

Упражнения: Класове и обекти

Можете да тествате решенията си в Judge системата: <https://judge.softuni.bg/Contests/3161/Classes-and-Objects>

1. Кола

Важно: трябва да имате **публичен StartUp** клас в namespace **CarManufacturer**.

Създайте **публичен клас** с име **Car**.

Класът трябва да има **частни полета** за:

- **make:** стринг
- **model:** стринг
- **year:** цяло число

Класът трябва да има **публични свойства** за:

- **Make:** стринг
- **Model:** стринг
- **Year:** цяло число

Трябва да можете да използвате класа по следния начин:

```
public static void Main(string[] args)
{
    Car car = new Car();

    car.Make = "VW";
    car.Model = "MK3";
    car.Year = 1992;

    Console.WriteLine($"Make: {car.Make}\nModel: {car.Model}\nYear: {car.Year}");
}
```

2. Разширение на клас Car

Важно: трябва да имате **публичен клас StartUp** в namespace **CarManufacturer**.

Създайте **публичен клас Car** с допълнителни полета.

Класът трябва да има **частни полета** за:

- **make:** стринг
- **model:** стринг
- **year:** цяло число (int)
- **fuelQuantity:** реално число (double)
- **fuelConsumption:** реално число (double)

Класът трябва да има **публични свойства** за:

- **Make:** стринг

- **Model:** стринг
- **Year:** цяло число (int)
- **FuelQuantity:** реално число (double)
- **FuelConsumption:** реално число (double)

Класът трябва да има **методи** за:

- **Drive(double distance): void** – този метод проверява дали разликата между количеството гориво (fuelQuantity) и разстоянието (distance), умножено по консумацията на гориво (fuelConsumption), е **по-голяма от 0**. Ако е по-голяма от 0, извадете произведението на разстоянието (distance) и консумацията на гориво (fuelConsumption) от наличното количество (fuelQuantity). В противен случай, отпечатайте следното съобщение:
"Not enough fuel to perform this trip!"
- **WhoAmI(): string** – връща следното съобщение:
"Make: {this.Make}\nModel: {this.Model}\nYear: {this.Year}\nFuel:
{this.FuelQuantity:F2}L"

Трябва да можете да използвате класа по следния начин:

```
public static void Main(string[] args)
{
    Car car = new Car();

    car.Make = "VW";
    car.Model = "MK3";
    car.Year = 1992;
    car.FuelQuantity = 200;
    car.FuelConsumption = 200;
    car.Drive(2000);
    Console.WriteLine(car.WhoAmI());
}
```

3. Конструктор на клас Car

Използвайки класа от предишната задача, създайте конструктор без параметри със следните стойности по подразбиране:

- **Make** – VW
- **Model** – Golf
- **Year** – 2025
- **FuelQuantity** – 200
- **FuelConsumption** – 10

Създайте втори конструктор, който приема **make**, **model** и **year** и извиква базовия конструктор със стойности по подразбиране за **fuelQuantity** и **fuelConsumption**.

```

public Car(string make, string model, int year)
: this()
{
    this.Make = make;
    this.Model = model;
    this.Year = year;
}

```

Създайте трети конструктор, който приема **make**, **model**, **year**, **fuelQuantity** и **fuelConsumption** при инициализация и преизползва втория конструктор, за да присвои стойност на **make**, **model** и **year**.

```

public Car(string make, string model, int year, double fuelQuantity, double fuelConsumption)
: this(make, model, year)
{
    this.FuelQuantity = fuelQuantity;
    this.FuelConsumption = fuelConsumption;
}

```

Отидете във файла **StartUp.cs** и създайте 3 различни инстанции на класа **Car**, всяка с различен вариант (overload) на конструктора.

```

public static void Main(string[] args)
{
    string make = Console.ReadLine();
    string model = Console.ReadLine();
    int year = int.Parse(Console.ReadLine());
    double fuelQuantity = double.Parse(Console.ReadLine());
    double fuelConsumption = double.Parse(Console.ReadLine());

    Car firstCar = new Car();
    Car secondCar = new Car(make, model, year);
    Car thirdCar = new Car(make, model, year, fuelQuantity,
        fuelConsumption);
}

```

4. Двигател и гуми на колата

Използвайки класа **Car**, който вече създадохте, дефинирайте друг клас с име **Engine**.

