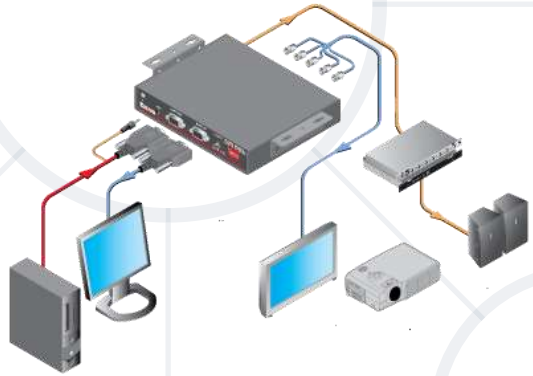
Абстрактни класове и интерфейси

Разлика между абстракция и енкапсулация
Разлика между интерфейси и абстрактни класове



SoftUni Team

Technical Trainers

Software University

SoftUni

Software University

https://softuni.bg

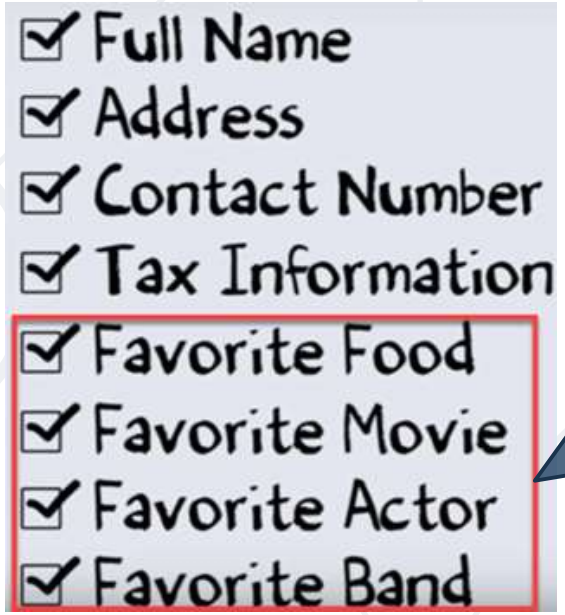
Съдържание

1. Абстракция
2. Интерфейси
3. Абстрактни класове
4. Разлика между интерфейси и абстрактни класове

Постигане на абстракция

Абстракция в ООП

- **Абстракцията** “показва" само **най-важните атрибути** и “крие“ ненужната информация
- Помага да се **управлява** комплексността
- Абстракцията позволява да се фокусираме върху това **какво прави даден обект**, вместо **как го прави**



Не се нуждаем от тази информация в приложение на банка

Как постигаме абстракция?

- Има **два начина** да постигнем абстракция
  - чрез **интерфейси**
  - чрез **абстрактни класове**

```
public interface IAnimal {}
public abstract class Mammal {}
public class Person : Mammal, IAnimal {}
```

Разлика между абстракция и енкапсулация

- Абстракция
  - Процес на **скриване на детайлите по имплементацията** и показване само на функционалността на потребителя
  - Постига се чрез **интерфейси** и **абстрактни класове**
- Енкапсулация
  - Използва се, за да се **скрие кода** и **данните** в **даден клас**, за да се **защитят** от **външния свят**
  - Постига се чрез **модификатори за достъп** (private, protected, public…)

Работа с интерфейси

Интерфейси в реалността

- Вътрешно допълнение от компилатора

```
public interface IPrintable
{
  void Print();
}
```

Ключова дума

Име (започва с I по конвенция)

compiler

```
public interface IPrintable
{
  public abstract void Print();
}
```

Пример за интерфейс

- Имплементацията на `Print()` се задава в клас `Document`

```
public interface IPrintable
{
  void Print();
}
```

Само сигнатурите

```
class Document : TextDocument, IPrintable, IWritable
{
  public void Print() { Console.WriteLine("Hello"); }
}
```

Класовете се изписват първи

Един или повече интерфейси

Експлицитен интерфейс (1)

