

## Week 3 assignments

When creating the program code, you must apply the following basic principles:

- create a separate project for each assignment;
- use name 'assignment1', 'assignment2', etcetera for the projects;
- create one solution for each week containing the projects for that week;
- make sure the output of your programs are the same as the given screenshots;

**Note:** for assignment 1 and 2, your output must contain a dot (.) as a decimal separator, and not a comma (,), see screenshots on the next page. To make sure your program uses a dot, add the following code to your program:

```
using System;
using System.Globalization;
using System.Threading;

void Start()
{
    // set culture of program
    CultureInfo ci = new CultureInfo("en-US");
    Thread.CurrentThread.CurrentUICulture = ci;
    Thread.CurrentThread.CurrentCulture = ci;

    // your code here...
}
```


## CodeGrade auto checks

Make sure all CodeGrade auto checks pass (10/10) for your assignments. The manual check will be done by the practical teacher.

Auto checks assignment 1-<sup>10</sup>/<sub>10</sub> **AT**

Auto checks assignment 2-<sup>10</sup>/<sub>10</sub> **AT**

Manual check


Automatic checks for assignment 1 


0	10	10
	100	%

Submit

6.67

20 / 30

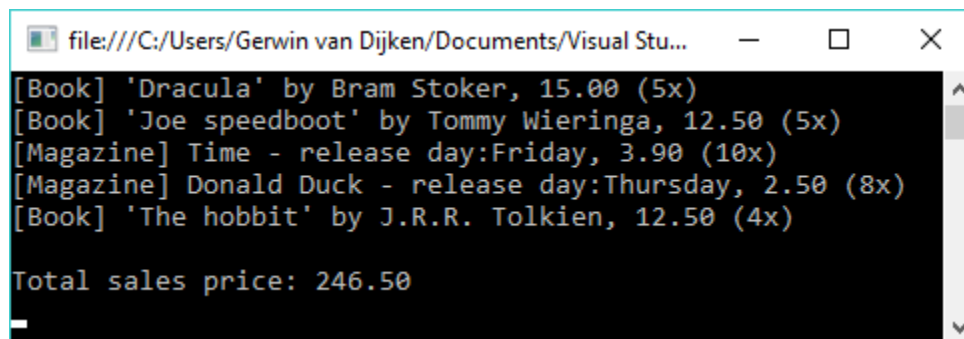




## Assignment 1 – Bookshop revisited

We will revisit the Bookshop of last week, improving it by using properties (instead of public members).

- Modify (abstract) class `BookStoreItem` by converting the public members (title and price) into **properties**.
- Add an extra property 'Count' that contains the number of copies of a bookstore item. Also add a **calculated** property 'TotalPrice', that returns the total sales price of all available copies.
- Modify the classes `Book` and `Magazine` by converting the public members into properties. Property `Author` (class `Book`) must be a **read only** property, since the author of a book does not change.
- Modify method 'PrintCompleteStock' of class `BookStore`: instead of using the price of each item, use the 'total price' (of all copies of an item). Furthermore, make sure the list with items is **private**, we don't want the list to be accessible by other classes.
- In the `Start` method, create some books and magazines, and add these to the bookstore. You can use the screenshot below, to check your own program (pay special attention to the total price of the complete stock).



```
file:///C:/Users/Gerwin van Dijken/Documents/Visual Stu...  
[Book] 'Dracula' by Bram Stoker, 15.00 (5x)  
[Book] 'Joe speedboot' by Tommy Wieringa, 12.50 (5x)  
[Magazine] Time - release day:Friday, 3.90 (10x)  
[Magazine] Donald Duck - release day:Thursday, 2.50 (8x)  
[Book] 'The hobbit' by J.R.R. Tolkien, 12.50 (4x)  
  
Total sales price: 246.50
```

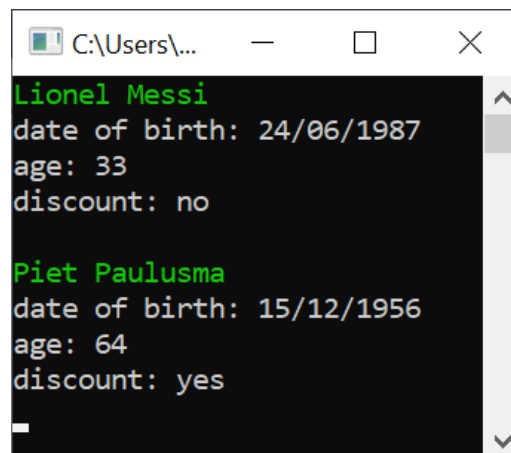
## Assignment 2 - Cinema

In this assignment we will create some classes for a reservation system, for booking cinema tickets. We don't implement a complete system, only the classes Customer, Ticket and Reservation.

