Practice Assignments Enumerations and Constants

Quiz questions, practical assignments and

answers to quiz questions

**Version:** 0.1 **Last updated:** 20 september 2023

Contents

[1 Quiz: Classes & Objects 3](#_Toc145421074)

[1.1 Question 1 3](#_Toc145421075)

[1.2 Question 2 3](#_Toc145421076)

[1.3 Question 3 3](#_Toc145421077)

[1.4 Question 4 4](#_Toc145421078)

[2 Practical assignments: Classes & Objects 5](#_Toc145421079)

[2.1 Programming Assignment 1: Pizza toppings 5](#_Toc145421080)

[2.1.1 Case description 5](#_Toc145421081)

[2.2 Programming Assignment 2: Planet Information 6](#_Toc145421082)

[2.2.1 Case description 6](#_Toc145421083)

[2.3 Programming Assignment 3: Constant-keyword 7](#_Toc145421084)

[2.3.1 Case description 7](#_Toc145421085)

[2.4 Programming Assignment 4: Streaming Music Application 8](#_Toc145421086)

[2.4.1 Case description 8](#_Toc145421087)

[2.4.2 Provided material 9](#_Toc145421088)

[2.4.3 Additional features 9](#_Toc145421089)

[3 Quiz answers 10](#_Toc145421090)

# Quiz: Classes & Objects

Answers to the quiz-questions can be found in the last chapter.

## Question 1

We have the following declaration of the enum *Month*:

enum Month

{JAN, FEBR, MRCH, APR, MAY, JUN, JUL, AUG, SEPT, OCT, NOV, DEC }

Consider the following piece of code:

Month m1, m2;

m1 = Month.APR;

m2 = m1;

m1++;

int k;

k = (int)m2;

What are the values of m1, m2 and k after performing this piece of code?

## Question 2

Again, the enum *Month* as declared in question 1, and we also have the enum *Prox*:

enum Prox

{NACL, JUN, HOOI, DEC, OCT, FEO, JAN }

Consider the following piece of code:

Month m;

m = Month.JUN;

Prox p;

p = Prox.JUN;

int i = (int)m + (int)p;

Prox q= (Prox)((int)m);

If the code executes correctly, what are the values of i and q after performing this piece of code?

## Question 3

Consider the following piece of code:

enum Moments {century, year, month, day, while, minute, if }

Visual Studio does not accept the above declaration. Why not? Note that the slides use uppercase for enum-values, it is not wrong to use lower case.

## Question 4

Consider the following piece of code:

01 class SavingsAccount {

02 private const decimal INTEREST\_RATE = 0.01;  
03 private decimal balance;

04 public SavingsAccount(decimal balance, decimal interestRate) {

05 INTEREST\_RATE = interestRate;

06 this.balance = balance;

07 }  
08

09 public void ApplyInterest() {

10 balance \*= 1 + SavingsAccount.INTEREST\_RATE;

11 }

12 }

13  
14 MessageBox.Show($"Current interest rate is {SavingsAccount.INTEREST\_RATE}");

Which of the line(s) would give a syntax error and why?

# Practical assignments: Classes & Objects

## Programming Assignment 1: Pizza toppings

Difficulty: C:\Users\874156\Desktop\flatastic-icons-part-1-by-custom-icon-design\png\16x16\star-2_5.png.

The assignment covers the following learning goals:

* Apply enum

### Case description

For this assignment, you are tasked to create an app to allow a user to order a pizza with several kinds of topping. Create a simple application that let you choose toppings, show what toppings you have chosen already and what the price will be.

The price of a pizza is 5,00 euro + 50 cents for each topping. It is allowed to have a certain topping more than once.

Use an enum for the toppings. Examples of toppings are: cheese, tomatoes, mushrooms, egg, ham, onions, anchovies, ham, peppers.

An example of the user interface could be:

Graphical user interface, text, application

Description automatically generated

Have a combobox for the kind of toppings the user can chose.

*Hint: have them in the combobox in the same order as in the enum. If the user chooses a topping in the combobox, you can use the property SelectedIndex of the combobox to easily find which value of the enum it concerns*.

Use a collection (array or list) to store the chosen toppings. Of course, the elements of this collection should be of your enum-type.

*The above screen dump: the user is a ham-lover. She had already chosen 2 times ham and now she has selected ham in the combobox.*

*Clicking the add-button should add ham to the toppings of the pizza. Clicking the remove-button should remove ham.*

*Do not forget to show the chosen toppings and the adjusted price on the screen.*

## Programming Assignment 2: Planet Information

Difficulty: C:\Users\874156\Desktop\flatastic-icons-part-1-by-custom-icon-design\png\16x16\star-2_5.png.

The assignment covers the following learning goals:

* Apply enum
* Apply switch/if statement

### Case description

Create a simple application which shows information (for instance the gravity) of a planet in our solar system. It is required to use an enum type and also try to use the switch statement.

An example of the user interface would be:

Graphical user interface, application

Description automatically generated

## Programming Assignment 3: Constant-keyword

Difficulty: C:\Users\874156\Desktop\flatastic-icons-part-1-by-custom-icon-design\png\16x16\star-2_5.png.