- Клас, който **имплементира** интерфейс, може **експлицитно** да имплементира **членове** от този **интерфейс**

```
public interface IFile
{
  void ReadFile();
}
```

```
public interface IBinaryFile
{
  void ReadFile();
}
```

```
class FileInfo : IFile, IBinaryFile
{
  void IFile.ReadFile()
  {
    Console.WriteLine("Reading File");
  }
}
```

Експлицитно имплементиран член

Експлицитен интерфейс (2)

- Експлицитно имплементиран член **не може** да бъде достъпен през инстанцията на класа, а само през **инстанцията на интерфейса**

```
public interface IBinaryFile
{
  void ReadFile();
  void OpenBinaryFile();
}
```

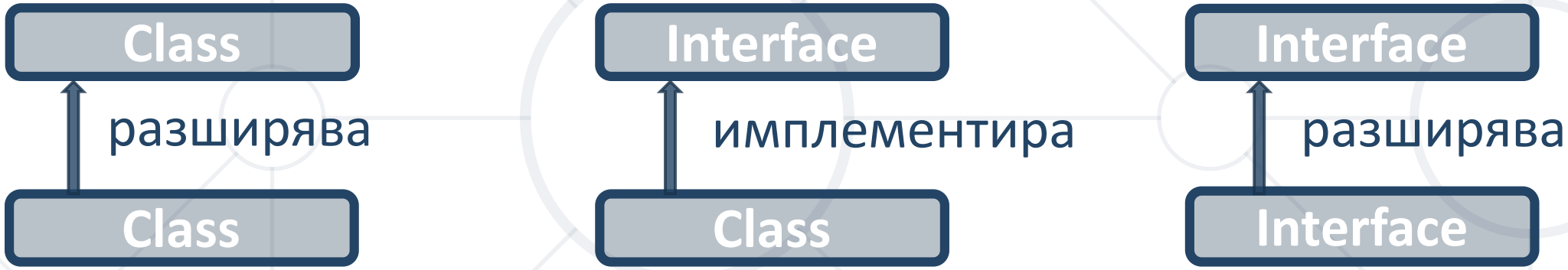
```
class FileInfo : IFile, IBinaryFile
{
  void IFile.ReadFile() {…}
  void OpenBinaryFile() {…}
}
```

Достъп през инстанцията

```
public static void Main()
{
  IBinaryFile file = new FileInfo();
  file.OpenBinaryFile();
}
```

Множествено наследяване

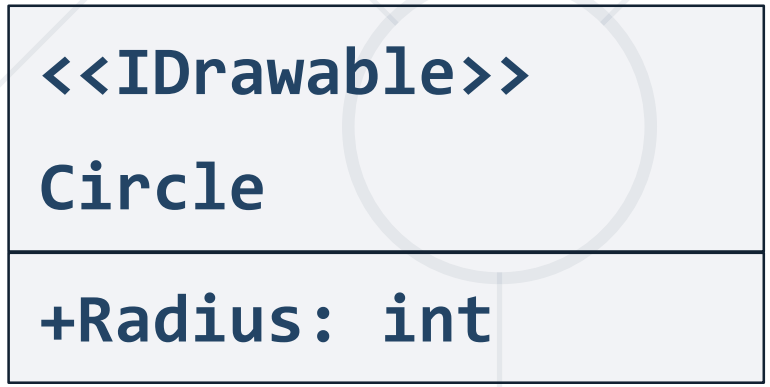
- Връзка между **класовете** и **интерфейсите**

