String.dedent

For Stage 2

Play around with REPL: bit.ly/3tiuGkV

In the interest of time…

This presentation is essentially the same as last meeting.

- I removed Syntax discussion.
- I removed Indentable follow-on discussion.

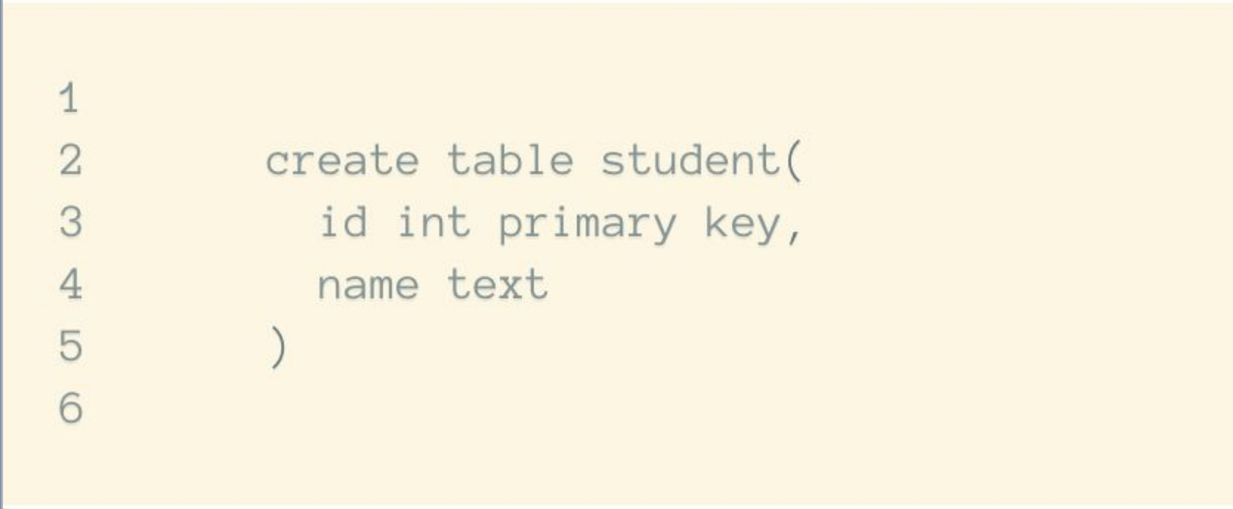
If the committee doesn’t need me to explain the proposal again, I could skip to asking for Stage 2.

A note on dots and dashes

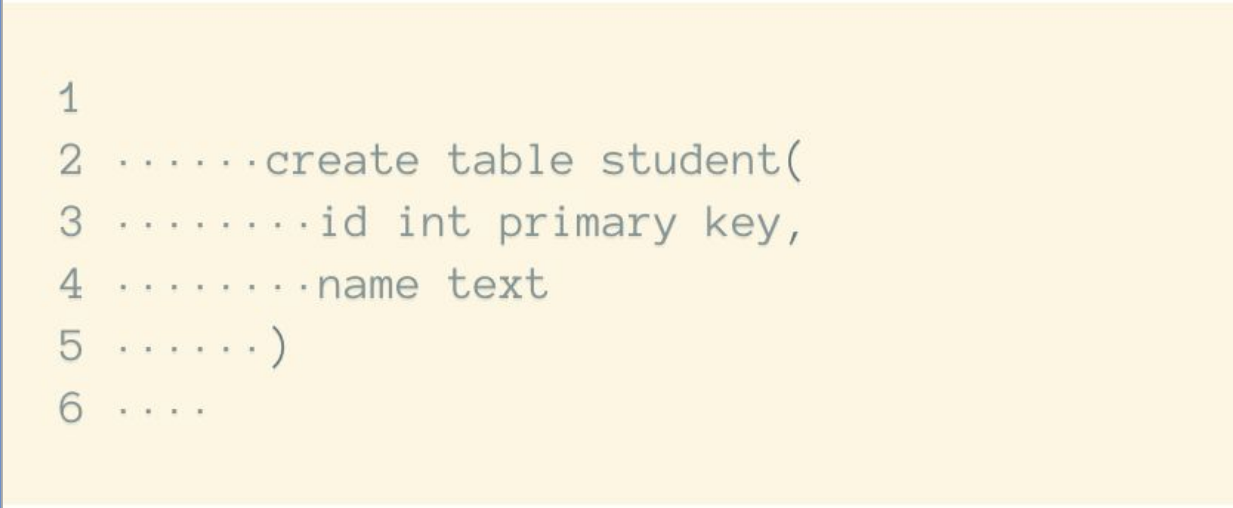
It's really hard to talk about whitespace if you can't see it. When necessary for clarity:

- Every line is significant (maybe empty, maybe all whitespace, maybe chars)
- A dot · represents a space
- A dash – represents a tab

These two outputs represent the same text.



```

      create table student(
        id int primary key,
        name text
      )

```



```

······create table student(
········id int primary key,
········name text
······)
····
```

Problem

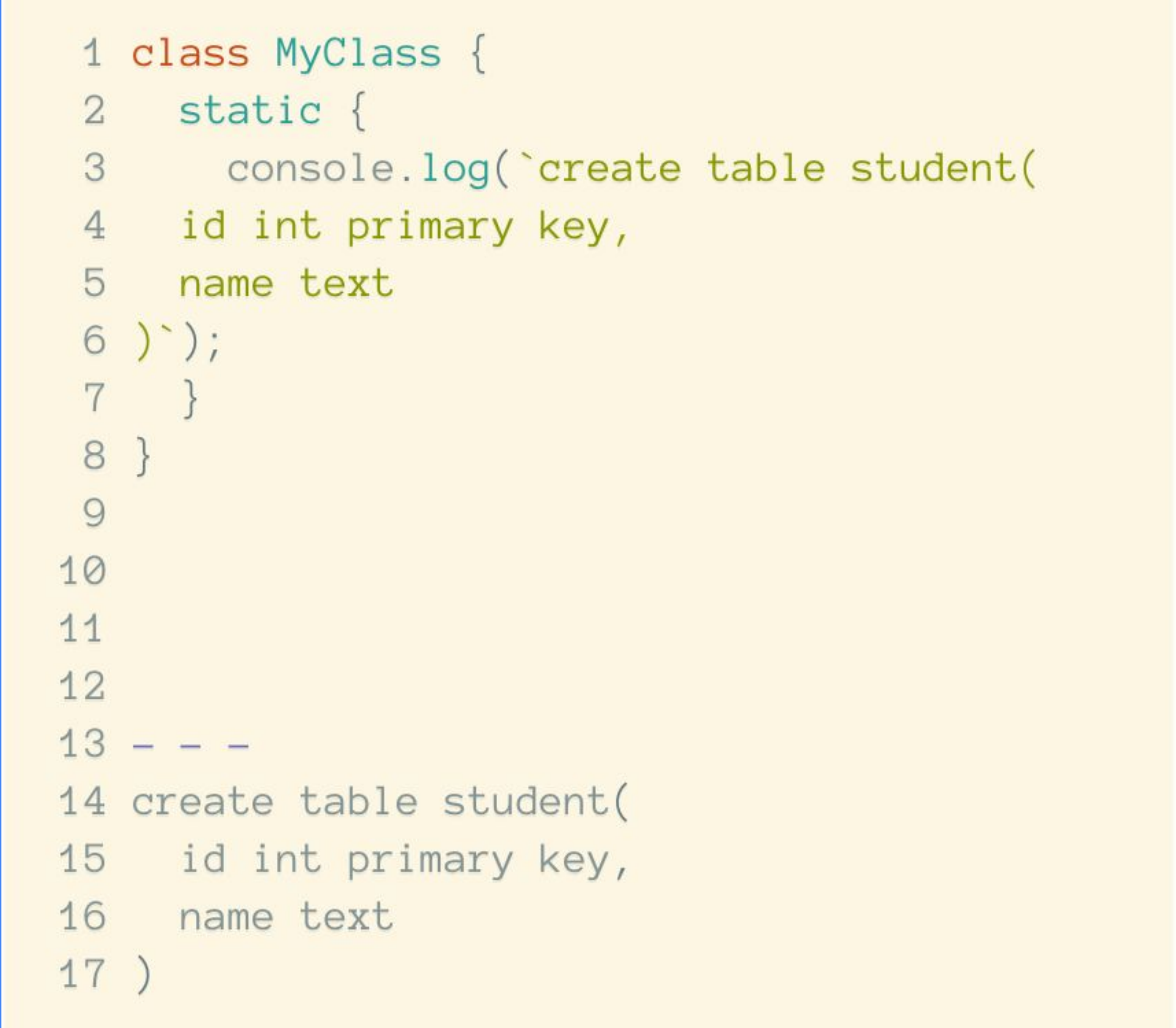
Multiline strings are extremely common in messages…

