.comparingInt(s -> -s.grade()))
```

```
record Student(String name, int grade){}

List<String> best(List<Student> ss){
   ????
}
```

```
.limit(indexAt10pc)
```

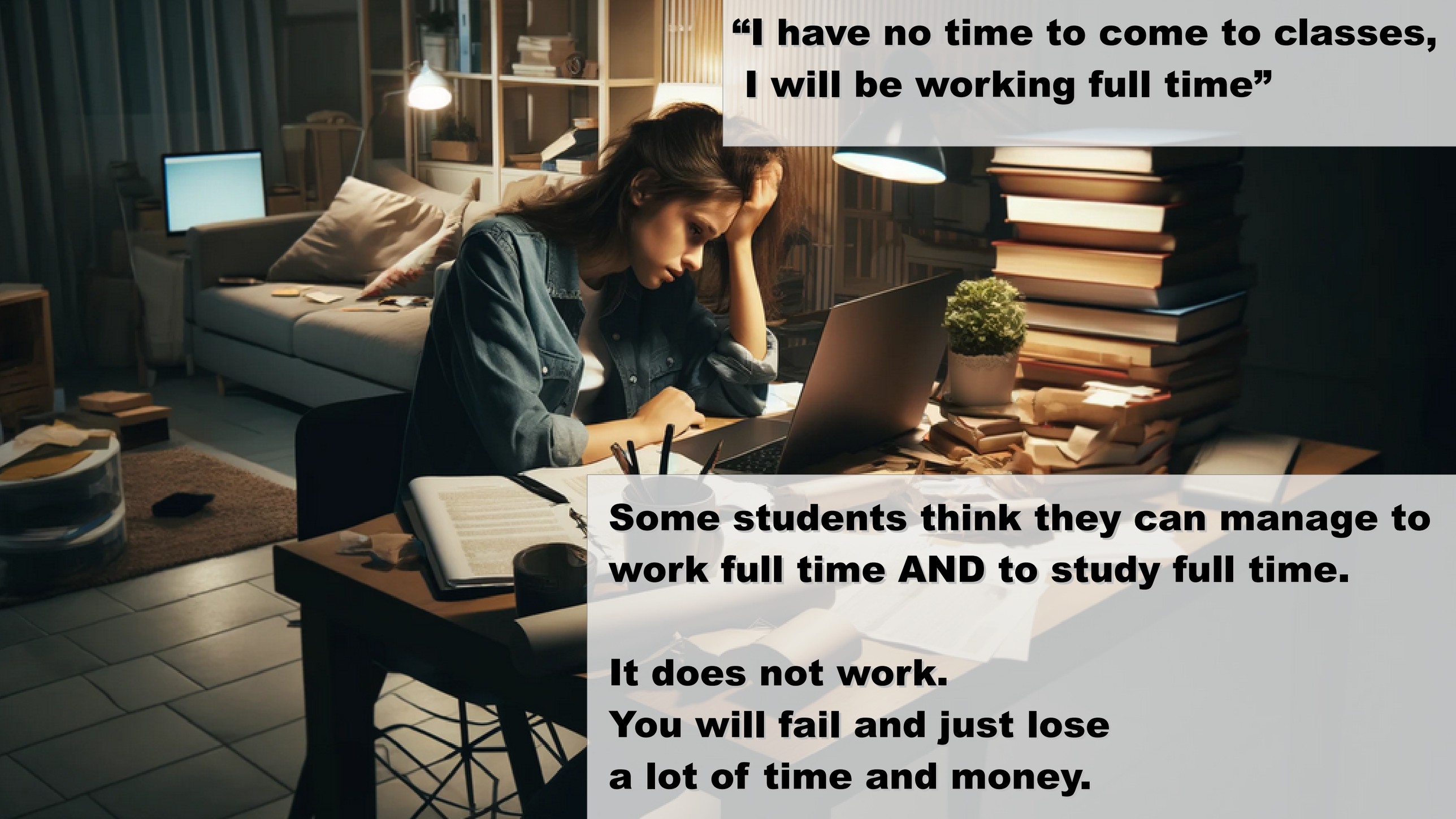
```
.skip(indexAt10pc)
```


Still, no matter my effort,
I’m powerless alone.
If you do not show up to lecture,
you will not learn.

We are evolved animals.
Paying attention and learning consumes a lot of brain juice, thus our instinct is to avoid it.
But we are also
tribal social animals

In the class we can feel the whole tribe trying to pay attention.
Our tribal instincts awaken and we are actually able to pay attention to difficult content, up to a level impossible at home watching a recording.

Yes, the classroom serves a real role. It is needed to trick our primitive brains into focus. To convince it that programming is good for its own sake.



“I have no time to come to classes, I will be working full time”

Some students think they can manage to work full time AND to study full time.

It does not work.
You will fail and just lose
a lot of time and money.