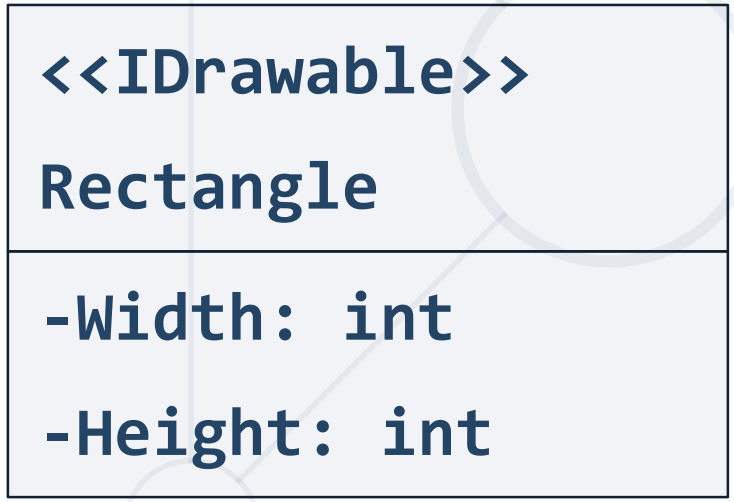


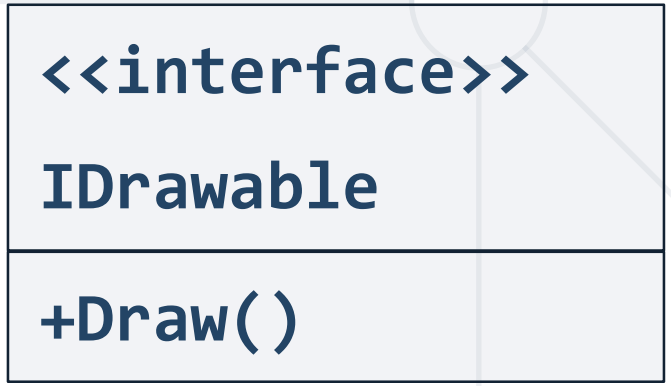
- Множествено наследяване

Задача: Фигури

- Създайте проект, който съдържа **интерфейс** за **рисуваеми обекти**
- Имплементирайте два вида фигури: **Circle (кръг)** и **Rectangle (правоъгълник)**
- И двата класа трябва да отпечатват фигурата с "*"







Решение: Фигури

```
public interface IDrawable
{
  void Draw();
}
```

```
public class Rectangle : IDrawable
{
  // TODO: Добавете полета и конструктор
  public void Draw() { // TODO: implement } }
```

```
public class Circle : IDrawable
{
  // TODO: Добавете полета и конструктор
  public void Draw() { // TODO: implement } }
```

Решение: Фигури – правоъгълник

```
public void Draw()
{
  DrawLine(this.width, '*', '*');
  for (int i = 1; i < this.height - 1; ++i)
    DrawLine(this.width, '*', ' ');
  DrawLine(this.width, '*', '*');
}
private void DrawLine(int width, char end, char mid)
{
  Console.Write(end);
  for (int i = 1; i < width - 1; ++i)
    Console.Write(mid);
  Console.WriteLine(end);
}
```

Решение: Фигури – кръг

```
double rIn = this.radius - 0.4;
double rOut = this.radius + 0.4;
for (double y = this.radius; y >= -this.radius; --y)
{
  for (double x = -this.Radius; x < rOut; x += 0.5)
  {
    double value = x * x + y * y;
    if (value >= rIn * rIn && value <= rOut * rOut)
      Console.Write("*");
    else
      Console.Write(" ");
  }
  Console.WriteLine();
}
```

Проверете решението си тук : https://judge.softuni.bg/Contests/Practice/Index/3165#0

Абстрактни класове и методи

Абстрактен клас

- Можем да използваме **абстрактен клас** като **базов клас** и всички производни класове трябва да имплементират абстрактните членове

```
abstract class Shape
{
  public abstract int GetArea();
}
```

Наименуван метод

```
class Square : Shape
{
  int side;
  public Square(int n) => side = n;
  public override int GetArea() => side * side;
}
```