The tradeoff between code formatting and output formatting leaves a lot to be desired.

```
class MyClass {
  static {
    console.log(`
      create table student(
        id int primary key,
        name text
      )
    `);
  }
}
- - -

······create table student(
········id int primary key,
········name text
······)
····
```

Problem

We can optimize for the output format, but oh it hurts.



```
class MyClass {
  static {
    console.log(`create table student(
  id int primary key,
  name text
)`);
  }
}




- - -
create table student(
  id int primary key,
  name text
)
```

String.dedent

Gives us pleasant code formatting *and* output format!

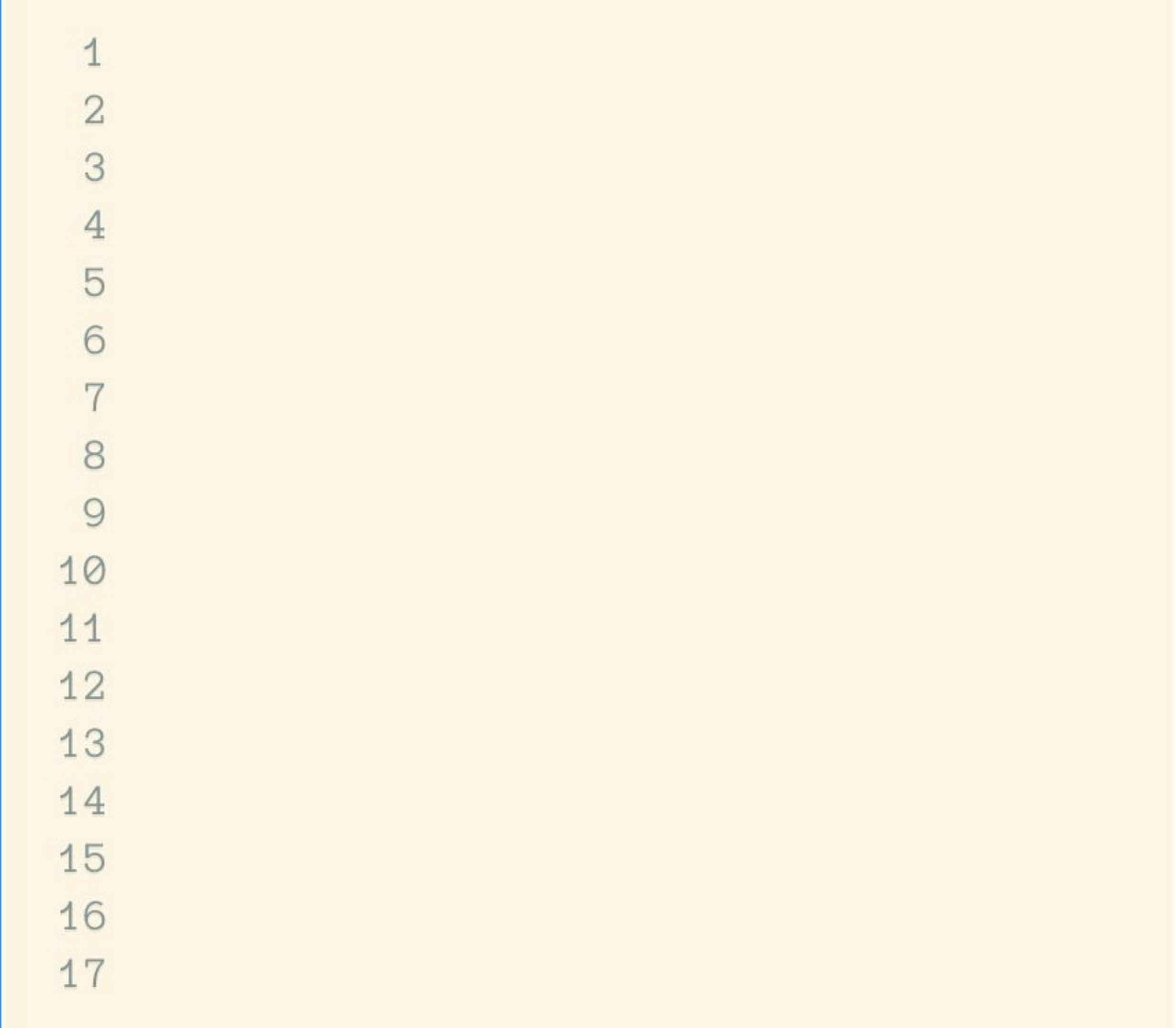
```
class MyClass {
  static {
    console.log(String.dedent`
      create table student(
        id int primary key,
        name text
      )
    `);
  }
}


- - -
create table student(
  id int primary key,
  name text
)
```

