

# Упражнения: Полиморфизъм

Можете да проверите решенията си в Judge системата: <https://judge.softuni.bg/Contests/3167/Polymorphism>

## 1. Математически операции

**Важно:** Трябва да имате публичен клас **StartUp** в namespace **Operations**.

Създайте клас **MathOperations**, който има 3 пъти метода **Add()**. **Add()** трябва да има следните варианти:

- **Add(int, int): int**
- **Add(double, double, double): double**
- **Add(decimal, decimal, decimal): decimal**

Трябва да можете да използвате класа по следния начин:

```
MathOperations mo = new MathOperations();
Console.WriteLine(mo.Add(2, 3));
Console.WriteLine(mo.Add(2.2, 3.3, 5.5));
Console.WriteLine(mo.Add(2.2m, 3.3m, 4.4m));
```

## Примери

Изход
5
11
9.9

## Решение

Създайте клас **MathOperation**, който изглежда по следния начин:

```
public int Add(int a, int b)
{
    return a + b;
}

public double Add(double a, double b, double c)
{
    return a + b + c;
}

public decimal Add(decimal a, decimal b, decimal c)
{
    return a + b + c;
}
```

## 2. Животни

**Важно:** Трябва да имате публичен клас StartUp в namespace.

Създайте клас Animal, който има следните полета:

- `name` - `string`
- `favouriteFood` - `string`

Animal има един виртуален метод `ExplainSelf(): string`.

Добавете още **два класа** - `Cat` and `Dog`. **Презапишете** метода `ExplainSelf()`, като добавите конкретния звук на животното на нов ред.

Трябва да можете да използвате класа по следния начин:

```
Animal cat = new Cat("Pesho", "Whiskas");
Animal dog = new Dog("Gosho", "Meat");

Console.WriteLine(cat.ExplainSelf());
Console.WriteLine(dog.ExplainSelf());
```

## Примери

Изход
I am Pesho and my favourite food is Whiskas MEEOW I am Gosho and my favourite food is Meat DJAAF

## Решение

```
public abstract class Animal
{
    2 references
    public string Name { get; protected set; }

    2 references
    public string FavoriteFood { get; protected set; }

    0 references
    protected Animal(string name, string favoriteFood)
    {
        this.Name = name;
        this.FavoriteFood = favoriteFood;
    }

    0 references
    public virtual string ExplainSelf()
    {
        return $"I am {this.Name} and my favourite food is {this.FavoriteFood}";
    }
}
```

```

public class Cat : Animal
{
    0 references
    public Cat(string name, string favouriteFood) : base(name, favouriteFood)
    {
    }

    4 references
    public override string ExplainSelf()
    {
        return base.ExplainSelf() + Environment.NewLine + "MEEOW";
    }
}

```

### 3. Фигури

**Важно:** Трябва да имате публичен клас **StartUp** в namespace **Shapes**.

Създайте йерархия, започваща с **абстрактен** клас **Shape**:

- **Абстрактни методи:**
  - **CalculatePerimeter(): double**
  - **CalculateArea(): double**
- **Виртуални методи:**
  - **Draw(): string**

Разширете класа **Shape** с два дъщерни класа:

- **Rectangle**
- **Circle**

Всеки от тях трябва да има:

- **Полета:**
  - **height** (височина) и **width** (ширина) за **Rectangle** (правоъгълник)
  - **radius** (радиус) за **Circle** (кръг)
- Енкапсулация за тези методи
- Публичен конструктор
- Методи за периметър и лице
- Override методи за рисуване

### 4. Превозни средства

Напишете програма, която има класове за 2 превозни средства (**Car** (кола) и **Truck** (камион)) и симулира **каране** и **зареждане на гориво**. И колата, и камиона имат **количество гориво (fuel quantity)** и **консумация на гориво (fuel consumption)** в литри за км. В допълнение, превозното средство може да се **кара за определена**

дистанция и да се **презарежда**. Тъй като е лято, и двете превозни средства използват климатик и тяхната консумация на гориво за км се **увеличава** с **0.9** литра за **колата** и с **1.6** литра за **камиона**.

Освен това **камионът** има малка дупка в своя резервоар и когато се **презареди**, запазва **само 95** от даденото гориво.

Ако превозното средство **не може** да измине дадената дистанция, **не променяйте** наличното гориво.

