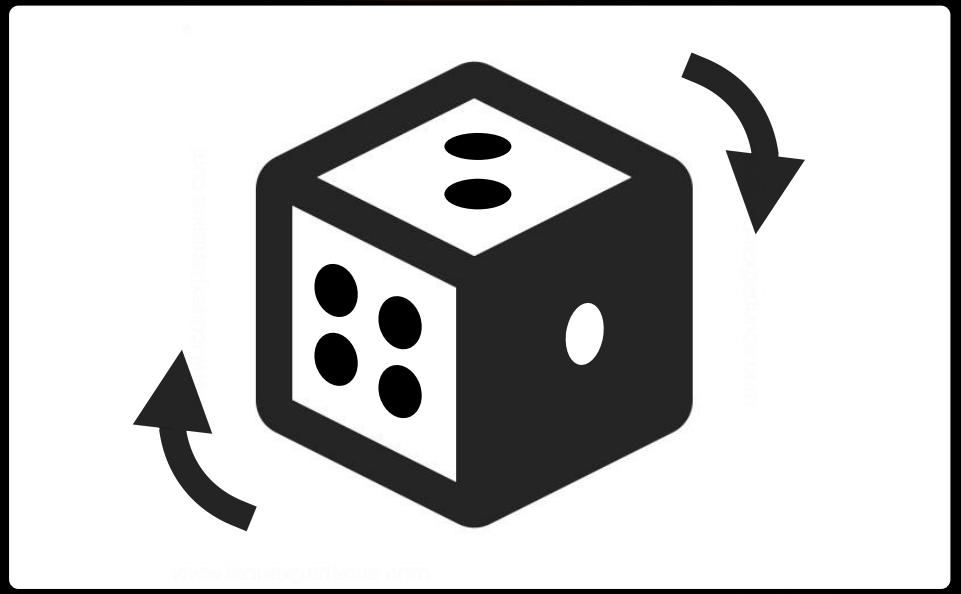
Елементи на класа

Полета, свойства, методи и конструктори

Учителски екип

Обучение за ИТ кариера

https://it-kariera.mon.bg/e-learning/



Съдържание

1. Методи
2. Полета
3. Свойства
4. Getter и Setter методи
5. Конструктори

Елементи на класа

- Клас се дефинира чрез състояние и поведение
- Полетата съхраняват състоянието
- Методите описват поведението

```
class Dice {
  int sides;
  string type;

  void Roll(){ … }
}
```

Полета

Метод

Методи

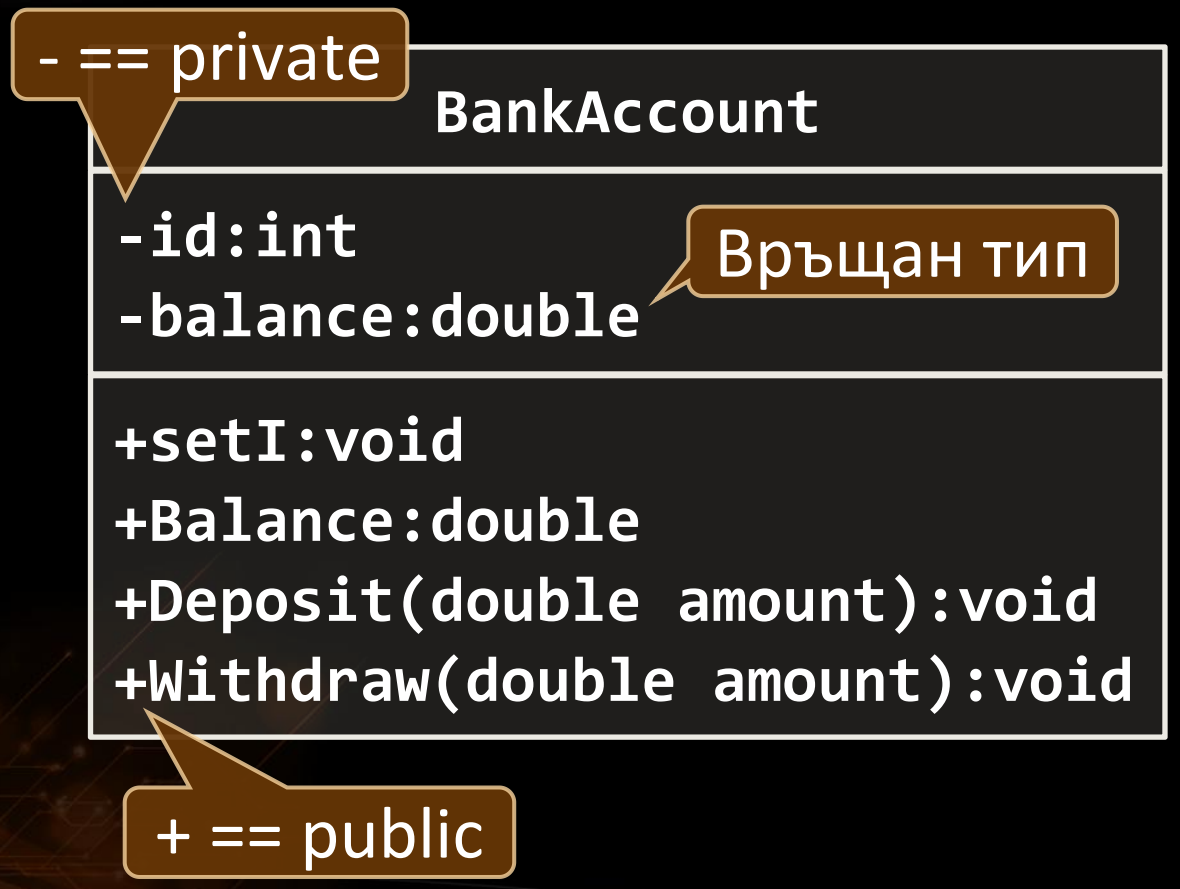
- Те са изпълним код (алгоритъм), който променя състоянието

```
class Dice {
  public int sides;
  private Random rnd = new Random();
  public int Roll()
  {
      int rollResult = rnd.Next(1, this.sides + 1);
      return rollResult;
  }
}
```

