Item 56정보를 감추는 목적으로 private 사용하지 않기

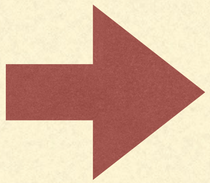
강철원 ryan-dia

TS에는 public, protected, private 접근 제어자를 사용해서
공개규칙을 강제할 수 있는 것으로 오해할 수 있다.

그러나 public, protected, private 같은 접근 제어자는
TS 키워드이기 때문에 컴파일 후에는 제거된다.

```
class Diary {
    private secret = 'cheated on my English test';
}

const diary = new Diary();
diary.secret
```



```
class Diary {
    constructor() {
        this.secret = 'cheated on my English test';
    }
}
const diary = new Diary();
diary.secret
```

TS 접근 제어자들은 런타임에 아무런 효력이 없다.

심지어 단언문을 사용하면 타입스크립트 상태에서도 private 속성에 접근할 수 있다.

```ts
class Diary {
  private secret = 'cheated on my English test';
}

const diary = new Diary();
(diary as any).secret; // OK
```

❗ 따라서 정보를 감추기 위해 private을 사용하면 안된다.

📌접근 가능한 위치

구분	선언한 클래스 내	상속받은 클래스 내	인스턴스
private	O	X	X
protected	O	O	X
public	O	O	O

정보를 숨기기 위해 좋은 방법으로는 두 가지가 있습니다.

1. 클로저
2. 비공개 필드

1. 클로저

```
declare function hash(text: string): number;

class PasswordChecker {
  checkPassword: (password: string) => boolean;
  constructor(passwordHash: number) {
    this.checkPassword = (password: string) => {
      return hash(password) === passwordHash;
    }
  }
}

const checker = new PasswordChecker(hash('s3cret'));
checker.checkPassword('s3cret');  // Returns true
```

주의사항

- passwordHash를 생성자 외부에서 접근할 수 없기 때문에 접근해야하는 메서드는 전부 생성자 내부에 정의되어야 한다.
- 메서드 정의가 생성자 내부에 존재하게 되면, 인스턴스를 생성할 때마다 각 메서드의 복사본이 생성되기 때문에 메모리가 낭비된다.

2. 비공개 필드

```
declare function hash(text: string): number;
class PasswordChecker {
    #passwordHash: number;

  constructor(passwordHash:number) {
    this.#passwordHash = passwordHash;
  }

  checkPassword(password: string) {
    return hash(password) === this.#passwordHash
  }
}

const checker = new PasswordChecker(hash("s3cret"))
checker.checkPassword("secret")  // false
checker.checkPassword("s3cret")  // true
```

클로저 기법과 다르게 클래스 메서드나 동일한 클래스의 개별 인스턴스끼리 접근이 가능