Дъщерните класове допълват имплементацията

Абстрактен клас

- Абстрактните класове **може** да съдържат **абстрактни методи** и **access-ори**

```
abstract class BaseClass
{
  protected int x = 100;
  public abstract void AbstractMethod();
  public abstract int X { get; }
}
```

```
class DerivedClass : BaseClass
{
  public override void AbstractMethod() { x++; }
  public override int X   // overriding property
  { get { return x + 10; } }
}
```

Абстрактен клас

- Трябва да задава **имплементация** за всички **наследени** членове на интерфейса

```
interface IService
{
  int Add();
}
```

```
abstract class ServiceBase : IService
{
  public abstract int Add();
}
```

```
public static void Main()
{
  ServiceBase service = new ServiceBase();
}
```

Абстрактният клас **не може** да бъде инстанциран

Абстрактни методи

- **Абстрактният метод** имплицитно е **виртуален метод**
- Декларации на абстрактни методи са позволени **само в абстрактни класове**
- При декларацията на абстрактен метод не се задава имплементация:

```
abstract class ServiceBase : IService
{
  public abstract int Add();
}
```

Разлика между интерфейси и абстрактни класове

Разлика между интерфейси и абстрактни класове (1)

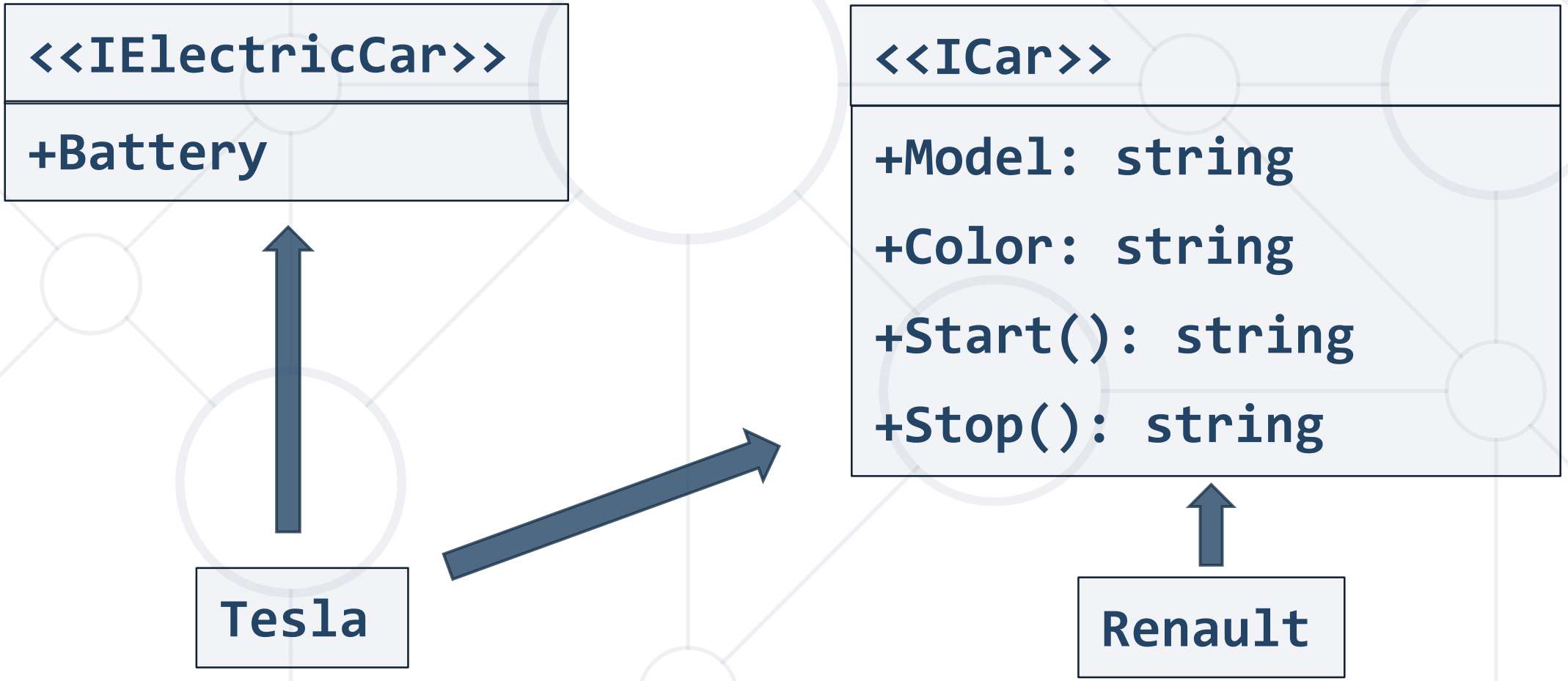
- Интерфейс
  - Един клас може да **имплементира множество интерфейси**
  - **Не може да има модификатори за достъп**, всичко е публично
  - **Не задава код**, само сигнатури
- Абстрактен клас (АС)
  - Може да **наследи само един абстрактен** клас
  - Може да **има модификатори за достъп** за полетата, функциите и свойствата
  - Може да **зададе имплементация** и/или само **сигнатура** , която трябва да бъде презаписана

