

# Полиморфизъм чрез интерфейси



Учителски екип

Обучение за ИТ кариера

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



## Задача: Машина

- Създайте интерфейс **IMachine**, за машина, която трябва да има функционалност за пускане и спиране, добавете и свойство за вида на машината. Създайте класове **Car**, **LawnMower**, **Truck**, **Airplane**, които да имплементират интерфейса. Всеки от класовете трябва да има и конструктор, който задава вида на машината. Създайте клас **MachineOperator**, чиято цел е да пуска и спира машина, която му се подава чрез конструктора

# Решение: Машина (1)

- Нека да създадем **интерфейса** за машина и да заложим методите за пускане и спиране

```
interface IMachine
{
    string MachineType { get; set; }
    bool Start();
    bool Stop();
}
```

## Решение: Машина (2)

- Нека да създадем класа **Car**, имплементиращ интерфейса

```
public string MachineType { get; set; }  
public Car() {  
    this.MachineType = "Car";  
}  
public bool Start() {  
    Console.WriteLine("Car starting...");  
    return true;  
}  
public bool Stop() {  
    Console.WriteLine("Car stopping...");  
    return true;  
}
```



## Решение: Машина (3)

- Останалите класове – **Truck**, **Airplane**, **LawnMower**, имат напълно аналогична имплементация с тази на **Car**
- Следва дефиниране на класа за **MachineOperator**

## Решение: Машина (4)

- Нека да създадем класа **MachineOperator**

```
private IMachine entity;  
public IMachine Entity { get { return this.value; }  
    set {  
        this.entity = value;  
        Console.WriteLine("Machine operator is operating:  
"+value.MachineType);  
    }  
}  
public MachineOperator(IMachine entity){  
    this.Entity = entity;  
}
```

## Решение: Машина (5)

- А сега да добавим методи `Start()` и `Stop()`:

```
public bool Start() {  
    return Entity.Start();  
}
```

```
public bool Stop() {  
    return Entity.Stop();  
}
```

## Решение: Машина (6)

- В Program.cs създаваме по един обект от всяка машина и един обект от **MachineOperator**

```
Car car = new Car();  
LawnMower lawnMower = new LawnMower();  
Airplane airplane = new Airplane();  
Truck truck = new Truck();  
  
MachineOperator mo = new MachineOperator(car);  
mo.Start(); //пускаме машината  
mo.Stop(); //спираме машината  
mo.Entity = lawnMower; //сменяме машината  
mo.Start();  
mo.Stop();  
//TODO: аналогично можем да сменим машината с другите
```



# Какво научихме от тази задача

- Интерфейсите ни позволяват да постигнем полиморфистично поведение, което да не зависи от кода и да е лесно за промяна.
- Можем с лекота да добавим още машини без да счупим кода, като отразим и спецификите на всяка една от тях



# Полиморфизъм чрез интерфейси



Въпроси?



# Министерство на образованието и науката (МОН)

- Настоящият курс (презентации, примери, задачи, упражнения и др.) е разработен за нуждите на Национална програма "**Обучение за ИТ кариера**" на МОН за подготовка по професия "Приложен програмист"



Министерство  
на образованието  
и науката



Национална  
програма  
„Обучение за  
ИТ кариера“

- Курсът е базиран на учебно съдържание и методика, предоставени от **фондация "Софтуерен университет"** и се разпространява под свободен лиценз **CC-BY-NC-SA**



SoftUni  
Foundation

