Practice Assignments Constructors and UML

Quiz questions, practical assignments and

answers to quiz questions

**Version:** 0.1 **Last updated:** 15 november 2023

Contents

[1 Quiz: Classes & Objects 3](#_Toc147863317)

[1.1 Question 1 3](#_Toc147863318)

[1.1 Question 2 3](#_Toc147863319)

[1.2 Question 3 4](#_Toc147863320)

[2 Practical assignments: Classes & Objects 5](#_Toc147863321)

[2.1 Programming Assignment 1: Extending the Course application 5](#_Toc147863322)

[2.1.1 Case description 5](#_Toc147863323)

[2.2 Programming Assignment 2: Refactor assignment about finances 6](#_Toc147863324)

[2.2.1 Case description 6](#_Toc147863325)

[2.3 Programming Assignment 3: Snackbar 7](#_Toc147863326)

[2.3.1 Case description 7](#_Toc147863327)

[2.3.2 Provided material 7](#_Toc147863328)

[2.3.3 Additional features 8](#_Toc147863329)

[2.4 Programming Assignment 4: UML Class Diagram 9](#_Toc147863330)

[2.4.1 Case description 9](#_Toc147863331)

[2.4.2 Provided material 9](#_Toc147863332)

[3 Quiz answers 10](#_Toc147863333)

# Quiz: Classes & Objects

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

## Question 1

How many constructors does the class *Buzzer* have in the following implementation? You may assume the 3 dots stands for correct code.

class Buzzer

{

private int nrOfCurls;

private double pz;

public void Buzzer(int n, double value)

{

. . .

}

public void Buzzer(int n)

{

. . .

}

public string getInfo()

{

. . .

}

public void addCurls(int value)

{

. . .

}

}

## Question 2

Are the constructors correctly overloaded and why? You may assume the 3 dots stands for correct code.

class Buzzer

{

private int nrOfCurls;   
 private double pz;

public Buzzer(int n, double value)

{

. . .

}

public Buzzer(int nrOfCurls, double pzvalue)

{

. . .

}

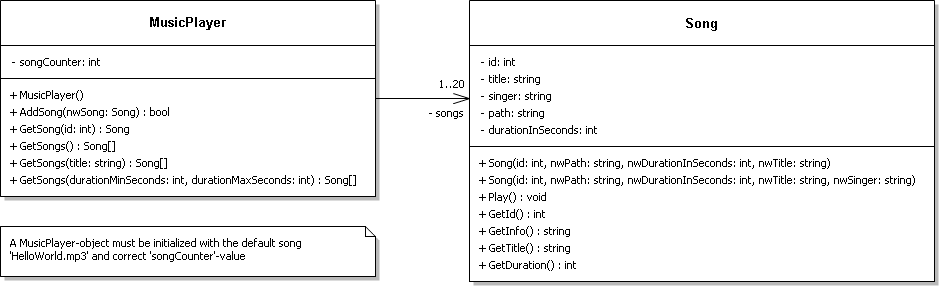
. . .

}

## Question 3

|  |  |
| --- | --- |
| C:\Users\874156\Desktop\flatastic-icons-part-1-by-custom-icon-design\png\48x48\alert.png | This quiz questions should be done after part 3 in the slides, where UML Class Diagrams are covered, or when you want to see what you current understanding related to UML Class Diagrams is. |

Answer the following question while considering the following UML Class Diagram:



1. How many typed methods does class *Song* have?
2. How many public instance variables does class *Song* have?
3. How would you implement the association between class *MusicPlayer* and *Song*?
4. Which method(s) are overloaded?
5. Does class *MusicPlayer* have a default constructor?

# Practical assignments: Classes & Objects

## Programming Assignment 1: Extending the Course application

*(Assignment 2 of ‘Intro Objects and Classes’-lesson)*

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

The assignment covers the following learning goals:

* You can define and use constructors for a class
* You know how to apply overloading in a class

### Case description

In the exercise for week 10 day 3, page 6 ‘Intro Object & Classes’, you created a simple application where Course-object is created (see Figure 1: GUI of the course assignment).

