Practice Assignments Intro Objects and Classes

Quiz questions, practical assignments and

answers to quiz questions

**Version:** 1.0 **Last updated:** 28 augustus 2023

Contents

[1 Quiz: Classes & Objects 2](#_Toc144115536)

[1.1 Question 1 2](#_Toc144115537)

[1.2 Question 2 3](#_Toc144115538)

[1.3 Question 3 3](#_Toc144115539)

[2 Practical assignments: Classes & Objects 5](#_Toc144115540)

[2.1 Programming Assignment 1: Defining methods 5](#_Toc144115541)

[2.1.1 Case description 5](#_Toc144115542)

[2.1.2 Provided material 5](#_Toc144115543)

[2.2 Programming Assignment 2: Working with objects 6](#_Toc144115544)

[2.2.1 Case description 6](#_Toc144115545)

[2.2.2 Provided material 6](#_Toc144115546)

[2.2.3 Screenshots 7](#_Toc144115547)

[2.3 Programming Assignment 3: Car application 8](#_Toc144115548)

[2.3.1 Case description 8](#_Toc144115549)

[2.3.2 User interaction 8](#_Toc144115550)

[2.3.3 Provided material 8](#_Toc144115551)

[2.3.4 Additional features 9](#_Toc144115552)

[3 Quiz answers 10](#_Toc144115553)

# Quiz: Classes & Objects

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

## Question 1

In a solution, we have the (incomplete) class and method call:

BankAccount accountA = new BankAccount();

accountA.SetBalance(123.45);

double newBalance = accountA.GetBalance();

public class BankAccount

{

private string client;

private int accountNo;

private double balance;

public void SetBalance(double newBalance)

{

this.balance = newBalance;

}

public double GetBalance()

{

return this.balance;

}

}

Why do you need to make use of the methods *SetBalance* and *GetBalance* to change or access the value of the instance variable *balance*, respectively?

## Question 2

In a solution, we have the class Buzzer:   
Given the code snippet below, specify where the errors in this piece of code:

Buzzer b;

b = new Buzzer();

b.nrOfCurls = 25;

b.SetPz(45.95);

string buzzerInfo = b.GetInfo();

b.GetInfo() = "0 xx 8";

public class Buzzer

{

private int nrOfCurls;

private double pz;

public void SetPz(double value)

{ this.pz = value; }

public string GetInfo()

{

string holder = this.nrOfCurls.ToString() + " xx " + this.pz.ToString();

return holder;

}

}

## Question 3

We will again look at the class Buzzer with an extra method (*addCurls*):

public class Buzzer

{

private int nrOfCurls;

private double pz;

public void SetPz(double value)

{ this.pz = value; }

public string GetInfo()

{

string holder = this.nrOfCurls.ToString() + " xx " + this.pz.ToString();

return holder;

}

public void AddCurls(int value)

{ this.nrOfCurls += value; }

}

We have the following code snippet. What will be the value the message variables after invoking the *runQuestion2()* method?

private void runQuestion2()

{

Buzzer b1 = new Buzzer();

Buzzer b2 = new Buzzer();

b1.SetPz(1.5);

b1.AddCurls(7);

string message1 = b1.GetInfo(); // Message1

string message2 = b2.GetInfo(); // Message2

b1.AddCurls(7);

string message3 = b2.GetInfo(); // Message3

b2.SetPz(22);

b1.AddCurls(-10);

string message4 = b1.GetInfo(); // Message4

string message1 = b2.GetInfo(); // Message5

}

# Practical assignments: Classes & Objects

## Programming Assignment 1: Defining methods

Difficulty:

The assignment covers the following learning goals:

* Define the methods to operate on the objects.

You’re going to implement your first class by ‘completing’ class *BankAccount*. As practice, these methods will be basic but they will require you to apply the concepts you learned during the orienting phase to!

### Case description

In this assignment you are going to ‘complete’ a *BankAccount* class by defining methods using the instance variables. Some of the methods must include logic to ensure the balance cannot become negative.

First create a new Form Application in visual studio and add a class *BankAccount.* When you’re done with this, you can copy/paste the members from class *BankAccount* (see provided material).

Now add the following methods to the class:

* *public void InitializeBankAccount(string newClient, int newAccountNo)*: When this method is called the values of the parameters have to be assigned to the corresponding fields. In addition, assign the value 0 to the instance variable *balance*
* *public void DepositMoney(double amount)*: When this method is called the *balance* must be increased by the amount.
* *public bool WithdrawMoney(double amount)*: When this method is called the *balance* must be decreased by the *amount*. In addition, money can only be withdrawn when the *balance* is sufficient (e.g. it should not be possible to withdraw money when the *balance* is 10 euro and the *amount* 10.01). Return *true* when the withdrawal was successful and *false* when it was not.
* *public string GetInfo()*: When this method is called a string must be returned with the format  
   Client <client> (<accountNo>): <balance> euro(s)  
  For example: “Client John Doe (12345): 100.50 euro(s)”, “Client Jane Jackson (982364): 0.00 euro(s)”, etc.

