ШПАРГАЛКА ПО КЛАССАМ В TYPESCRIPT

 ОСНОВЫ КЛАССОВ
 Объявление класса:
```
class MyClass {
  prop: string;
  constructor(value: string) {
    this.prop = value;
  }
}
```

 Поля:
- Указываются внутри класса.
- Можно задать значения по умолчанию.
- При --strictPropertyInitialization нужно инициализировать в конструкторе или использовать !.

 МОДИФИКАТОРЫ
public	доступен везде (по умолчанию)
private	только внутри самого класса
protected	внутри класса и его наследников
readonly	только для чтения (нельзя менять)

 КОНСТРУКТОР + СВОЙСТВА:
```
class User {
  constructor(public name: string, private age: number) {}
}
```

 МЕТОДЫ:
```
class A {
  say(msg: string): void {
    console.log(msg);
  }
}
```

 НАСЛЕДОВАНИЕ:
```
class Animal {
  move() {}
}
class Dog extends Animal {
  bark() {}
}
```

Переопределение:
```
class Dog extends Animal {
  override move() {
    super.move();
    console.log("собака двигается");
  }
}
```

 АБСТРАКТНЫЕ КЛАССЫ:
```
abstract class Shape {
  abstract getArea(): number;
}
```

```
class Circle extends Shape {
  getArea() {
    return Math.PI * 2;
  }
}
```

 СТАТИЧЕСКИЕ СВОЙСТВА/МЕТОДЫ:
```
class MathHelper {
  static PI = 3.14;
  static square(x: number) {
    return x * x;
  }
}
MathHelper.square(2); // 4
```

 ГЕТТЕРЫ / СЕТТЕРЫ:
```
class Person {
  private _name = "";
  get name() {
    return this._name;
  }
  set name(n: string) {
    this._name = n;
  }
}
```

 this как тип (цепочки):
```
class Builder {
  value = "";
  set(v: string): this {
    this.value = v;
    return this;
  }
}
```

 Выражение класса:
```
const MyClass = class {
  say() {
    console.log("Hi");
  }
};
```

 typeof для типа самого класса:
```
class User {
  static admin = true;
}
type UserType = typeof User;
```