Terminology

"Newlines" are JS syntax newlines

- `\n`
- `\r\n?`
- `U+2028`
- `U+2029`

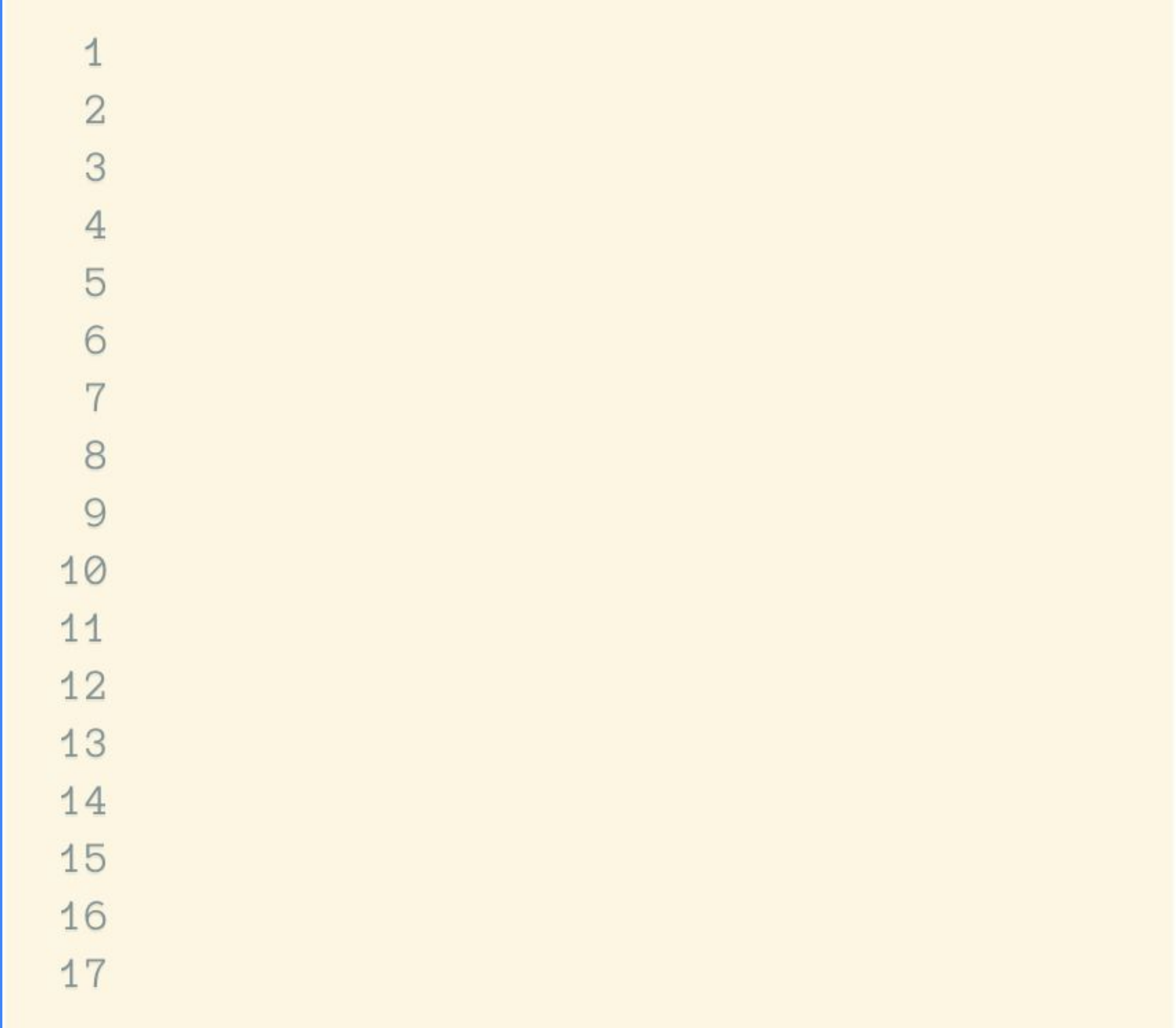
You can't see them, but there are a bunch of literal newline chars over there…



Terminology

"Whitespace" is all Unicode whitespace, minus the newlines

You can't see them, but there are a bunch of whitespace chars over there…



Terminology

The "opening line" is everything after the opening backtick.

The opening line must be empty (ie, backtick is immediately followed by literal newline char).

The opening line's newline is removed from the output.

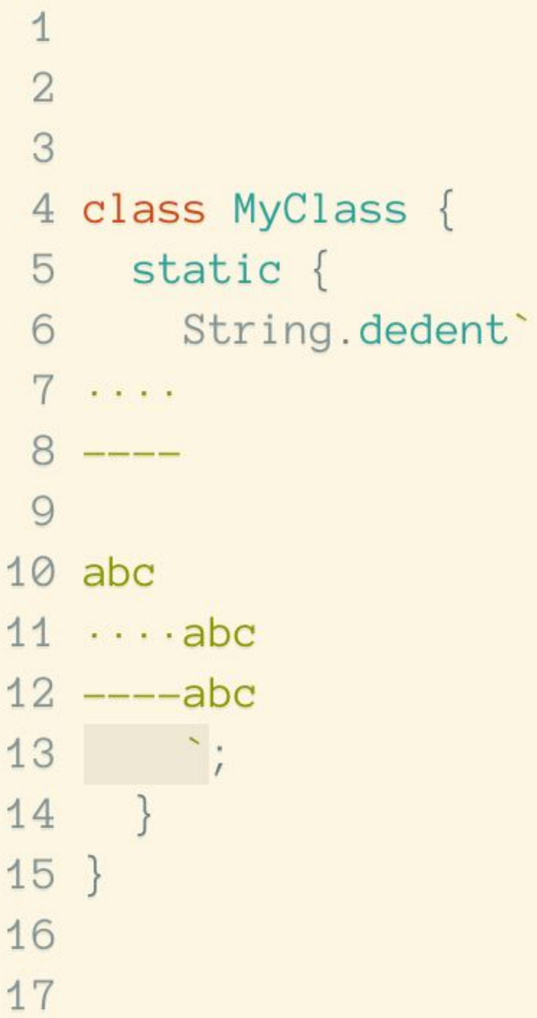
```
class MyClass {
  static {
    String.dedent`
····
----

abc
····abc
----abc
    `;
  }
}
```

Terminology

The "closing line" is everything before the closing backtick.

The closing line may optionally contain leading whitespace. It must not contain non-whitespace.

The closing line's preceding newline and any whitespace are removed from the output.

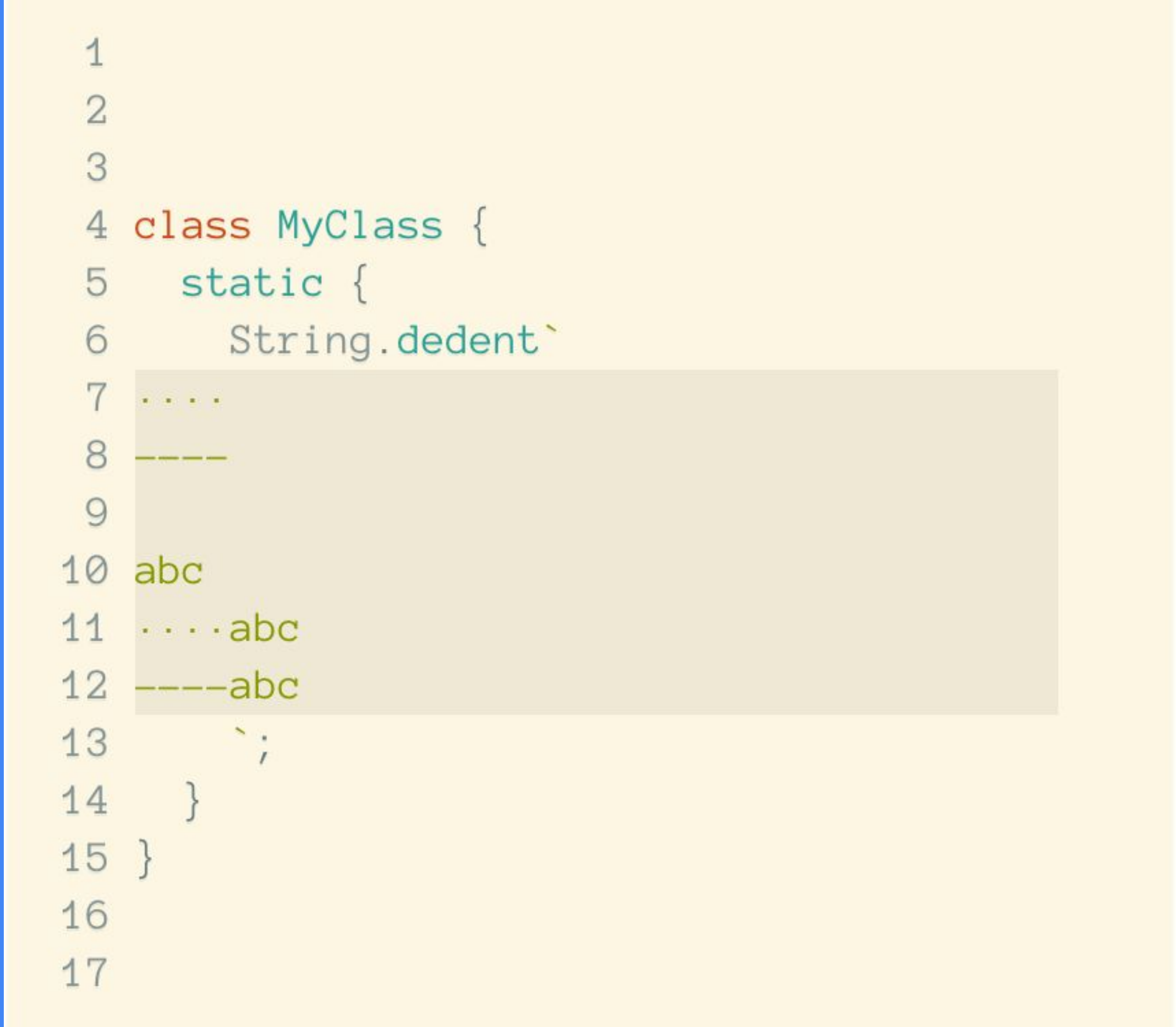


```
class MyClass {
  static {
    String.dedent`
····
----

abc
····abc
----abc
    `;
  }
}
```