The assignment covers the following learning goals:

* Apply constant
* Apply classes

### Case description

The following code is given:

public class PrintLog

{

private int pagesPrinted;

public PrintLog()

{ this.pagesPrinted = 0; }

// Increase count of printed pages and return the print cost

public double PrintPage()

{

this.pagesPrinted++;

return 0.05;

}

// Increase count of printed pages and return the print cost

public double PrintPage(int pageCount)

{

this.pagesPrinted += pageCount;

return 0.05 \* pageCount;

}

// Calculate total revenue based on total printed pages

public double GetCurrentRevenue()

{ return 0.05 \* this.pagesPrinted; }

}

Can you refactor (change) the code in such a way that it makes use of a constant for the value *0.05*? This value represents the cost for printing one page. Can you think of why the usage of a constant is usefull here? HINT: What would you need to change now compared to the given code when the printing price should be increased to *0.07*?

## Programming Assignment 4: Streaming Music Application

C:\Users\874156\Desktop\flatastic-icons-part-1-by-custom-icon-design\png\16x16\star-5_5.pngDifficulty:

The assignment covers the following learning goals:

* You implement an application by interpreting a UML class diagram

### Case description

You are tasked to complete an application managing songs of a streaming music service. A start-up project is provided via Canvas with the *MusicStreamingService* and *Song* classes implemented.

The supplied project is missing the *Genre* enum, a *User* class and the controls in the form to allow a user to manage the songs and users.

Based on *UML Class Diagram 4: Complete UML Class Diagram*, you will have to extend the start-up application with the following:

* Implement *Genre* enum and refactor the existing classes to make use of it:
  + The *Song* class should have an instance variable for a *Genre*; this requires you to also modify the constructors
  + The *MusicStreamingService* class also has to be changed to reflect the refactored *Song* class; can you determine what and can you implement the required changes?
* Implement the *User* class:
  + Determine for yourself what instance variables should be declared. HINT: how should you interpret the association to the *Song*-class?
  + Implement the constructors
  + *AddSongToFavorites()* should add, based on the value of the parameters, to the favorite list. Note that the amount of favorite songs can not exceed *MAX\_SONGS\_IN\_FAVORITES*.
  + *RemoveSongFromFavorites()* should remove, based on the value of the parameter, from the favorite list
  + *GetFavoriteSongs()* should return a string with all the songs bookmarked as favorite. For example as:  
    Favourite songs of John Doe:  
    - ID 01: Queen - Bohemian Rhapsode (354 seconds)  
    - ID 04: Queen – Don’t Stop Me Now (218 seconds)  
    or  
    Jane Jackson has no favorite songs
* Complete the *StreamingMusicService* class:
  + Determine for yourself what instance variable should be declared
  + *AddUser()* should create and add a *User*-object based on the value of the parameters
  + *GetUser()* should return the correct *User-object*, if any, based on the value of the parameter
  + *GetUsers()* should return all the *User*-objects as an array
  + *GetInfo()* should be modified to also include the number of users. For example:  
    Streaming Music service: YoMusicfy (5 songs & 2 users)
* Create the GUI by adding controls by using a TabControl among other controls to:
  + View all song(s)
  + Add a new song; make sure the text in the title bar is updated with new info
  + View all users(s)
  + Add a new user; make sure the text in the title bar is updated with new info
  + Add a song as favourite to a user

### Provided material

In Canvas the file *S-Week13\_StreamingMusicService\_StartUp.zip* contain the Visual Studio start-up project.

Diagram

Description automatically generated

UML Class Diagram 1: Incomplete UML Class Diagram

### Additional features

Modify the *GetInfo()*-method in class *Song* to show the duration as *<minutes>:<seconds>*. For example:   
 ID 01: Queen - Bohemian Rhapsode (5:54)

Can you refactor the application by including a *Playlist*-class. This class should make it possible to allow users to have one or more playlists. Make the application work with this new class to, for example, modify how a song is added as favorite (i.e. to the appropriate playlist), add playlist to a user, etc.

# Quiz answers

|  |  |
| --- | --- |
| Question | Answer |
| 1 | m1: Month.MAY,  m2: Month.APR ,  k : 3 |
| 2 | k: 6 ( = 5 + 1 )  q : Prox.FEO |
| 3 | The values "while" and "if" are reserved words. |
| 4 | 01 class SavingsAccount {  02 private const decimal INTEREST\_RATE = 0.01; 03 private decimal balance;  04 public SavingsAccount(decimal balance, decimal interestRate) {  **05 INTEREST\_RATE = interestRate;**  06 this.balance = balance;  07 } 08  09 public void ApplyInterest() {  10 balance \*= 1 + SavingsAccount.INTEREST\_RATE;  11 }  12 }  13 **14 MessageBox.Show($"Current interest rate is {SavingsAccount.INTEREST\_RATE}");**  **Line 5**: You cannot assign a new value to a constant. Note that the constants is used within the class so the inclusion SavingsAccount. is not required.  **Line 14**: The constant is declared as private and cannot be used ‘outside’ of the class *SavingsAccount*. If you want it to be accessible, you can declare the constant with the public-access modifier. |