Упражнения: Класове и обекти

1. Клас Човек

Дефинирайте клас **Person** с **public** полета **name** (име) и **age** (възраст).

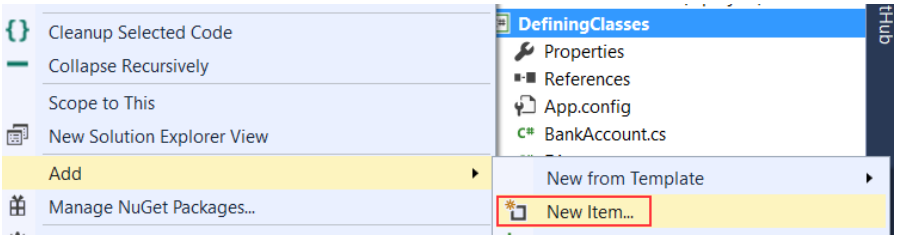
Бонус*

Опитайте се да създадете няколко обекта от Person:

Име	Възраст
Pesho	20
Gosho	18
Stamat	43

Решение

1. Създайте **нов клас**: от меню [Project] → [Add Class] или на десен бутон върху проекта [Add]→ [New Item] → [Class]



2. Подсигурете се, че **сте избрали подходящи имена** за класа и неговите елементи.

3. **Дефинирайте класа** във файла **Person.cs**:

```
public class Person
{
    public string name;
    public int age;
}
```

4. В **Program.cs**, в Main метода на класа Program **създайте инстанции на класа**:

```
0 references
class Program
{
    0 references
    static void Main(string[] args)
    {
        Person p1 = new Person();
        p1.name = "Pesho";
        p1.age = 20;

        Person p2 = new Person();
        p2.name = "Gosho";
        p2.age = 18;
    }
}
```

5. Създайте по същия начин **обект** и за **Stamat.**

2. Домашни любимци

Дефинирайте клас **Pet** с **public** полета **name** (име), **type** (вид) и **weight** (тегло).

Бонус*

Опитайте се да създадете няколко обекта от Pet:

Име	Вид	Тегло
Sharo	dog	15
Pisana	cat	3.5
Nemo	fish	0.02

3. Семейство

Дефинирайте клас **Family**. В него пазете информация за **майката** и **бащата** (от тип **Person**) и списък на техните **деца** (също от тип **Person**). После създайте обекти от този клас с данните от примерите.

Примери

Данни	Изход
Pesho 25 Annie 22 Goshko 4	Father: Pesho 25 Mother: Annie 22 Children: 1) Goshko 4

Данни	Изход
Steve 34 Maria 33 Christopher 8 Annie 4 Ivan 2	Father: Steve 34 Mother: Maria 33 Children: 1) Christopher 8 2) Annie 4 3) Ivan 2

Бонус*

Опитайте се да създадете отпечатате информацията за създадения обект за класа Family

Решение

1. Първо си **дефинирайте клас Person**, по същия начин, както направихте това в предната задача.
2. **Създайте нов клас Family**. Той ще съдържа първо две полета за **майката** и **бащата**. Те трябва да са от тип **Person**. За да можем да записваме информацията в тях, трябва да **създадем екземпляри от класа**:

```
class Family
{
    public Person father = new Person();
    public Person mother = new Person();
```

3. Сега трябва да създадем поле, което ще съхранява информацията за **децата**. Понеже различните семейства имат различен брой деца, няма как да използваме отделни променливи от тип Person. Вместо това ще създадем нов обект от тип **списък** (**List**) от **хора** (**Person**) - както създаваме обекти:

```
public List<Person> children = new List<Person>();
```

4. С това дефиницията на класа Family е готова.

5. В **Program.cs**, в Main метода на класа Program трябва да **създадем инстанция на класа Family**:

```
static void Main(string[] args)
{
    Family myFamily = new Family();
```

Така вече имаме къде да записваме информация и за родителите, и за децата от това семейство.