this сочи към тази инстанция

Задача: Getter-и и Setter-и

- Създайте клас BankAccount




```
public static void Main()
{
    BankAccount acc = new BankAccount();

    acc.ID = 1;
    acc.Deposit(15);
    acc.Withdraw(5);

    Console.WriteLine(acc.ToString());
}
```

Предефинирайте toString()

Решение: Getter-и и Setter-и

```
private double balance;
public void Deposit(double amount)
{
  this.balance += amount;
}
public void Withdraw(double amount)
{
  this.balance -= amount;
}
public override string ToString()
{
  return $"Account {this.id}, balance {this.balance}";
}
```

Полета

- Полетата на класа пазят информацията. Те имат тип и име.

```
class Dice {
   string type;
   int sides;
   int[] rollFrequency;
   Person owner;
   …
}
```

Полетата могат да са от всякакъв тип

Свойства

- Осигуряват достъп до данните. Използва се за създаване на методи за четене и методи за промяна (getters и setters).

```
class Dice {
  private int sides;

  public int Sides
  {
    get { return this.sides; }
    private set { this.sides = value; }
  }
}
```

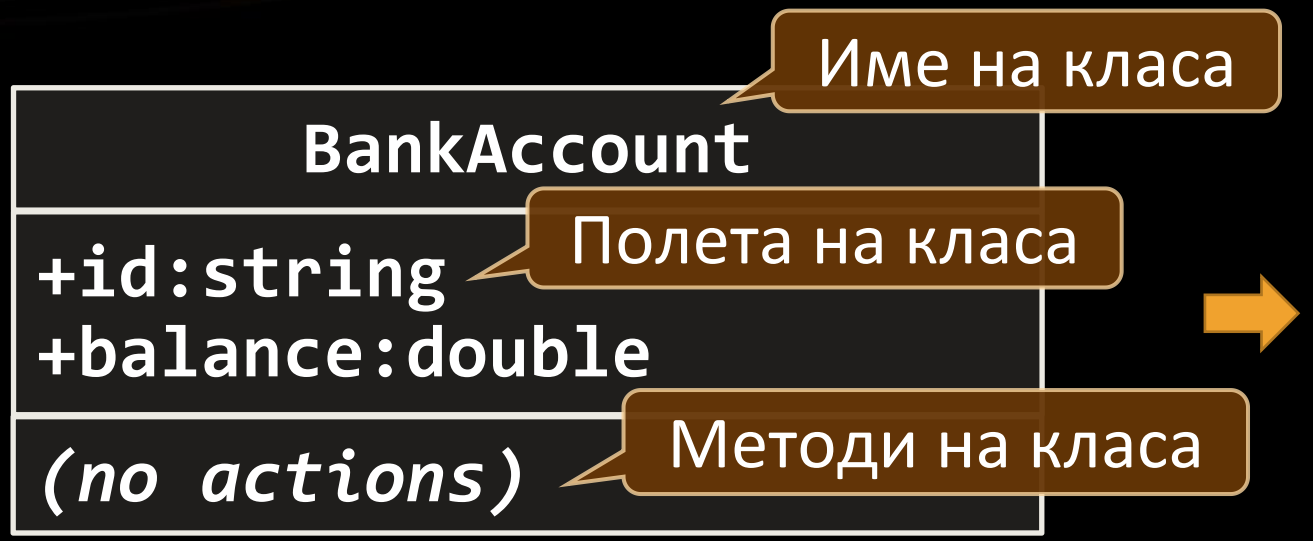
Полето е скрито

Getter-а предоставя достъп до полето

Setter-а позволява промяна на полето

Задача: Описване на клас Банкова сметка

- Създайте клас BankAccount




```
class BankAccount
{
    private string id;
    private decimal balance;

    public string Id
    {
        get { return this.id; }
        set { this.id = value; }
    }

    public decimal Balance
    {
        get { return this.balance; }
        set { this.balance = value; }
    }
}
```

- Подсигурете се, че сте избрали подходящи имена!