Класът трябва да има **частни** полета за:

- **horsePower**: цяло число (**int**)
- **cubicCapacity**: реално число (**double**)

Класът трябва да има **публични** свойства за:

- **HorsePower**: **int**
- **CubicCapacity**: **double**

Класът трябва да има и конструктор, който приема **horsepower** и **cubicCapacity** при инициализация:

```
public Engine(int horsepower, double cubicCapacity)
{
    this.HorsePower = horsepower;
    this.CubicCapacity = cubicCapacity;
}
```

Създайте нов клас с име **Tire**.

Класът трябва да има **частни полета** за:

- **year**: цяло число (int)
- **pressure**: реално число (double)

Класът трябва да има **публични свойства** за:

- **Year**: цяло число (int)
- **Pressure**: реално число (double)

Класът трябва да има и конструктор, който приема **year** и **pressure** при инициализация:

```
public Tire(int year, double pressure)
{
    this.Year = year;
    this.Pressure = pressure;
}
```

Отидете във файла **Car** и създайте частни полета и публични свойства за **Engine** и **Tire[]**.

Създайте друг конструктор, който приема **make**, **model**, **year**, **fuelQuantity**, **fuelConsumption**, **Engine** и **Tire[]** при инициализация:

```
public Car(string make, string model, int year, double fuelQuantity, double fuelConsumption,
    Engine engine, Tire[] tires)
    : this(make, model, year, fuelQuantity, fuelConsumption)
{
    this.Engine = engine;
    this.Tires = tires;
}
```

Трябва да може да използвате класовете по следния начин:

```

public static void Main(string[] args)
{
    var tires = new Tire[4]
    {
        new Tire(1, 2.5),
        new Tire(1, 2.1),
        new Tire(2, 0.5),
        new Tire(2, 2.3),
    };

    var engine = new Engine(560, 6300);

    var car = new Car("Lamborghini", "Urus", 2010, 250, 9, engine, tires);
}

```

```

public static void Main(string[] args)
{
    var tires = new Tire[4]
    {
        new Tire(1, 2.5),
        new Tire(1, 2.1),
        new Tire(2, 0.5),
        new Tire(2, 2.3),
    };

    var engine = new Engine(560, 6300);

    var car = new Car("Lamborghini", "Urus", 2010, 250, 9, engine, tires);
}

```

5. Специални коли

Това е последната и най-интересна част от тази задача. До получаване на команда **"No more tires"**, ще получавате информация за гуми в следния формат:

{year} {pressure}

{year} {pressure}

...

"No more tires"

От вас се изисква да съхранявате всички гуми. След това, до получаване на команда **"Engines done"**, ще получавате информация за двигател, която също трябва да съхраните.

{horsePower} {cubicCapacity}

{horsePower} {cubicCapacity}

...

"Engines done"

Финалната стъпка – до получаване на команда **"Show special"**, ще получавате информация за коли в следния формат:

```
{make} {model} {year} {fuelQuantity} {fuelConsumption} {engineIndex}
{tiresIndex}
```

...

Всеки път трябва да създадете нова кола (**new Car**) с информацията, която получавате. За двигателя на колата ще получавате **engineIndex**, а за гумите – **tiresIndex**. Когато получите команда **"Show special"**, намерете колите, които са **произведени през 2017 или по-късно**, имат **horsePower** над **330** и сумата от **налягането на гумите** (tire pressure) е между **9** и **10**. Всяка от тези коли изминава **дистанция от 20 км**. След това отпечатайте информация за всяка специална кола в следния формат:

```
"Make: {specialCar.Make}"
"Model: {specialCar.Model}"
"Year: {specialCar.Year}"
"HorsePowers: {specialCar.Engine.HorsePower}"
"FuelQuantity: {specialCar.FuelQuantity}"
```

Примери

| Вход | Изход |
|--|--|
| 2 2.6 3 1.6 2 3.6 3 1.6 1 3.3 2 1.6 5 2.4 1 3.2 No more tires 331 2.2 145 2.0 Engines done Audi A5 2017 200 12 0 0 BMW X5 2007 175 18 1 1 Show special | Make: Audi Model: A5 Year: 2017 HorsePowers: 331 FuelQuantity: 197.6 |

6. Служители

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

Дефинирайте **публичен клас Employee** в namespace **ClassesEmployee**.