### class Customer

Of each customer, we want to record the name and age. Customers with age 60 or older get a discount.

- Create a **property** for name, allowing only valid values (*name cannot be an empty string*).
- Since the age of a customer will of course change, we will not store the age itself, but the date of birth. Create a **property** DateOfBirth, allowing only valid values (*date of birth must be in the past*).
- Add a **constructor** with 2 parameters: name and date of birth.
- Create a **calculated** (int) property Age that uses the date of birth. Use DateTime.Today to get the current date.
- Create a **calculated** (bool) property Discount that returns true if the customer is at least 60 years old.
- In the Start method, check if all properties work correctly for several customers. Some customers will have had their birthday already this year, other customer will have it later this year. Also check incorrect input (like an empty name, or a 'future' date of birth).
- In class Program, create a separate method PrintCustomer (with a Customer parameter) to print a customer. You can see an example of the output below.



```
C:\Users\...  
Lionel Messi  
date of birth: 24/06/1987  
age: 33  
discount: no  
  
Piet Paulusma  
date of birth: 15/12/1956  
age: 64  
discount: yes  
_
```

### class Ticket

Class Ticket has the following properties: (string) MovieName, (int) CinemaRoom, (DateTime) StartTime, (decimal) Price, (int) MinimumAge.

- a) Create the **properties** and make sure that the following restrictions are met (*test this in the set-part of each property, throw an exception if an invalid value is used*):
  - the name of the movie is not empty;
  - there are 5 cinema rooms: 1, 2, 3, 4 and 5;
  - each movie starts on the hour or on the half hour;
  - the minimum age can be one of the following values: 0, 6, 9, 12, 16, where 0 means 'for all ages';
- b) Add a **constructor** with 2 parameters: movie name and price.
- c) Create a **calculated** property Discount that indicates if a discount will be given for the ticket. A discount is given for all movies on a Monday or Tuesday. (*use property DayOfWeek of class DateTime*)
- d) In the Start method, check if all properties work correctly for several tickets. Check the following cases: an empty movie name, an invalid cinema room, and invalid minimum age and an invalid start time.

```
C:\Users\gerwin.vandijken...
creating tickets
Error occured: Empty movie name!
```

```
C:\Users\gerwin.vandijken...
creating tickets
Error occured: Invalid cinema room (7)!
```

```
C:\Users\gerwin.vandijken\...
creating tickets
Error occured: Invalid minimum age (17)!
```

```
C:\Users\gerwin.vandijken\OneDrive - Hogeschool ...
creating tickets
Error occured: Invalid start time! (13-2-2021 19:15:00)
```

### class Reservation

Class Reservation is linked to a customer, and can have several tickets.

- Create **property** Customer (the customer of the reservation) and a property Tickets (a list containing all tickets of the reservation); tickets can be added to this list;
- Add a **constructor** with 1 parameter: the customer of the reservation.
- Create a **calculated** property TotalPrice, that returns the total price of all tickets in the list. This property checks the minimum age, and uses the discount of the tickets and the discount of the customer. The discount for a ticket is 5%. The discount for customers (60 years or older) is 10% and is given on the total price (of all tickets);
- In the Start method, create some reservations and check if the total price is calculated correctly. An example of the output is given below.  
*(the movie 'Green Book' is on a Monday: February 15th 2021, so a discount of 5% is applied; Piet Paulusma is 64 years old, so a discount of 10% is applied for the complete reservation)*

```
C:\Users\gerwin.vandijken\OneDrive - Hogeschool Inholland\Werk\Inholland\VS-projects\VS-2015-...
Lionel Messi
date of birth: 24/06/1987
age: 33
discount: no

Piet Paulusma
date of birth: 15/12/1956
age: 64
discount: yes

creating tickets (for Lionel Messi)
created ticket 'Bohemian Rhapsody', start time: 13/02/2021 21:30, price: 10.50, room: 1 (6+)
created ticket 'The Prodigy', start time: 13/02/2021 22:00, price: 10.50, room: 4 (16+)
created ticket 'Green Book', start time: 15/02/2021 19:00, price: 10.50, room: 5 (12+)
total price of reservation: 30.98

creating tickets (for Piet Paulusma)
created ticket 'Bohemian Rhapsody', start time: 13/02/2021 21:30, price: 10.50, room: 1 (6+)
created ticket 'The Prodigy', start time: 13/02/2021 22:00, price: 10.50, room: 4 (16+)
created ticket 'Green Book', start time: 15/02/2021 19:00, price: 10.50, room: 5 (12+)
total price of reservation: 27.88
```

