Public field (similar to property, available on created object)

Private fields (not accessible **outside** of class)

Static public field (available **only** on **class**)

Call to parent (super) class (necessary with `extend`). Needs to happen before accessing `this`

Instance property (available on created object)

Redefining private field

Public method

Referencing private field and method

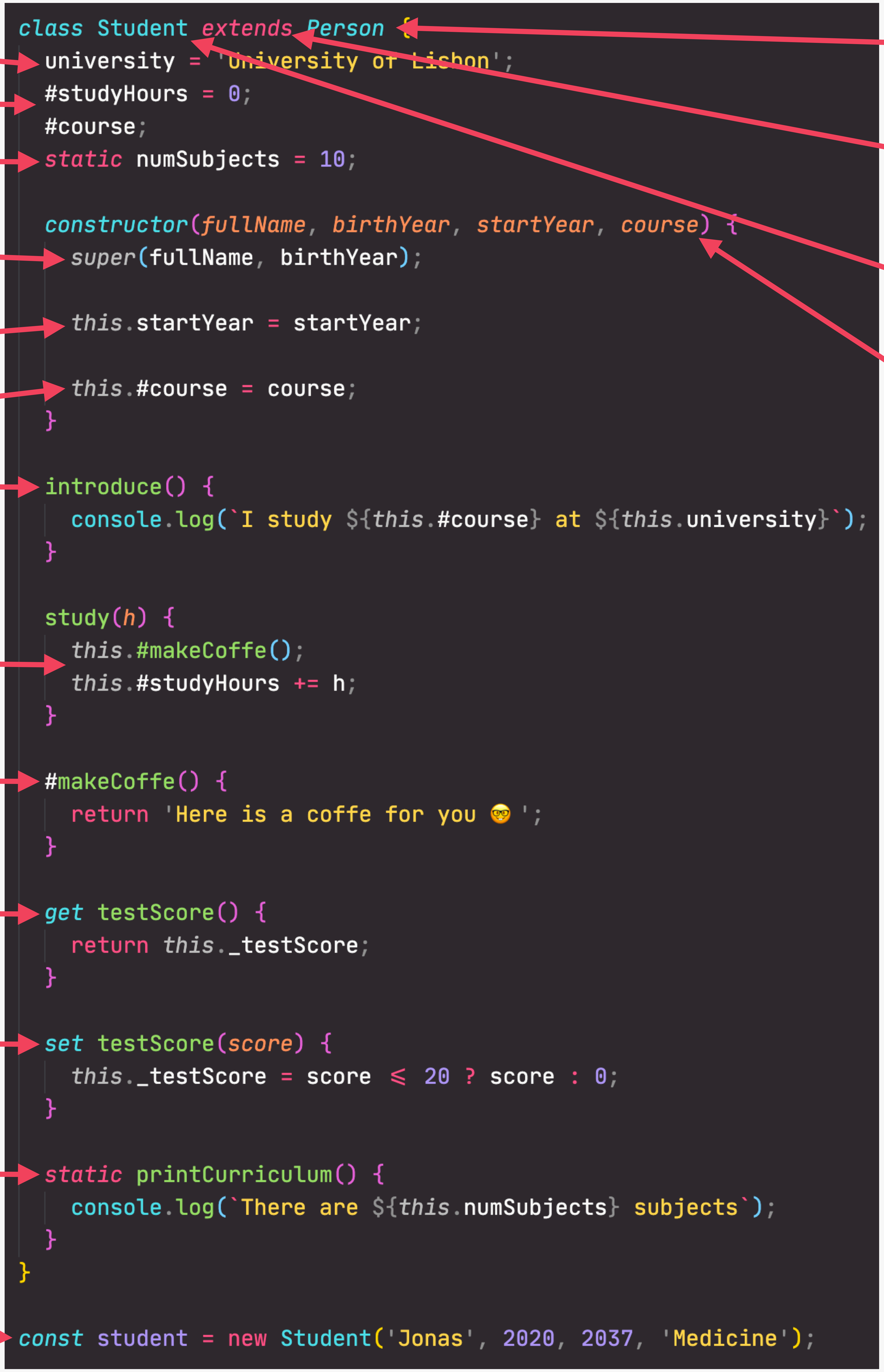
Private method (⚠️ Might not yet work in your browser. “Fake” alternative: _ instead of #)

Getter method

Setter method (use _ to set property with same name as method, and also add getter)

Static method (available **only** on **class**. Can **not** access instance properties nor methods, only static ones)

Creating new object with new operator



```
class Student extends Person {
  university = 'University of Lisbon';
  #studyHours = 0;
  #course;
  static numSubjects = 10;

  constructor(fullName, birthYear, startYear, course) {
    super(fullName, birthYear);

    this.startYear = startYear;

    this.#course = course;
  }

  introduce() {
    console.log(`I study ${this.#course} at ${this.university}`);
  }

  study(h) {
    this.#makeCoffe();
    this.#studyHours += h;
  }

  #makeCoffe() {
    return 'Here is a coffe for you 🤓';
  }

  get testScore() {
    return this._testScore;
  }

  set testScore(score) {
    this._testScore = score <= 20 ? score : 0;
  }

  static printCurriculum() {
    console.log(`There are ${this.numSubjects} subjects`);
  }
}

const student = new Student('Jonas', 2020, 2037, 'Medicine');
```

Parent class

Inheritance between classes, automatically sets prototype

Child class

Constructor method, called by new operator. Mandatory in regular class, might be omitted in a **child** class

- 👉 Classes are just “**syntactic sugar**” over constructor functions
- 👉 Classes are **not** hoisted
- 👉 Classes are **first-class** citizens
- 👉 Class body is always executed in **strict mode**