Terminology

"Content lines" are everything in between.

There may be 0 content lines, or as many as you want.

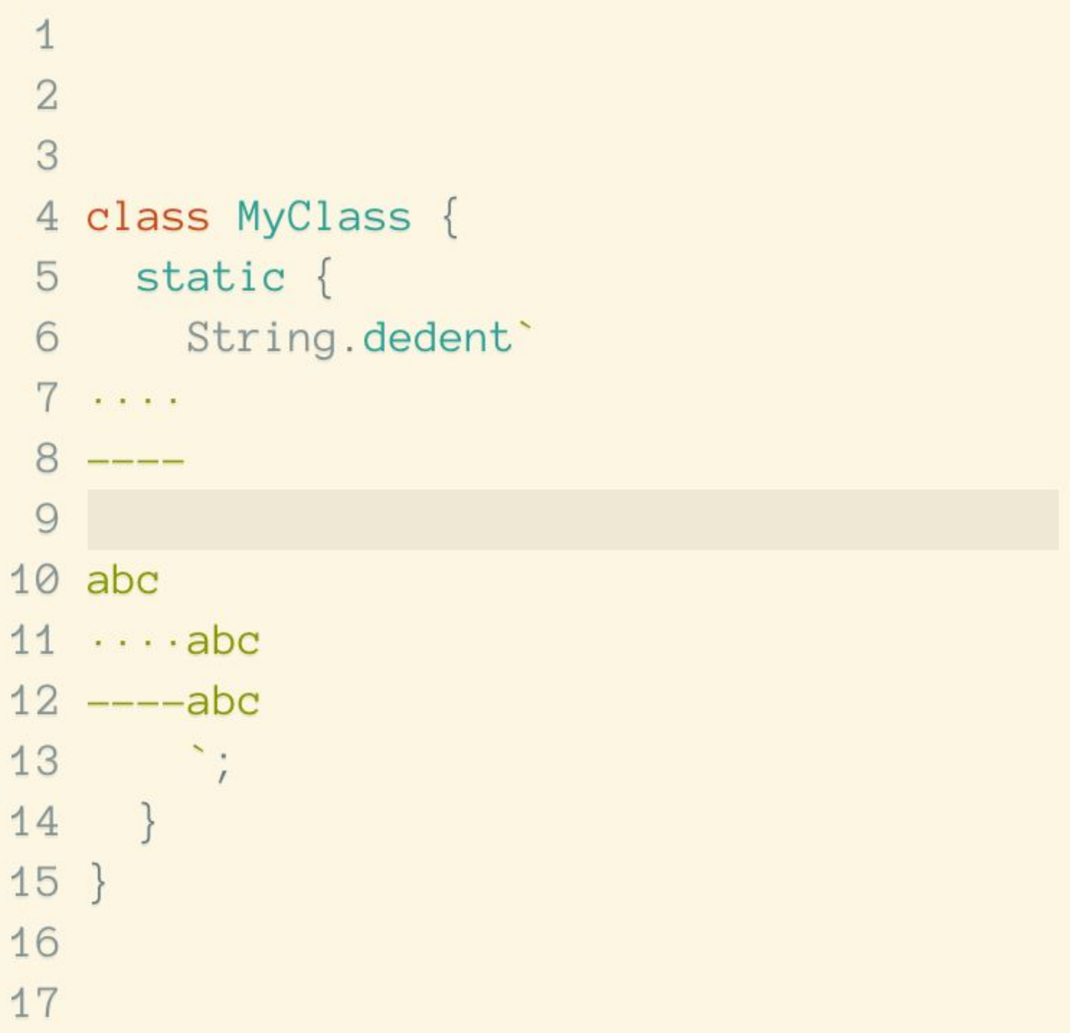


```
class MyClass {
  static {
    String.dedent`
····
----

abc
····abc
----abc
    `;
  }
}
```

Terminology

"Empty lines" are content lines that contain neither whitespace nor chars.

(Remember dots and dashes represent spaces and tabs respectively)