Решение: Описване на клас Банкова сметка

```
private string id;
private decimal balance;

public string Id
{
      get { return this.id; }
      set { this.id = value; }
}
public decimal Balance
{
      get { return this.balance; }
      set { this.balance = value; }
}
```

Конструктори

- Специален вид методи, извиквани при създаване на обекта

```
class Dice
{
  int sides;
  public Dice()
  {
    this.sides = 6;
  }
}
```

Предефиниране
на конструктора
по подразбиране

Начално състояние на обекта

- Конструкторите задават началното състояние на обекта

```
class Dice
{
  int sides; int[] rollFrequency;
  public Dice(int sides)
  {
    this.sides = sides;
    this.rollFrequency = new int[sides];
  }
}
```

Винаги подсигурете коректно състояние

Конструктори (2)

- Може да имате множество конструктори за даден клас

```
class Dice
{
  int sides;
  public Dice()
  {
    this.sides = 6;
  }
  public Dice(int sides)
  {
    this.sides = sides;
  }
}
```

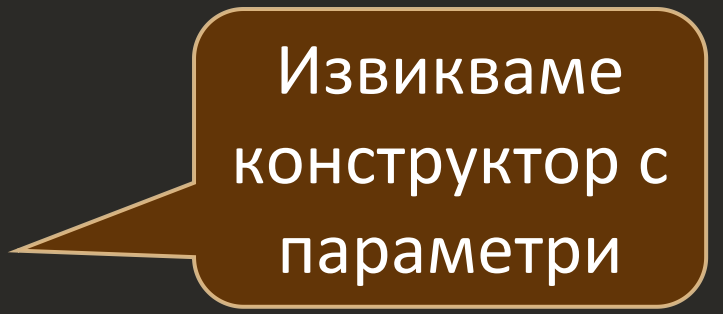
Конструктор без параметри

Конструктор с параметри

Верижно извикване на конструктори

- Конструкторите могат да се извикват един друг

```
class Dice
{
  int sides;
  public Dice() : this(6)
  {
  }
  public Dice(int sides)
  {
    this.sides = sides;
  }
}
```



Задача: Дефиниране на клас Физ.лице

- Създайте клас Person

Person
-name:string -age:int -accounts:List<BankAccount>
+Balance():double +Person(String name, int age) +Person(String name, int age, List<BankAccount> accounts)

```
public class Person
{
    private string name;
    private int age;
    private List<BankAccount> accounts;
}
```