## Вход

- **Първи ред:** информация за колата във формат: "Car {fuel quantity} {liters per km}"
- **Втори ред:** информация за камиона във формат: "Truck {fuel quantity} {liters per km}"
- **Трети ред:** броя на командите **N**, които ще получите на следващите редове
- **На следващите N реда:** команди във формата:
  - "Drive Car {distance}"
  - "Drive Truck {distance}"
  - "Refuel Car {liters}"
  - "Refuel Truck {liters}"

## Изход

- След всяка команда **Drive**, ако **има** достатъчно гориво, **отпечатайте** съобщение в следния формат:
  - "Car/Truck travelled {distance} km"
- Ако **няма** достатъчно гориво, **отпечатайте**: "Car/Truck needs refueling"
- След получаване на последната команда, **отпечатайте** оставащото гориво и за колата, и за камиона, закръглено до **2 знака** след десетичната запетая, в следния формат:
  - "Car: {liters}"
  - "Truck: {liters}"

## Примери

Вход	Изход
Car 15 0.3 Truck 100 0.9 4 Drive Car 9 Drive Car 30 Refuel Car 50 Drive Truck 10	Car travelled 9 km Car needs refueling Truck travelled 10 km Car: 54.20 Truck: 75.00
Car 30.4 0.4 Truck 99.34 0.9 5 Drive Car 500 Drive Car 13.5 Refuel Truck 10.300 Drive Truck 56.2 Refuel Car 100.2	Car needs refueling Car travelled 13.5 km Truck needs refueling Car: 113.05 Truck: 109.13

## 5. Разширение на Vehicle

Използвайте кода си от **предишната** задача като отправна точка и добавете още функционалност. Добавете ново превозно средство - **Bus** (автобус). Към **всяко превозно средство** добавете ново свойство – капацитет на резервоара (**tank capacity**). Едно превозно средство **не може** от началото да има повече гориво от **tank capacity** или при зареждане да надмине капацитета на резервоара.

Ако **горивото за презареждане** е **повече** от наличното пространство, **отпечатайте "Cannot fit {fuel amount} fuel in the tank"** и **не добавяйте никакво гориво** към резервоара на превозното средство. Ако се направи опит да се **създаде** превозно средство с гориво, **повече** от капацитета на резервоара, го **добавете**, но с **празен резервоар**.

Добавете **нова команда** за автобуса. Може да го **карате със** или **без** хора. Когато превозвате хора, **климатикът е включен** и **консумацията на гориво** за километър се **увеличава** с **1.4** литра. Ако **няма хора в автобуса**, климатикът е изключен и консумацията на гориво **не се променя**.

Добавете **валидация** за **количеството гориво**, дадено в команда **Refuel** – ако е **0** или **отрицателно**, отпечатайте **"Fuel must be a positive number"**.

### Вход

- На **първите три реда**: информация за превозни средства в следния формат:
  - **"Vehicle {начално количество гориво} {консумация на гориво} {капацитет на резервоара}"**
- **Четвърти ред** – броя на командите **N**, които ще получите
- На **следващите N реда** – команди във формата:
  - **"Drive Car {distance}"**
  - **"Drive Truck {distance}"**
  - **"Drive Bus {distance}"**
  - **"DriveEmpty Bus {distance}"**
  - **"Refuel Car {liters}"**
  - **"Refuel Truck {liters}"**
  - **"Refuel Bus {liters}"**

### Изход

- След всяка команда **Drive**, ако **има достатъчно гориво**, отпечатайте:
  - **"Car/Truck travelled {distance} km"**
- Ако **няма достатъчно гориво**, отпечатайте:
  - **"Car/Truck needs refueling"**
- Ако се опитате да презаредите със **стойност  $\leq 0$** , отпечатайте:
  - **"Fuel must be a positive number"**
- Ако горивото за презареждане **не може да се побере в резервоара**, отпечатайте:
  - **"Cannot fit {fuel amount} fuel in the tank"**

- След получаване на команда "End", отпечатайте **оставащото гориво** за всички превозни средства, закръглено до **2 знака** след десетичната запетая, в следния формат:
  - "Car: {liters}"
  - "Truck: {liters}"
  - "Bus: {liters}"

## Примери

Вход	Изход
Car 30 0.04 70	Fuel must be a positive number
Truck 100 0.5 300	Fuel must be a positive number
Bus 40 0.3 150	Cannot fit 300 fuel in the tank
8	Bus travelled 10 km
Refuel Car -10	Cannot fit 1000 fuel in the tank
Refuel Truck 0	Bus needs refueling
Refuel Car 10	Cannot fit 1000 fuel in the tank
Refuel Car 300	Car: 40.00
Drive Bus 10	Truck: 100.00
Refuel Bus 1000	Bus: 23.00
DriveEmpty Bus 100	
Refuel Truck 1000	