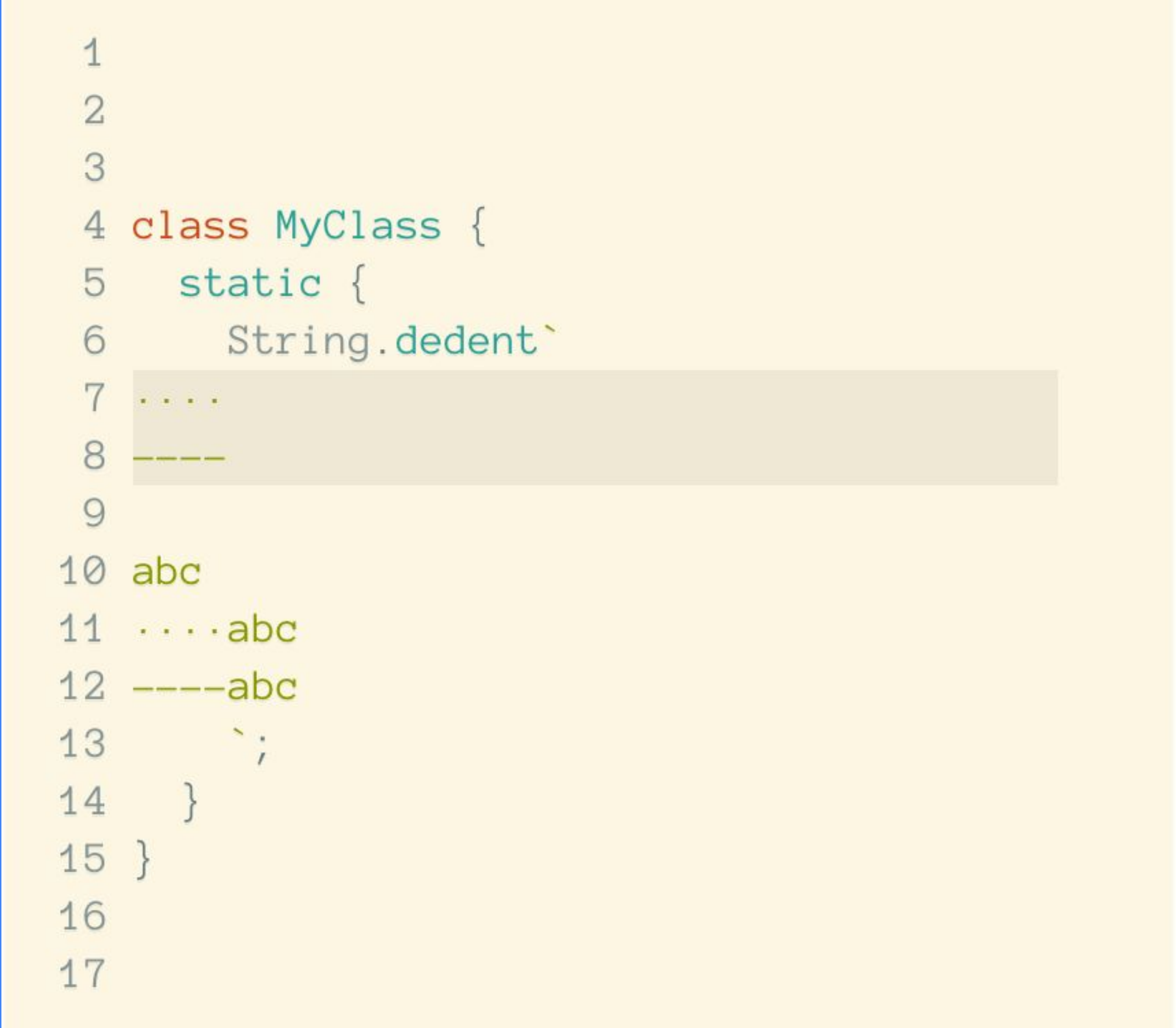
```



class MyClass {
  static {
    String.dedent`
····
----

abc
····abc
----abc
    `;
  }
}


```

Terminology

"Whitespace lines" are content lines that contain only whitespace.

Any Unicode whitespace is "whitespace", minus the newline chars.

(Remember dots and dashes represent spaces and tabs respectively)



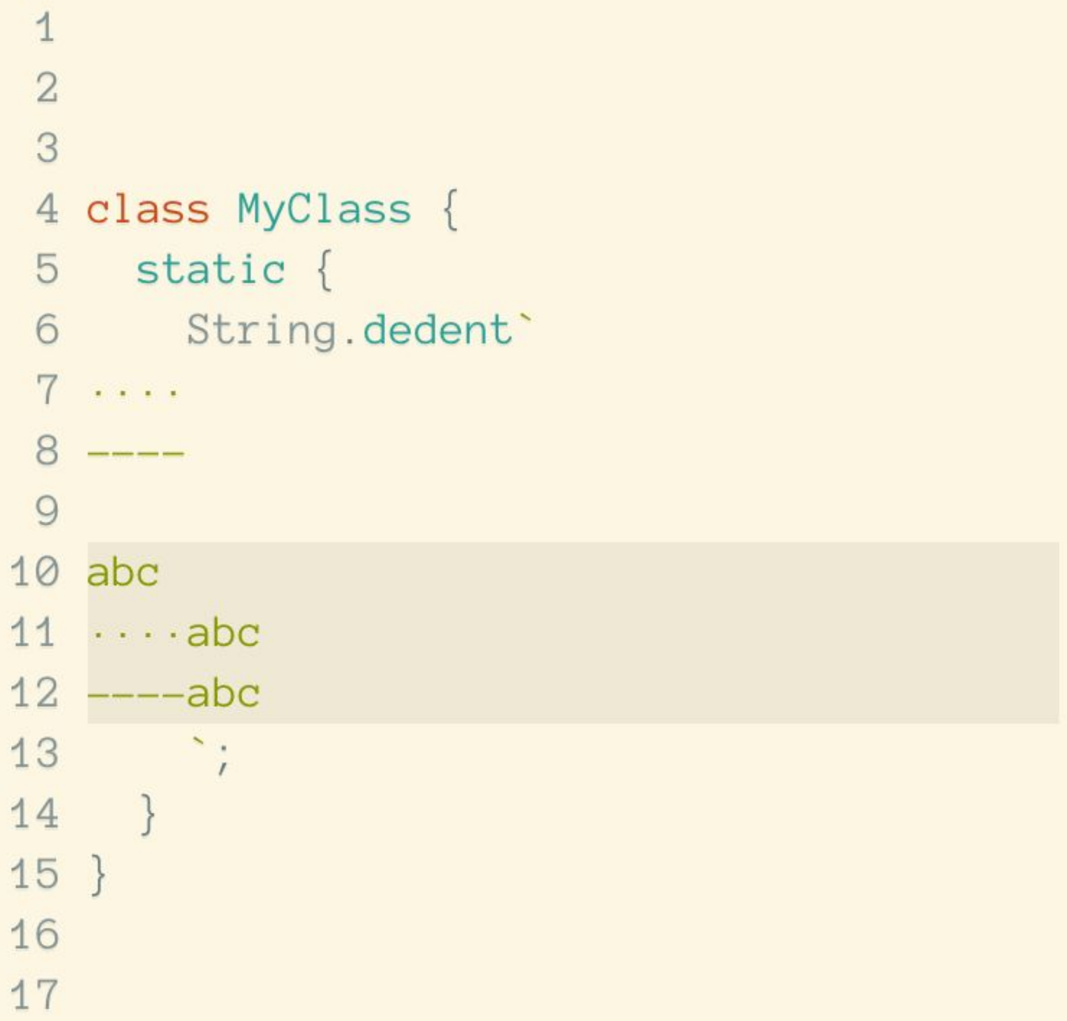
```
class MyClass {
  static {
    String.dedent`
····
----

abc
····abc
----abc
    `;
  }
}
```

Terminology

"Char lines" are content lines that contain a char.

Also lines with an expression, we'll get to that.

They may optionally start with whitespace.