Решение: Дефиниране на клас Физ.лице

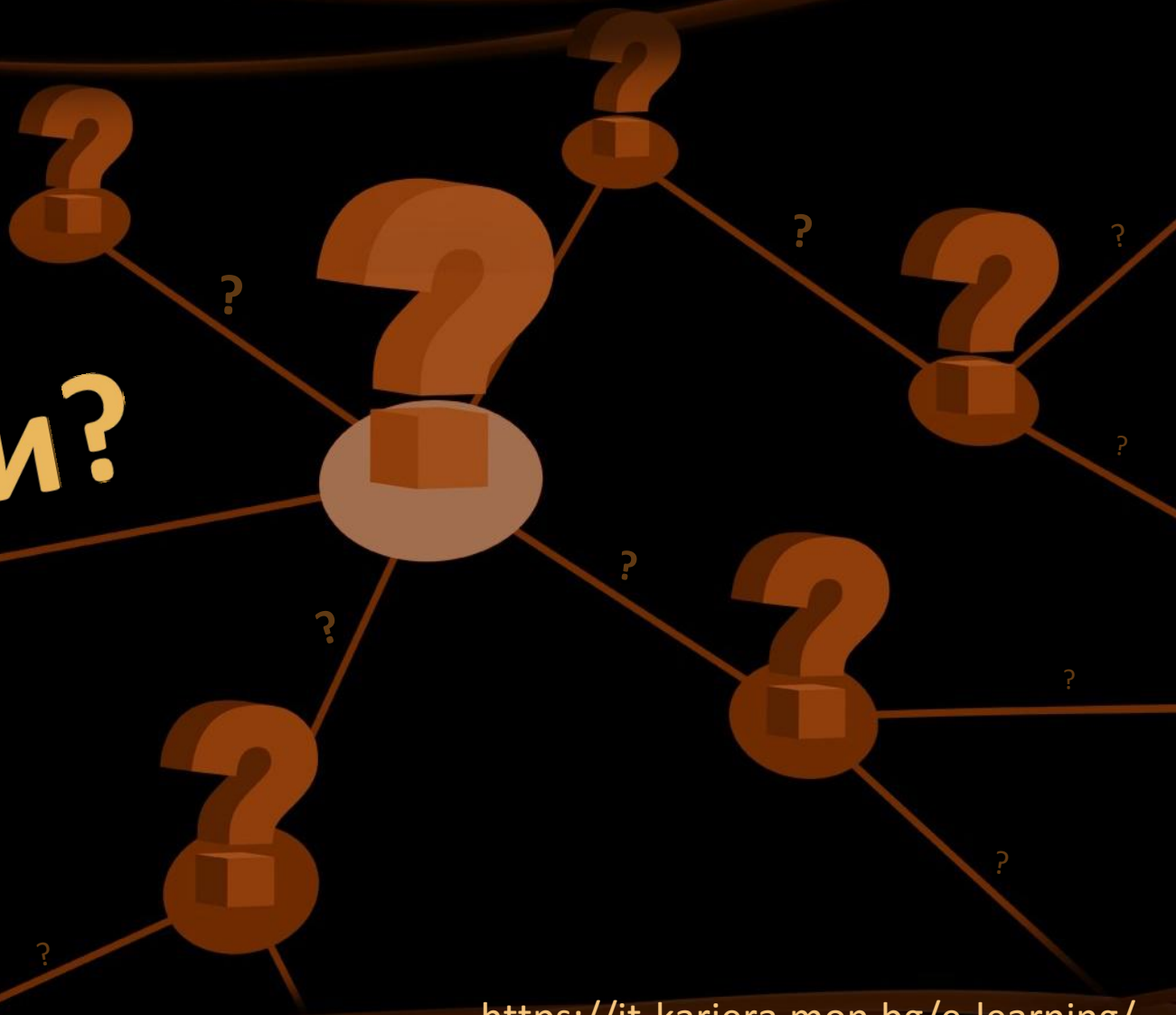
```
public class Person
{
  private string name;
  private int age;
  private List<BankAccount> accounts;
  public Person(string name, int age)
      : this(name, age, new List<BankAccount>))
    {}
  public Person(string name, int age, List<BankAccount> accounts)
  {
    this.name = name;
    this.age = age;
    this.accounts = accounts;
  }
}
```

Какво научихме днес?

- Методите описват поведението
- Полетата съхраняват състоянието
- Getter / setter методите са за достъп и промяна на полетата
- Свойствата предоставят достъп до и промяна на полетата на класа
- Конструкторите задават началното състояние на обекта
- Може да има множество различни конструктори за даден клас
- Конструкторите могат да се извикват един друг

Елементи на класа

Въпроси?



https://it-kariera.mon.bg/e-learning/

Лиценз

- Настоящият курс (слайдове, примери, видео, задачи и др.) се разпространяват под свободен лиценз "Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International"

- Благодарности: настоящият материал може да съдържа части от следните източници
  - Книга "Основи на програмирането със C#" от Светлин Наков и колектив с лиценз CC-BY-SA