Graphical user interface, text, application

Description automatically generated

Figure 1: GUI of the course assignment

Open this project and modify the *Course* to have two constructors:

* *First constructor*: has two parameters to initialize the object with values for the two instance variables.
* *Second constructor*: has one parameter to initialize the object with a value for the instance variable *name* and the instance variable *ec* should have a default value 3.

Then change your GUI to make use of the newly created constructors by require a user to supply the *Course name* and having the *Course ECs* as optional value.

## Programming Assignment 2: Refactor assignment about finances

*(Assignment 3 of lesson ‘List, Comboboxes and TabControl’)*

C:\Users\874156\Desktop\flatastic-icons-part-1-by-custom-icon-design\png\16x16\star-3_5.pngDifficulty:

The assignment covers the following learning goals:

* You can implement an application with multiple classes

### Case description

In the lesson for week 11 day 1 ‘List, Comboboxes and TabControl’, you created an application that allows a user to manage his/her finances (see Figure 2: The GUI of the finance application). This assignment existed out of one class *Expense* and a form containing a List<*Expense*>.

Graphical user interface, text, application

Description automatically generated

Figure 2: The GUI of the finance application

You are now tasked to refactor this application by introducing a class *FinancialManager*. The responsibility of this class is to manage all the transactions that the user enters. Note that it is good practice to include a *FinancialManager-*class and moving the appropriate code from the form to that class. This will separate the concerns and make your code better testable; these topics are covered in semester 2

By doing this you are separating the responsibilities:

* *Expense*-class: Store and returning information about an expense
* *FinancialManager*-class: Stores all the expenses as *Expense*-objects by managing the actions possible with the expenses.
* *Form1*-class: Responsible for processing all user interaction (GUI stuff) by calling the correct methods in the *FinancialManager*-class/object.

Try to determine what members *FinancialManager*-class should have and how the form can use this class as an object. Hint: you will need to move code from form to this class by making a distinction between GUI and business logic related code.

## Programming Assignment 3: Snackbar

C:\Users\874156\Desktop\flatastic-icons-part-1-by-custom-icon-design\png\16x16\star-4_5.pngDifficulty:

The assignment covers the following learning goals:

* You can define and use constructors for a class
* You know how to apply overloading in a class
* You can implement an application with multiple classes

You are asked to develop some software for the snackbar “MacSnack”. The application should be able to display information about snacks and process orders for some snacks.

### Case description

The snackbar "MacSnack" offers 3 kinds of snacks. There should be a possibility to indicate how many of each snack a client wants to order. For example: 3 of the first snack, 0 of the second snack and 4 of the third snack.   
After filling in these numbers, the user can click a button to order the snacks. Two things can happen:

1. The application displays the total amount of money the client must pay for the order. Be sure to also update the amount in stock of each snack and revenue[[1]](#footnote-1) accordingly.
2. The application displays a message indicating the order is not possible because not enough snacks are in stock to process the order. To keep it simple, an order is processed as a whole so either all ordered snacks are process or not at all.

The snackbar owner should also have the possibility, for example by clicking a button, to see what the current revenue is.

Create a windows form application and the required classes. Make use of the description provided in *Provided material* for the classes*.*Try to determine what controls your form needs and which instance variables are required to perform the actions (i.e. method) in a class.