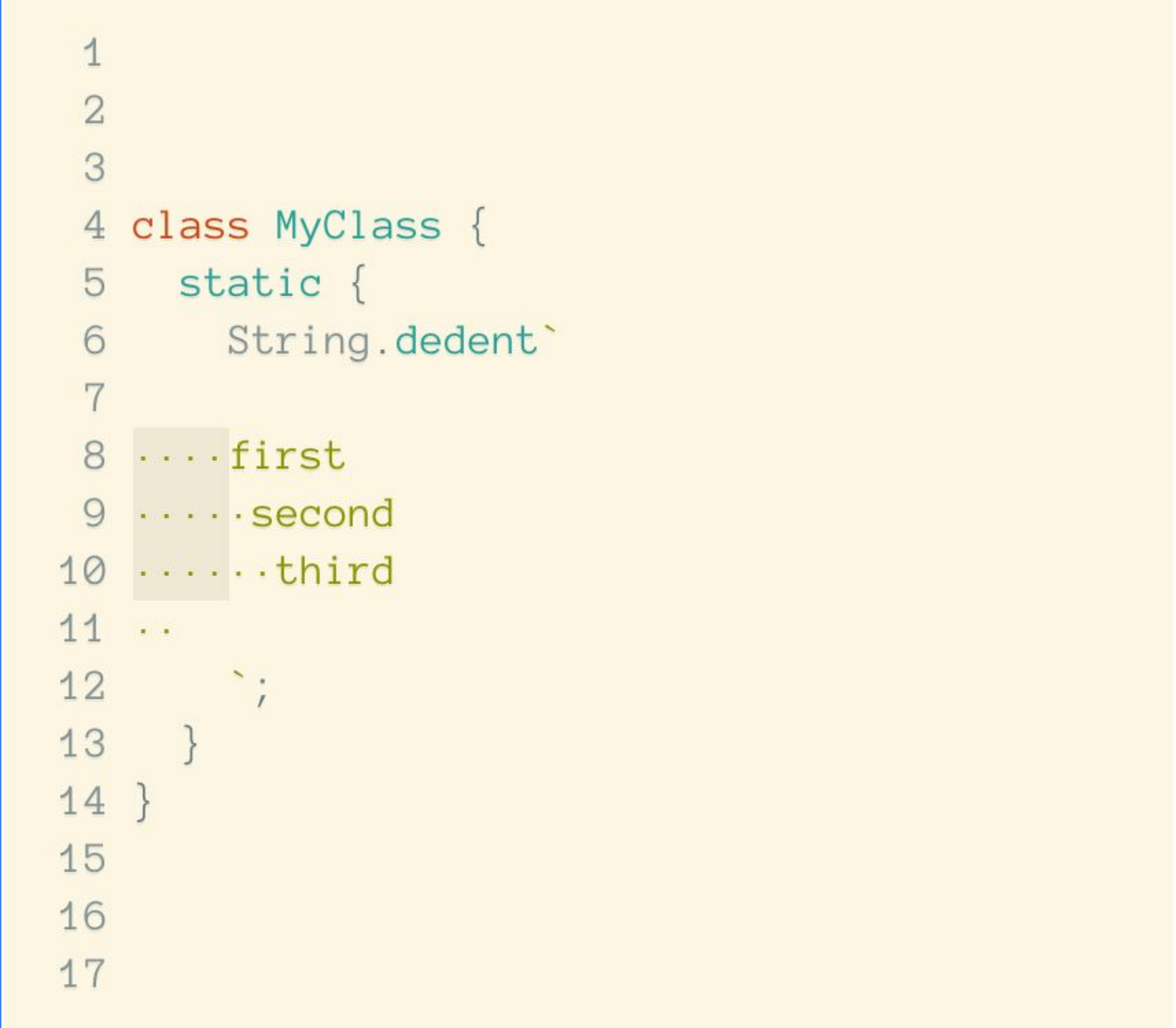


```



class MyClass {
  static {
    String.dedent`
····
----

abc
····abc
----abc
    `;
  }
}


```

Terminology

"Common indentation" is the maximum whitespace that matches exactly on every char line's leading whitespace indentation.

Empty lines don't affect common indentation. Neither do whitespace lines.



```



class MyClass {
  static {
    String.dedent`

····first
·····second
······third
··
    `;
  }
}



```

Common Indentation

A char line without any leading whitespace makes the common indentation 0.

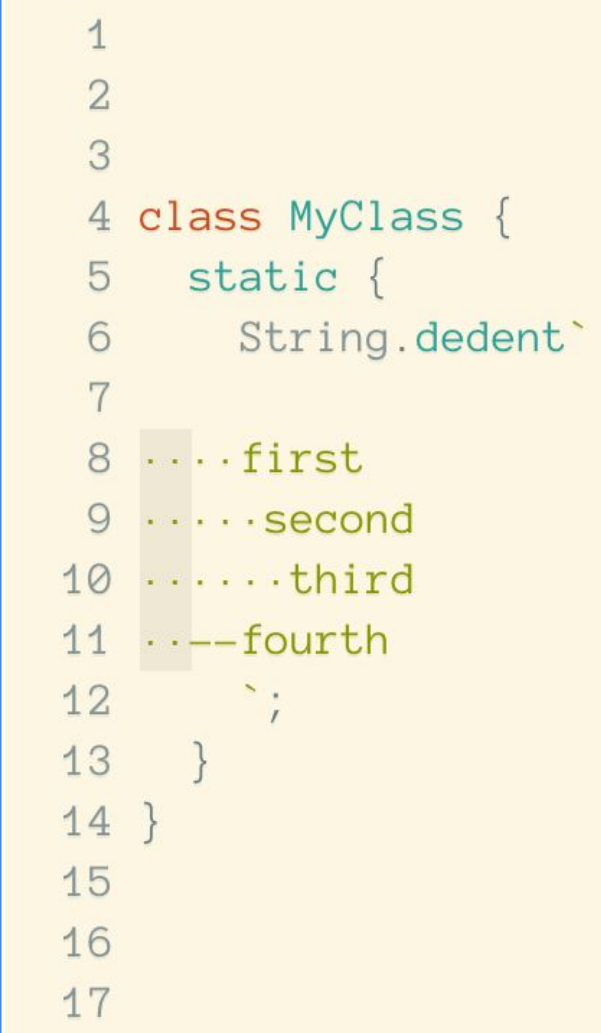
This output will not have any indentation removed.

```



class MyClass {
  static {
    String.dedent`

····first
·····second
······third
fourth
    `;
  }
}



```

Common Indentation

A char line with mismatching leading whitespace shortens the common indentation.

Tabs and spaces aren't interchangeable.

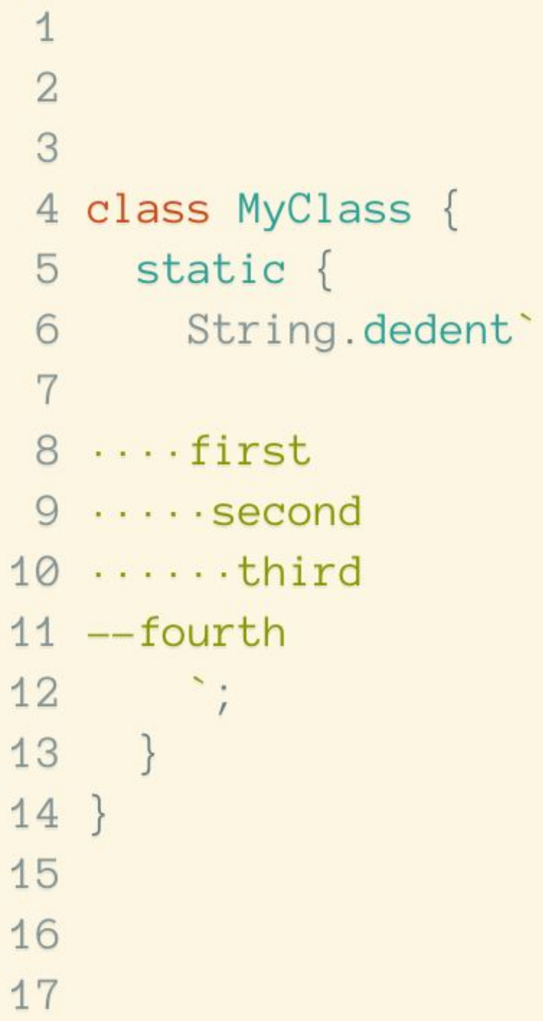


```
class MyClass {
  static {
    String.dedent`

····first
·····second
······third
··——fourth
    `;
  }
}
```

Common Indentation

A mismatched leading whitespace at column 1 makes the common indentation 0.

