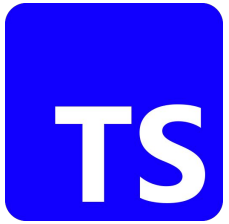
Code Academy



OOP
Abstrakcija ir polimorfizmas

TypeScript

10 paskaita

Paskaitos eiga

Kas yra Abstrakcija?

Kas yra Polimorfizmas?

Abstrakcijos ir polimorfizmo įgyvendinimas

Kas yra Abstrakcija?

Kas yra Abstrakcija?

Abstrakcija, tai klasės **įpareigojimas aprašyti** savybes ir/arba metodus.

Įpareigoti klasę išpildyti abstrakciją galima 2 būdais:

1. Naudojant abstrakčią klasę:
   Tėvinė klasė padaroma abstrakti, ir tuomet ji negali turėti savo objektų. Abstrakčios klasės metodai kurie turi papildomi direktyva "**abstract**" privalo būti išpildyti (angl: ***implemented***) vaikinėse klasėse kurios paveldės abstrakčią klasę.
2. Naudojant karkasą (angl: ***interface***):
   Interface'e aprašomi savybių ir metodų tipai. Tuomet bet kokia klasė gali implementuoti abstrakciją aprašytą karkasu. Tai daroma naudojant žodį "**implements**"

Abstrakcijos tikslas

Abstrakcijos tikslas - bendros paskirties, bet skirtingų klasių objektams suteikti vienodos struktūros ir pasiekiamų metodų įpareigojimus.

Abstrakcijos tikslas

Įpareigojus vienos paskirties, bet skirtingų klasių objektus pasižymėti bendrais bruožais, galime juos laikyti bet kokioje duomenų kolekcijoje ir atlikti veiksmus pagal tuos bendrus bruožus, pvz. :

- Pašerti visus gyvūnus
- Išmokėti visiems darbuotojams atlyginimus
- Nukreipti visus žaidimo veikėjus viena kryptimi
- Pridėti visus objektus į žemėlapį
- Groti melodiją su visais instrumentais ir t.t.

Kuom skiriasi abstrakti klasė nuo interface'o?

Tai tikriausiai dažniausiai užduodamas klausimas darbo pokalbio metu. Verta perprasti, arba bent jau atsiminti.

Atsakymas:

Klasė, gali paveldėti tik vieną tėvinę klasę, tačiau gali implementuoti bet kokį kiekį interface'ų.

Abstrakčios klasės turėtų būti naudojamos aprašyti metodams kurie vaikinėse klasėse implementuos logiką susijusią su klasės esybe, nes pats paveldimumo ryšys yra sudaromas esybės pagrindu - "klasė **A** yra klasė **B**".

Interface'ai naudojami įpareigoti klases išpildyti funkcionalumą nesusijusį su klasės esybe.

Praktikoje geriau vengti abstrakčių klasių ir naudoti interface'us, nes plečiantis projektui reikia papildyti klases abstrakcijomis. Jeigu tos abstrakcijos įgalintos abstrakčių klasių pavidalu, beveik visada reikia keisti klasių paveldimumo hierarchijos struktūrą, o kartais susidaro atvejų, kuomet sudaryti klasių hierarchinį medį tiesiog neįmanoma.

Kas yra Polimorfizmas?

Kas yra polimorfizmas

Polimorfizmas - tai daugiakūniškumas, dviprasmiškumas.

Programavimo inžinerijoje ši savoka reiškia, kad vykdant taip pat atrodančius metodų iškvietimus, kviečiame visiškai skirtingas funkcijas.

Taip įvyksta todėl, kad iš pirmo pažiūros taip pat atrodantys kintamieji yra iš tiesų yra skirtingų klasių objektai pasižymintys tais pačiais metodas. **Tokia elgsena yra abstrakcijos pasekmė.**

Abstrakcija ir polimorfizmas yra tarpusavyje susijusios objektinio programavimo savybės.

Abstrakcijos ir polimorfizmo įgyvendinimas

Abstrakčios klasės kūrimas

Panagrinėsime abstrakcijos ir polimorfizmo pavyzdį naudojant 2D figūras. Visos figūros, turi perimetrą ir plotą, tačiau jis skaičiuojamos skirtingai.

Figūra yra abstrakti - todėl ir jos klasė turi būti abstrakti.

Negalime sukurti *tiesiog* figūros - kuriame arba keturkampį, arba stačiakampį, arba kitą konkrečią figūrą. Figūra yra abstraktus apibūdinimas apjungiantis konkrečias figūras pagal bendrus bruožus.

```
abstract class Shape2D {
  public abstract getPerimeter(): number;
  public abstract getArea(): number;
}
```

Konkreti keturkampio klasė

```
class Rectangle extends Shape2D {
  private height: number;
  private width: number;

  public constructor(width: number, height: number) {
    super();

    this.width = width;
    this.height = height;
  }

  public getPerimeter = (): number => 2 * this.height + 2 * this.width;

  public getArea = (): number => this.width * this.height;
}
```

Konkreti apskritimo klasė

```
class Circle extends Shape2D {
  private radius: number;

  public constructor(radius: number) {
    super();

    this.radius = radius;
  }

  public getPerimeter = (): number => 2 * Math.PI * this.radius;

  public getArea = (): number => Math.PI * this.radius ** 2;
}
```

Bendros abstrakcijų kolekcijos kūrimas ir polimorfizmas

Abstrakcija

```
const shapes: Shape2D[] = [
  new Rectangle(500, 200),
  new Rectangle(300, 300),
  new Circle(40),
  new Circle(70),
];

shapes.forEach((shape) => {
  console.log(shape.getPerimeter()); 1400, 1200, 251.32741228718345, 439.822971502571
});

shapes.forEach((shape) => {
  console.log(shape.getArea()) 100000, 90000, 5026.5482245743669, 15393.804002589986
});
```

Polimorfizmas

Klausimai?

Paskaitos darbas

Paskaitos darbas

Paskaitoje atliksime užduotis, tokia eiga:

1. Sprendžiame užduotis savarankiškai
2. Po savarankiško sprendimo laiko (10-30 min.) dėstytojas išsprendžia 1 užduotį argumentuodamas sprendimą
3. Studentai užduoda klausimus apie sprendimą
4. Sprendimų palyginimas
5. Atliekama sekanti užduotis

Jeigu išsprendėte užduotį anksčiau nei kiti, spręskite sekančias užduotis.

Užduoties aptarimo metu, nesidrovėkite klausti kuo daugiau klausimų. Nebūtinai jūsų sprendimas yra prastesnis. Galbūt net geresnis?

Iki kito karto!