Класът трябва да има **частни полета** за:

- **name:** стринг
- **age:** цяло число (int)

Класът трябва да има **публични свойства** за:

- **Name:** стринг
- **Age:** цяло число (int)

Създайте няколко обекта от тип **Employee**, като използвате следните данни:

| Name | Age |
|------|-----|
|------|-----|

| | |
|-------|----|
| Dan | 20 |
| Joey | 18 |
| Tommy | 43 |

7. Създаване на конструктори

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

Добавете 3 конструктора към класа **Employee** от предишната задача:

1. Първият не приема аргументи създава служител с име по подразбиране **"No name"** и възраст по подразбиране **age = 1**.
2. Вторият приема само един параметър – цяло число – за възрастта и създава служител с име по подразбиране **"No name"** и възраст, равна на подадения параметър.
3. Третият приема два параметъра – един стринг за името и едно цяло число за възрастта, след което създава нов служител със съответното име и възраст.

8. Най-възрастен служител

Използвайте **класа Employee** от предишните задачи. Създайте нов клас с име **Department**. Класът трябва да има:

- Списък със служители (**list of employees**)
- Метод за добавяне на служители (**void AddMember(Employee member)**)
- Метод за връщане на най-възрастния служител на департамента (**Employee GetOldest()**).

Напишете програма, която чете имената и възрастта на **N** членове и ги **добавя към департамента**. След това отпечатайте **името и възрастта** на най-възрастния член.

Примери

| Вход | Изход | Вход | Изход |
|---------------------------------------|-----------|--|----------|
| 3 Nick 23 Jorge 34 Sophie 55 | Sophie 55 | 5 Steve 29 Christopher 25 Annie 24 Brian 35 Nicole 24 | Brian 35 |

9. Анкета

Използвайки **класа Employee**, напишете програма, която чете от конзолата **N** реда информация за служители и след това отпечата тези, които са на **възраст повече от 30 години**, сортирани **по азбучен ред**.

Примери

| Вход | Изход |
|--|---|
| 3 Angela 22 Joshua 31 Connor 48 | Connor - 48 Joshua - 31 |
| 5 Molly 33 Peter 88 Paul 22 Johnny 44 Martin 21 | Johnny - 44 Molly - 33 Peter - 88 |

10. Състезатели във Формула 1

Напишете софтуер, който анализира информация за състезателите във Формула 1.

На първия ред ще получите число **N**, което показва колко реда с данни за състезателите ще получите. На следващите **N** реда ще получите информация за шофьорите.

Дефинирайте **клас Driver**. Всички шофьори трябва да имат:

- **Name:** стринг
- **Age:** цяло число (int)
- **TotalTime:** реално число (double)
- **Speed:** реално число (double)

Отпечатайте информацията за шофьора с **най-добро време** в следния формат:

DriverName: { Name }

DriverAge: { Age }

Time: { TotalTime }

Speed: { Speed }

Примери

| Вход | Изход |
|---|---|
| 3 Michael Schumacher 52 1.56 290 Lewis Hamilton 36 1.42 303 Sebastian Vettel 33 1.59 278 | DriverName: Lewis Hamilton DriverAge: 36 Time: 1.42 Speed: 303 |
| 5 Kevin Magnussen 28 1.59 240 Nikita Mazepin 22 1.53 257 Charles Pic 31 1.47 265 | DriverName: Charles Pic DriverAge: 31 Time: 1.47 Speed: 265 |

| | |
|---|--|
| Daniel Ricciardo 31 2.10 200 Nico Rosberg 35 1.50 260 | |
| 2 Alexander Rossi 29 1.37 273 Takuma Sato 44 1.54 262 | DriverName: Alexander Rossi DriverAge: 29 Time: 1.37 Speed: 273 |

11. Банкова сметка

Създайте програма, която съхранява информация за банковите сметки на служителите. Създайте нов клас с име **BankAccount**.

Класът трябва да съдържа следните членове:

AccountNumber: стринг

OwnerName: стринг

AccountBalance: дробно число (decimal)

Класът трябва да има метод **public void MakeDeposit**. Методът приема сума (**amount**), увеличава стойността на сметката (**balance**) и отпечатва като стринг **новата стойност на сметката** в следния формат:

- "Account balance: { the new account balance }"

Класът трябва да има метод **public void MakeWithdrawal**. Методът приема сума (**amount**) и проверява дали **има достатъчно средства**, които да бъдат изтеглени.