Tabs and spaces really aren't interchangeable.



```



class MyClass {
  static {
    String.dedent`

····first
·····second
······third
--fourth
    `;
  }
}



```

Common Indentation

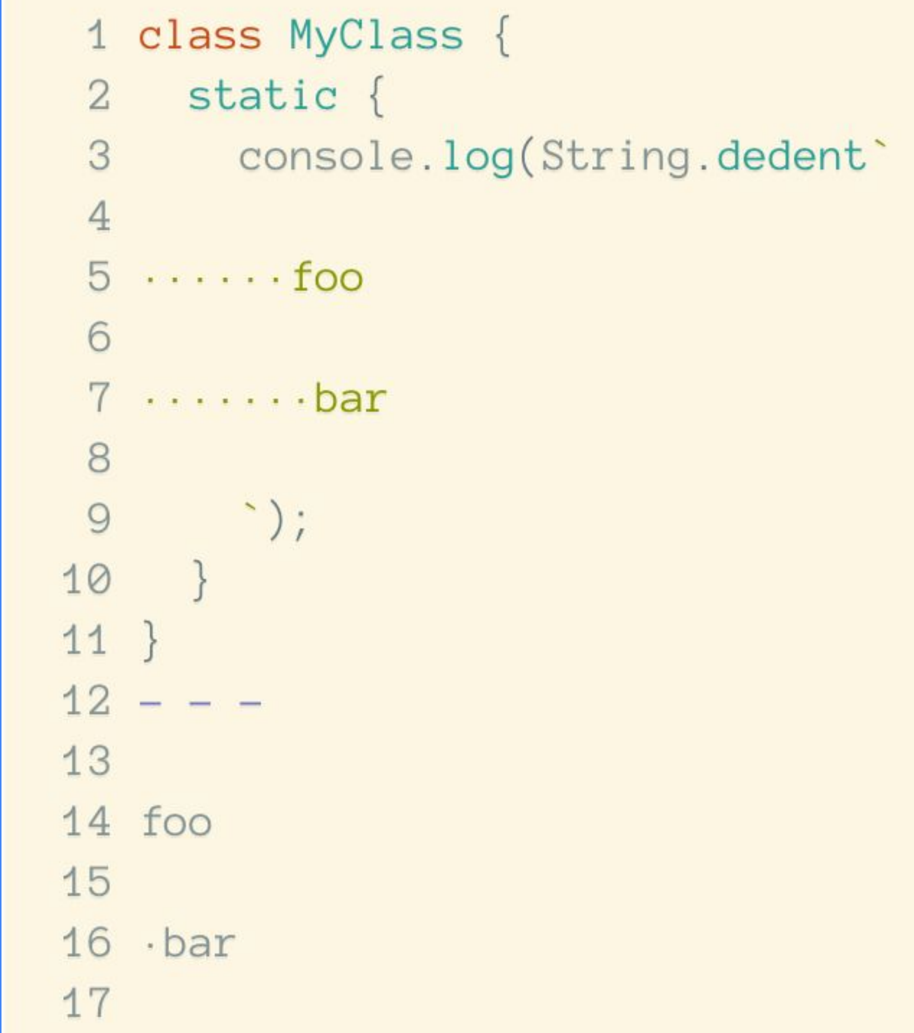
Whitespace can be freely mixed, common indentation will take the maximum leading whitespace that matches.

```



class MyClass {
  static {
    String.dedent`

·–·–first
.–.–.second
.–.–..third
–.
    `;
  }
}



```

Empty lines

Empty lines do not affect common indentation. They're preserved in output.

Leading and trailing empty content lines are preserved, too.



```
class MyClass {
  static {
    console.log(String.dedent`

······foo

·······bar

    `);
  }
}
- - -

foo

·bar

```

Whitespace lines

Whitespace do not affect the common indentation.

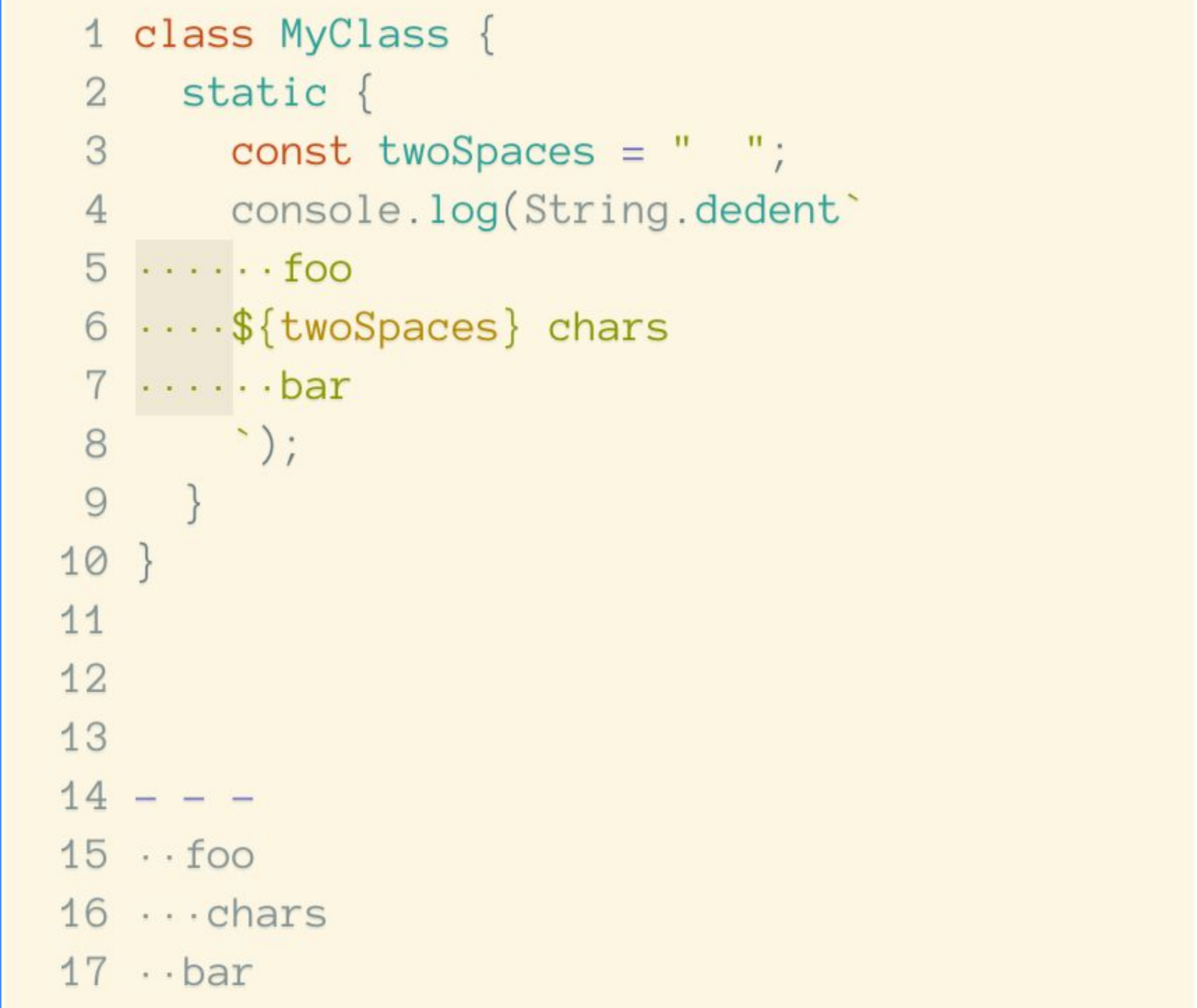
The easiest thing to do is empty them out in the output. Python's dedent sets precedent.

```
class MyClass {
  static {
    console.log(String.dedent`
·····foo
······························
······bar
    `);
  }
}




- - -
foo

·bar
```

Expressions

Expressions *do* affect common indentation. They shorten the common indentation at their starting `${`.