### Provided material

|  |  |
| --- | --- |
| Class **Snack** | |
| *CONSTRUCTOR* | Should initialize the *Snack*-object with a *name*, *price* and the *amount in stock* |
| *METHOD* public bool DecreaseStock(int amount) | This method should decrease the *amount in stock* by a certain *amount*.  The boolean to be returned indicates whether it was possible to decrease the stock (i.e. amount in stock is high enough). |
| *METHOD* public string GetName() | This is a getter method returning the *name* of the snack. |
| *METHOD* public string GetPrice() | This is a getter method returning the *price* of the snack. |
| *METHOD* public string GetAmountInStock() | This is a getter method returning the *amount in stock* of the snack. |

|  |  |
| --- | --- |
| Class **SnackBar** | |
| *CONSTRUCTOR* | Should initialize the *SnackBar*-object with three *Snack*-objects; you choose which snacks and the objects can be ‘hardcoded’ in this constructor |
| *METHOD* public double ProcessOrder(int snack1, int snack2, int snack3) | This method processes the order by decreasing the quantity of the three snack based on the int-value.  The double to be returned should be the total price of the order or, when one of the snacks did not have enough stock, *-1* indicating the order was unsuccessful (i.e. not enough stock for one or more snacks). |
| *METHOD* public double GetRevenue() | This is a getter method returning the *revenue* (i.e. all the total price of the processed orders). |

### Additional features

Firstly, extend your application to also show which snack(s) does not have enough stock when an order cannot be processed.

Secondly, add the functionality to change the price and stock of a snack. For instance, have 3 RadioButtons to indicate for which snack the price and number in stock should change.

Lastly, can you make the required changes so it is possible to add new snacks instead of having three hardcoded in the constructor of *SnackBar.*

## Programming Assignment 4: UML Class Diagram

|  |  |
| --- | --- |
| C:\Users\874156\Desktop\flatastic-icons-part-1-by-custom-icon-design\png\48x48\alert.png | This question should be done after part 3 in the slides, where UML Class Diagrams are covered, or when you want to see what you current understanding is related to UML Class Diagrams. |

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

The assignment covers the following learning goals:

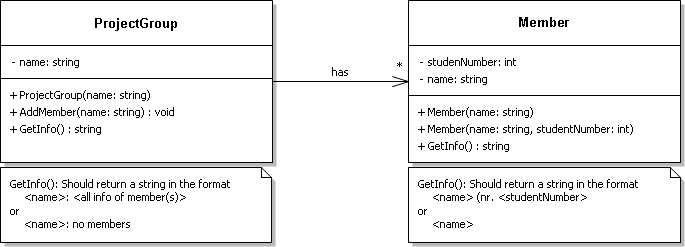
* You implement an application by interpreting a UML class diagram.

### Case description

In this assignment you are going to work with two classes to represent a project group. Interpret the UML Class Diagram 1: Project group to determine which classes must be created and what their class members are.

When done: add controls to the form to allow a user to add members to a project group and show the info of the project group. HINT: Initialize one *ProjectGroup*-object in the constructor of the form to ensure the object is available when the application ‘starts’.

### Provided material



UML Class Diagram 1: Project group

# Quiz answers

|  |  |
| --- | --- |
| Question | Answer |
| 1 | 0 constructors. The class only contains instance variable, void and typed methods. |
| 2 | The constructor is incorrectly overloaded. When overloading a method/constructor-header must have the same name but different parameter. In this case different parameters also means different order of parameter types.  The two constructors in class Buzzer both have a header like public Buzzer(int, double), which violates the rules of overloading. |
| 3 | 1. *How many typed methods does class Song have?* 4; GetId(), GetInfo(),GetTitle() and GetDuration(). 2. *How many public instance variables does class Song have?* 0; none of the instance variables have a - symbol in front of them. 3. *How would you implement the association between class MusicPlayer and Song?* Either as an instance variable of the types *List<Song>* or *Song[].* The association has an multiplicity indicating that a MusicPlayer-object must always have between 1- 20 *Song*-objects; a collection types is ideal for this. 4. *Which method(s) are overloaded?* In class *MusicPlayer*: GetSongs() In class *Song* no methods, but the constructor is overloaded 5. Does class *MusicPlayer* have a default constructor? No, a default constructor is when no constructor will be defined in a class. The UML Class Diagram depicts that the class *MusicPlayer* must have a constructor declared with logic in it. |

1. Total amount of money earned by selling the snacks [↑](#footnote-ref-1)