6. Понеже обектите за майката и бащата са вече създадени при дефинирането на класа във Family.cs, то в Main() метода можем просто да им попълним данните:

```
myFamily.father.name = "Pesho";
myFamily.father.age = 25;

myFamily.mother.name = "Annie";
myFamily.mother.age = 22;
```

Тъй като имаме обект вътре в друг обект (обектът **баща**, който е част от това **семейство**), използваме . (точка) за да достигнем до неговите полета: `myFamily.father.name = "Pesho";` Може да го прочетете така: в семейството **myFamily** на бащата (**father**) името му (**name**) е **"Pesho"**.

7. За децата е малко по-различно: имаме създаден само списък с децата, но не и конкретните обекти, съхраняващи информацията за всяко дете. Така че тук трябва да минем през три стъпки:
   a. създаване на обект за детето
   b. попълване на информацията му
   c. добавяне на това дете към списъка с деца за това семейство

8. Програмният код за семейство с едно дете би трябвало да изглежда така:

```
static void Main(string[] args)
{
    Family myFamily = new Family();

    myFamily.father.name = "Pesho";
    myFamily.father.age = 25;

    myFamily.mother.name = "Annie";
    myFamily.mother.age = 22;

    Person child1 = new Person();
    child1.name = "Goshko";
    child1.age = 4;

    myFamily.children.Add(child1);
}
```

9. Ако решим да отпечатваме информацията за семейството, обръщаме внимание, че информацията за родителите и децата е в един и същи формат, само текстът отпред е различен. Следователно може да добавим метод за извеждането ѝ в класа Person:

```
public void IntroduceMyself(string text)
{
    Console.WriteLine(text + name + " " + age);
}
```

10. И накрая може да добавим метод IntroduceFamily() в класа Family, който да прави извеждането на информацията на всички членове на семейството. За извеждане на децата използваме цикъл for:

```
public void IntroduceFamily()
{
    father.IntroduceMyself("Father: ");
    mother.IntroduceMyself("Mother: ");
    Console.WriteLine();
    for(int i = 0; i < children.Count; i++)
    {
        children[i].IntroduceMyself((i + 1) + ") ");
    }
}
```

4. Статистика

Използвайки класът **Person**, напишете програма, която въвежда от конзолата **N** реда информация за хора и отпечатва хората на възраст **по-голяма от 30** години.

Примери

Вход	Изход
3 Pesho 12 Stamat 31 Ivan 48	Stamat – 31 Ivan - 48
5 Nikolai 33 Yordan 88 Tosho 22 Lyubo 44 Stanislav 11	Yordan - 88 Nikolai - 33 Lyubo - 44

Подсказка

Създайте **списък** (**List**) от **хора** (**Person**) за съхраняването на информацията за всички хора, които са над 30. После създайте един обект от тип Person, прочетете в него информацията за един човек и ако той е над 30, го запомнете в списъка. Можете да ползвате една и съща променлива, тъй като тя съдържа само референция към обекта, а не самия обект. Така че няма опасност информацията на следващия човек да бъде презаписана.

5. Рационални числа

Дефинирайте клас **RacionalNumber** с **public** полета **numerator**(числител) и **denumerator**(знаменател). Създайте 3 обекта и ги изведете във формат “Numerator/denumerator {numerator}/{denumaerator}”.

Примери

Вход	Изход
3 4 5 6 7 8	3/4 5/6 7/8

Забележка:

За задачите с дроби да се спазва общото ограничение, че знаменателя винаги е естествено число. Т.е. не може да е 0 или отрицателно число.

6. Рационални числа, на един ред *

Дефинирайте клас **RacionalNumber** с **public** полета **numerator**(числител) и **denumerator**(знаменател). Въведете ги на един ред, разделени с интервал и ги изведете във формат “Numerator/denumerator {numerator}/{denumaerator}”, на един ред, разделени със “; “

Примери

Вход	Изход
3 4 5 6 7 8	3/4; 5/6; 7/8