The expression's evaluation is not considered part of the template's whitespace.



```
class MyClass {
  static {
    const twoSpaces = "  ";
    console.log(String.dedent`
······foo
····${twoSpaces} chars
······bar
    `);
  }
}



- - -
··foo
···chars
··bar
```

Tagged Dedent

String.dedent can be composed with another template tag.

Both the `cooked` and `raw` arrays are dedented, the resulting `TemplateStringsArray` is cached, and it's passed to the composed template tag.

```

class MyClass {
  static {
    // IndentationError in Python
    pythonInterpreter`
      print('Hello Python World')
    `;
    // Works
    pythonInterpreter`
print('Hello Python World')
`;
    // Works
    String.dedent(pythonInterpreter)`
      print('Hello Python World')
    `;
  }
}
```

Discussion?

Play around with REPL: bit.ly/3tiuGkV

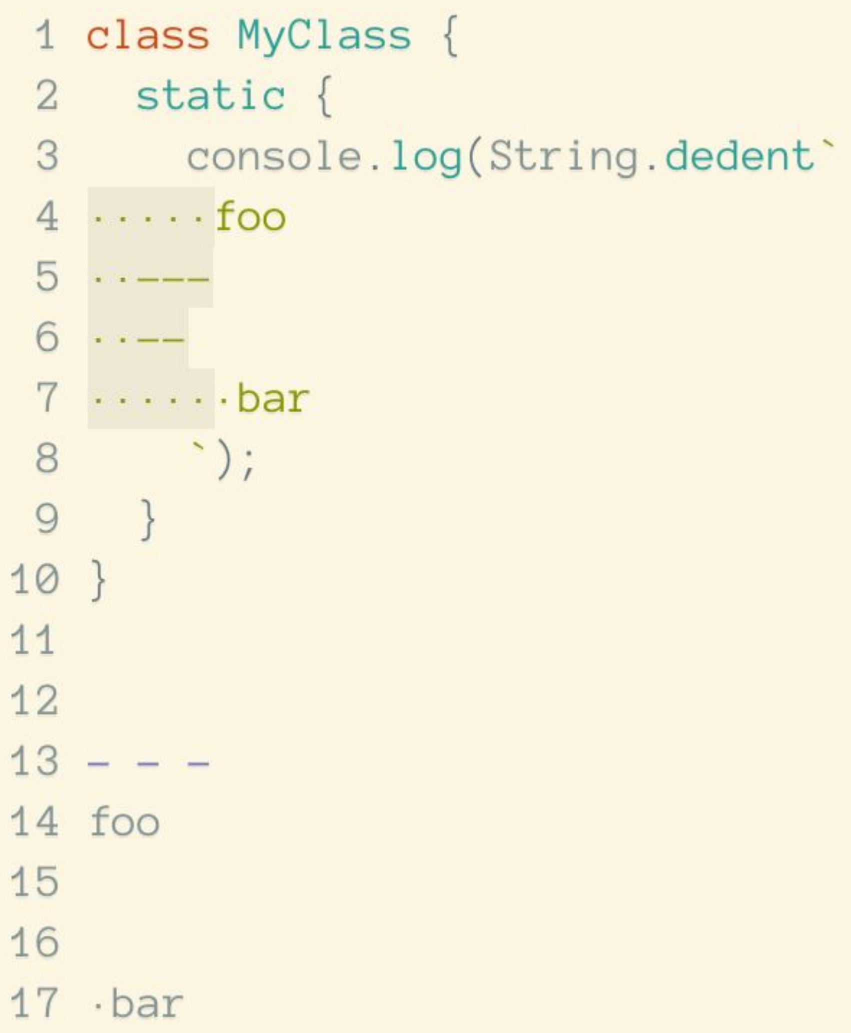
Stage 2?

What if… Whitespace lines

What if whitespace lines didn't participate, but the common indentation was still removed?

This violates the "spaces and tabs don't mix" rule. It also violates shared number of chars.

I hate it.

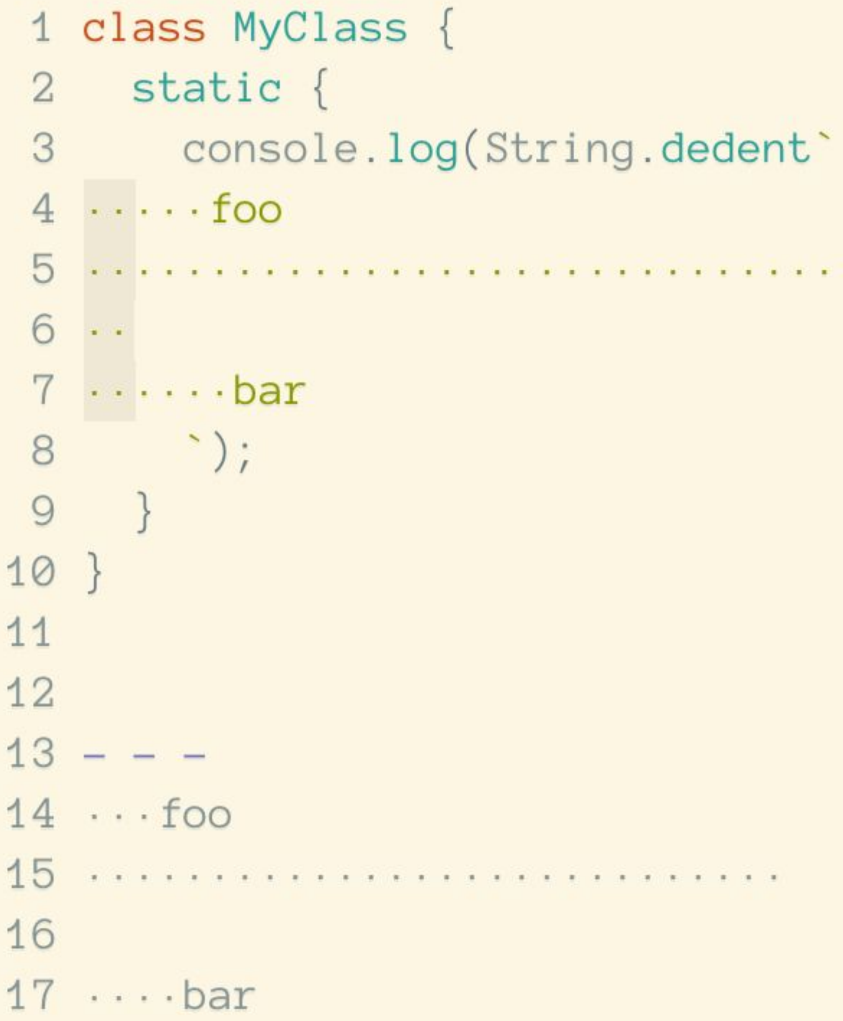


```
class MyClass {
  static {
    console.log(String.dedent`
·····foo
··———
··——
······bar
    `);
  }
}


— — —
foo


·bar
```

What if… Whitespace lines

What if whitespace lines participated in the common indentation?

A whitespace only line could shorten the common indentation. It might be hard to spot, but I don't think it'd be terrible.



```
class MyClass {
  static {
    console.log(String.dedent`
·····foo
······························
··
······bar
    `);
  }
}


- - -
···foo
····························

····bar
```