Разлика между интерфейси и абстрактни класове (2)

- Интерфейс
  - **Не може** да се дефинират полета и константи
  - Ако добавим **нов метод, трябва да проследим всички имплементации** на интерфейса и **да дефинираме имплементацията** за новия метод
- Абстрактен клас
  - **Може** да се дефинират полета и константи
  - Ако добавим **нов метод**, имаме опцията да **осигурим имплементация по подразбиране**

Задача: Коли (1)

- Създайте йерархия от класове и интерфейси



Задача: Коли (2)

- Създайте йерархия от класове и интерфейси
  - Създайте интерфейс с име **IElectricCar**
    - Трябва да има свойство **Battery**
  - Създайте интерфейс с име **ICar**
    - Трябва да има следните свойства: **Model: String**, **Color: String**
    - Трябва да има и следните методи: **Start(): String**, **Stop(): String**
- Създайте клас **Tesla**, който имплементира **IElectricalCar** и **ICar**
- Създайте клас **Renault**, който имплементира **ICar**

Решение: Коли (1)

```
public interface ICar {
    string Model { get; }
    string Color { get; }
    string Start();
    string Stop();
}

public interface IElectricCar {
    int Batteries { get; }
}
```

Решение: Коли (2)

```
public class Tesla : ICar, IElectricCar {
  public string Model { get; private set; }
  public string Color { get; private set; }
  public int Batteries { get; private set; }
  public Tesla (string model, string color, int batteries)
  { // TODO: Add Logic here }
  public string Start()
  { // TODO: Add Logic here }
  public string Stop()
  { // TODO: Add Logic here }
}
```

Решение: Коли (3)

```
public class Seat : ICar {
  public string Model { get; private set; }
  public string Color { get; private set; }
  public Tesla(string model, string color)
  { // TODO: Add Logic here }
  public string Start()
  { // TODO: Add Logic here }
  public string Stop()
  { // TODO: Add Logic here }
}
```

Проверете решението си тук: https://judge.softuni.bg/Contests/Practice/Index/3165#1

Обобщение

- **Абстракция** – “показва" само най-важните атр ибути и “крие" ненужната информация
- Как постигаме абстракция – чрез интерфейси и ли чрез абстрактни класове
- **Интерфейси** – Съдържат само сигнатурата на методите и свойствата
- **Абстрактни класове** – базовия клас и всички п роизводни класове трябва да имплементират абстрактни членове

Въпроси?

SoftUni

Software University | SoftUni Svetlina | SoftUni Creative | SoftUni Digital | SoftUni Foundation | SoftUni Kids

Лиценз

- Този курс (презентации, примери, демонстрационен код, упражнения, домашни, видео и други активи) представлява **защитено авторско съдържание**
- Нерегламентирано копиране, разпространение или използване е незаконно
- © СофтУни – https://softuni.org
- © Софтуерен университет – https://softuni.bg