- В случай че **няма достатъчно средства**, отпечатайте "Non-Sufficient Funds"
- Ако в банковата сметка има достатъчно голяма сума, извадете **amount** от **текущата сума** в сметката и отпечатайте **оставащите средства** в следния формат:
"Withdrawn funds: { amount of funds withdrawn}. Funds available on the account: { the amount of funds available }"

На **първия ред** ще получите **информация за клиентите**, разделена с интервал. Всеки **номер на сметка** ще бъде последван от **името** на притежателя и от **наличната сума** в сметката.

До достигане на команда "End" ще получавате команди за депозит (**MakeDeposit**) или изтегляне (**MakeWithdrawal**).

Няма да получите невалидни данни или отрицателни числа.

Примери

| Вход | Изход |
|--|---|
| 3543653456 Jorge Cooper 587 Deposit 200 Withdrawal 100 End | Account balance: 787 Withdrawn funds: 100. Funds available on the account: 687 |
| 3963185629 Peter Davis 692 Withdrawal 700 Deposit 8 Withdrawal 40 Withdrawal 660 Withdrawal 200 | Non-Sufficient Funds Account balance: 700 Withdrawn funds: 40. Funds available on the account: 660 Withdrawn funds: 660. Funds available on the account: 0 Non-Sufficient Funds |

12. Антикварна книжарница

Ваш приятел ще отваря антикварна книжарница и ви е помолил за помощ. Искате да напишете програма, която съхранява информация за книгите в книжарницата. Програмата трябва да има клас с име **Book**, който има следните членове:

Title: стринг

Author: стринг

PubDate: цяло число (int) – дата на публикуване

Price: дробно число (decimal)

Publisher: стринг

Discount: стринг

В програмата трябва да могат да се добавят книги **без publisher (издател) и discount (отстъпка)**. В тези случаи, стойността по подразбиране за издателя трябва да е **"Unknown"**, а тази за отстъпката - **"Unavailable"**. Може да има книга **със издател**, но **без отстъпка**, но няма да има книга **без издател и със отстъпка**.

До получаване на команда **"End"** ще получавате информация за книги, разделени с **" , "**. Първо е написано **заглавието (title)**, последвано от **author, publication date, price** и **опционален publisher и discount**.

След получаване на команда **"End"**, отпечатайте всички книги, **сортирани по име на автора във възходящ ред** в следния формат:

Title : {book title}

Author: {book author}

Publication Date: {PubDate}

Price: {book price}

Publisher: {publisher}

Discount: {discount}

Примери

| Вход | Изход |
|---|--|
| The Years, Virginia Woolf, 1937, 195, Hogarth Press, 10% The Lure of the Mask, Harold MacGrath, 1908, 200 Ask the Dust, John Fante, 1939, 185, Stackpole Sons, 15% End | Title : The Lure of the Mask Author: Harold MacGrath Publication Date: 1908 Price: 200 Publisher: Unknown Discount: Unavailable Title : Ask the Dust Author: John Fante Publication Date: 1939 Price: 185 Publisher: Stackpole Sons Discount: 15% |

| | |
|---|---|
| | <p>Title : The Years Author: Virginia Woolf Publication Date: 1937 Price: 195 Publisher: Hogarth Press Discount: 10%</p> |
| <p>Star Maker, Olaf Stapledon, 1937, 215, Methuen & Co Foundation, Isaac Asimov, 1951, 199, Gnome, 7% Stuart Little, E.B. White, 1945, 259, Harper and Brothers, 5% Moonraker, Ian Fleming, 1955, 199 End</p> | <p>Title : Stuart Little Author: E.B. White Publication Date: 1945 Price: 259 Publisher: Harper and Brothers Discount: 5%</p> <p>Title : Moonraker Author: Ian Fleming Publication Date: 1955 Price: 199 Publisher: Unknown Discount: Unavailable</p> <p>Title : Foundation Author: Isaac Asimov Publication Date: 1951 Price: 199 Publisher: Gnome Discount: 7%</p> <p>Title : Star Maker Author: Olaf Stapledon Publication Date: 1937 Price: 215 Publisher: Methuen & Co Discount: Unavailable</p> |