### Provided material

public class BankAccount

{

private string client;

private int accountNo;

private double balance;

}

## Programming Assignment 2: Working with objects

Difficulty:

The assignment covers the following learning goals:

* Create an object of a class-type.

### Case description

You’re going to create your first object. As practice, you will be supplied with the code for the class but you will have to create the code for the object by yourself.

Start a new Windows Form Application project and add a class *Course* to it. When the file has been created, you can copy/paste or recreate the code supplied in the provided material section.

Add controls to Form1 such as shown in Figure 1 (see screenshot). After you’ve done that, implement the required code for the *Create Course-object*-button. Hint: you will have to create the *Course*-object and call the correct methods to set the fields.

When done you can also implement the *Show info . . .*-button. Can you make it in such a way that the buttons make use of the same *Course*-object?

### Provided material

public class Course

{

// Instance variables

private string name;

private int ec;

// Methods

public void SetName(string name)

{ this.name = name; }

public string GetName()

{ return this.name; }

public void SetEc(int ec)

{ this.ec = ec; }

public int GetEc()

{ return this.ec; }

}

### Screenshots

Graphical user interface, text, application

Description automatically generated

Figure 1: Possible GUI

## Programming Assignment 3: Car application

Difficulty:

The assignment covers the following learning goals:

* Define a class with its members
* Create an object of a class-type
* Declare and use the fields necessary to store data
* Define the methods to operate on the objects

### Case description

We want to implement a simulation for a car accelerating and braking. Each car has a current speed and can become higher (by accelerating) or lower (by braking). Every car also has a maximum speed, for example, a Ferrari has a much higher maximum speed than a Fiat Panda.

### User interaction

The application should display some information about a car. There are two buttons to modify the speed of a car. Clicking one of the buttons should result in updating some data in the car-object and showing information about the car in the label.

Graphical user interface, application

Description automatically generated

Figure 2: Possible GUI

### Provided material

Start a new solution and add a class *Car* to it. Add some private fields to the class to store its *brand*, its *current speed* and its *maximum speed*.

To be able to store a brand, for instance “Ferrari”, add a method to the class Car to set the brand:

public void SetBrand(string brand)

Add also a method to the class *Car* :

public string GetInfo()

that returns a string with information about this car, as pictured in the label in the screenshot above.

To be able to store the maximum speed, add a method to the class *Car*:

public void SetMaxSpeed(int maxSpeed)

We don’t want to have a method to store the current speed. Instead of this, we want to have a method to accelerate and a method for slowing down when using the breaks:

public void SpeedUp()

public void SlowDown()

In the real world you can speed up or slow down with a variable acceleration. In this assignment, we assume speeding up is always by increasing the current speed with 7 km/h. Slowing down is always by decreasing the current speed with 10 km/h. Implement these two methods in the Car-class.

Now let’s work on *Form1*. Add some controls to your form to make it look like the screenshot above. Now add a variable:

private Car myCar;

to *Form1* to hold an object of type Car. At start-up of your app, you should create the *Car-*object and assign it to *myCar*. After creating the object, give it a brand by calling the correct method; the actual brand is up to you.

Start your app to see if it runs. If so, we do not see anything on the screen yet and clicking the buttons has no effect. But it runs, so there are no syntactical errors in it.

Now, show the information of *myCar* in the label by calling the correct method. Start your app and you will see that the brand is shown on the screen. Since the *current speed* and *maximum speed* are not set by you, they have the value 0.

To be able to store the *maximum speed*, add a method to the Car-class:

public void SetMaxSpeed(int maxSpeed)

In the code for *Form1*, after creating the object *myCar* and giving it a brand, set the maximum speed of *myCar* to a certain value.

Make sure that the buttons are working by calling the proper method for the object in *myCar* and then showing the information of the object *myCar* in the label.

Question: if you accelerate several times, is it possible that the speed of the car exceeds the maximum speed? If that is the case, adjust your implementation for your method public void SpeedUp() in such a way, that the current speed is never bigger than the maximum speed.

Same question for using the brakes: could the speed become negative?

### Additional features

Can you extend the application by simulating 2 cars.

# Quiz answers

|  |  |
| --- | --- |
| Question | Answer |
| 1 | The instance variable *balance* has the access modifier *private*. This ensures that the value of *balance* can only be change by the methods in class *BankAccount*.  When an object of type *BankAccount* is created, only the *public* members can be accessed. In other words, all ‘interactions’ with the *balance* will have to go via the public methods *SetBalance* and *GetBalance* |
| 2 | The incorrect statements are:   * b.nrOfCurls = 25; * b.GetInfo() = "0 xx 8"; |
| 3 | message1: 7 xx 1,5  message2: 0 xx 0  message3: 0 xx 0  message4: 4 xx 1,5  message5: 0 xx 22 |