1. 클래스 개념
* 클래스(class) = 설계도. "어떤 속성(정보)과 동작(함수)을 가질지"를 미리 정해둔 틀.
* 인스턴스(instance) = 설계도로 찍어낸 실물. `new 클래스이름()`으로 만든 실제 객체.
- 클래스(설계도)로부터 인스턴스(실물 객체)를 `new`로 만든다.

```
<script>
        class Person {
          // 생성자
          constructor(name, age){
            this.name = name;
            this.age = age;
          }
          // 메서드
          sayHello(){
            console.log(`안녕하세요! 저는 ${this.name}, ${this.age}살입니다.`);
          }
        }
        // 인스턴스
        const p1 = new Person('홍길동', 20);
        p1.sayHello();
</script>
```

포인트

* `class` 키워드로 클래스 만들기
* `constructor` 안에서 `this.속성 = 값`으로 속성 저장
* 메서드는 `function` 키워드 없이 이름만 쓰면 됨
* `new Person(...)`로 인스턴스 생성

3. 실습 1

문제
`Item` 클래스를 만들고, 생성자에서 `title`, `price`를 받아 저장하세요.
메서드 `info()`는 `"제목 - ₩가격"`을 콘솔에 출력합니다.
인스턴스를 2개 만들어 `info()`를 각각 호출하세요.

정답 예시

```html
<script>
        class Item {
          constructor(title, price){
            this.title = title;
            this.price = price;
          }
          info(){
            console.log(`${this.title} - ₩${this.price.toLocaleString()}`);
          }
```

```
        }

        const i1 = new Item('노트북', 1290000);
        const i2 = new Item('마우스', 19000);
        i1.info(); // 노트북 - ₩1,290,000
        i2.info(); // 마우스 - ₩19,000
</script>
```

4. `this`

```
<script>
        class Counter {
          constructor(){
            this.value = 0; // 처음 값
          }
          plus(){ this.value = this.value + 1; }
          show(){ console.log(`현재값: ${this.value}`); }
        }

        const c = new Counter();
        c.plus();
        c.show(); // 현재값: 1
</script>
```

5. 실습 2 — 초간단 카운터 (DOM 사용 X)

`Counter` 클래스를 만들어 `value`를 0으로 시작하고,
`inc()`는 1 증가, `dec()`는 1 감소, `reset()`은 0으로.`print()`는 콘솔에 현재 값 출력
메서드들을 순서대로 호출해 동작을 확인

```
<script>
        class Counter {
          constructor(){ this.value = 0; }
          inc(){ this.value++; }
          dec(){ this.value--; }
          reset(){ this.value = 0; }
          print(){ console.log('값:', this.value); }
        }

        const ct = new Counter();
        ct.inc(); // 1
        ct.inc(); // 2
        ct.dec(); // 1
        ct.print(); // 값: 1
        ct.reset();
        ct.print(); // 값: 0
</script>
```

6. DOM 연결 기초
버튼 2개(증가/감소)와 숫자 한 줄을 화면에 보이게 하고, 클래스 메서드로 조작해본다.

```
<!DOCTYPE html>
<meta charset="utf-8" />
<title>DOM 카운터(쉬움)</title>
<style>
.box{ width:240px; margin:30px auto; padding:20px; border:1px solid #ddd; border-radius:10px; text-align:center; }
.num{ font:700 28px/1 system-ui; margin:10px 0; }
button{ padding:6px 10px; margin:0 6px; }
</style>
<div class="box">
  <div class="num" id="num">0</div>
  <button id="del">-1</button>
  <button id="add">+1</button>
</div>
<script>
class CounterView {
  constructor(view){
    this.value = 0;
    this.view = view;
    this.render();
  }
  addf(){ this.value++; this.render(); }
  delf(){ this.value--; this.render(); }
  render(){ this.view.textContent = this.value; }
}

const cv = new CounterView(document.getElementById('num'));
document.getElementById('add').onclick = () => cv.addf();
document.getElementById('del').onclick = () => cv.delf();
</script>
```

문제 1
`Card` 클래스를 만들고 `name`, `job`을 저장.
`text()` 메서드는 `"이름(직업)"` 문자열을 반환.
인스턴스 2개를 만들어 콘솔에 출력하세요.

** 예시 **
```
<script>
class Card {
  constructor(name, job){
    this.name = name;
    this.job = job;
  }
  text(){
    return `${this.name}(${this.job})`;
  }
```

```
}

const a = new Card('Kim', 'Developer');
const b = new Card('Lee', 'Designer');
console.log(a.text()); // Kim(Developer)
console.log(b.text()); // Lee(Designer)
</script>
```

클래스의 게터(getter) / 세터(setter)
게터(get) : "값을 읽을 때" 자동으로 실행되는 함수
세터(set) : 값을 저장(=대입)할 때" 자동으로 실행되는 함수

- 게터(getter) 예시
```
class User {
  constructor(name) {
    this.name = name;    // 내부 저장소
  }
  get name() {               // 읽을 때 실행
    return this.name.toUpperCase();
  }
}
const u = new User("hong");
console.log(u.name); // HONG   ← 함수처럼 보이지 않고 속성처럼!
```

- 세터(setter) 예시
```
class Person {
  constructor(age) {
    this.age = age;  // 여기서 세터가 자동 호출
  }

  get age() {
    return this._age;  // _는 내부용이라는 표시 this.age와 동일
  }

  set age(v) {
    if (v < 0) throw new Error("나이는 음수 불가!");
    this._age = v;
  }
}

const p = new Person(20);
console.log(p.age); // 값을 읽어주는 게터 호출 하여 20을 출력
p.age = -5;            // 값을 저장하는 세터 호출하여 -5값을 저장 ->  에러 발생: 나이는 음수 불가!
```

예제 1 회원 가입시 비밀번호 검증하는 클래스

```
class User {
  constructor(username, password) {
    this.username = username;
    this.password = password; // 세터 자동 호출
  }

  get password() {
    return "*".repeat(this._password.length); // 외부 노출 시 마스킹
  }

  set password(v) {
    if (v.length < 8) throw new Error("비밀번호는 최소 8자리 이상이어야 합니다.");
    this._password = v;
  }
}

const u = new User("kim", "12345678");
console.log(u.password); // ********
u.password = "123"; //      에러 발생
```

문제 1. 상품명과 가격을 주고, 가격과 부가세를 출력할 수 있는 클래스, 단 가격은 음수 입력 불가

```
class Product {
  constructor(name, price) {
    this.name = name;
    this.price = price; // 세터 호출
  }

  get priceVAT() {
    return this._price * 1.1; // 읽을 때 10% 부가세 포함 반환
  }

  get price() {
    return this._price;
  }

  set price(v) {
    if (v < 0) throw new Error("가격은 음수 불가");
    this._price = v;
  }
}

const p = new Product("Laptop", 1000);
console.log(p.price);             // 1000
console.log(p.priceVAT); // 1100
```