

EE 441 Recitation Notes:

Project Creation, Code Writing and Formatting

1 CONTENTS:

- Integrated Development Environment (IDE) Installation
- Before Starting
 - Project Creation on Code::Blocks
- How to Write a Clear Source Code?
 - Formatting in Your Code
 - Commenting
- Homework Format
 - Project and Folder Format
 - Naming Format
- Homework Sending Procedure
 - METU-Online Course Page
 - Proper Homework Sending

2 IDE INSTALLATION

Integrated Development Environment is a software application that generally consists of a source code editor, compiler, linker and a debugger. This creates an easy and ready-to-use software creation environment for code writers.

We will use Code::Blocks for this year homework submissions processes. Code::Blocks is chosen because it is an open source software and it can work on all widely used platforms such as Windows, Linux, and Mac OS X.

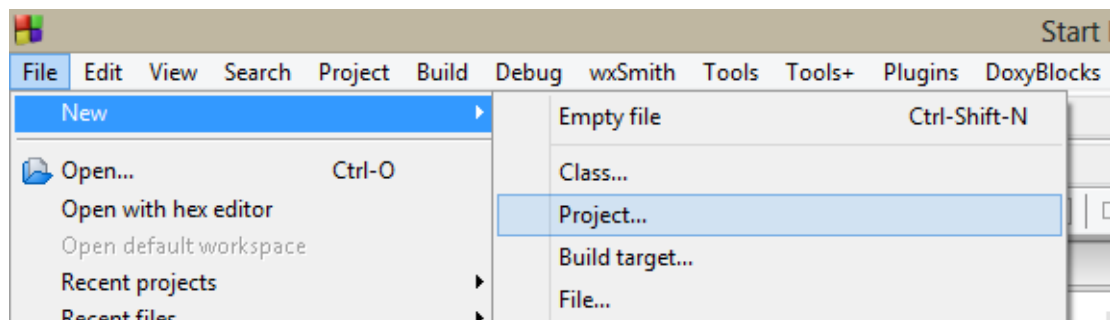
In order to install the Code::Blocks software, an installation should be downloaded from the Internet as a first step. The setup files can be downloaded from the following link: <http://www.codeblocks.org/downloads/binaries>. There is a tricky part while choosing the best choice for our system. If you use Windows as your operating system (OS), then you should download the link with MinGW (Minimalist GNU for Windows) since Code::Blocks uses GNU C Compiler (*gcc*) as its primary compiler. We want all homeworks to be able to work on all platforms, and because for that reason, they all need to have the same specifications. Hence, the download link for Windows users is: <http://sourceforge.net/projects/codeblocks/files/Binaries/13.12/Windows/codeblocks-13.12mingw-setup.exe/download>. The Linux users should choose the proper option related to their distributions (such as Ubuntu users should choose Debian based download link). Mac OS X users may prefer to download the SP1 version (<http://sourceforge.net/projects/codeblocks/files/Binaries/12.11/MacOS/>) in order to avoid the “stability issues”.

After the download, the installation should be done. It is advised that the program installation should be completed with administrator right to avoid any circumstances.

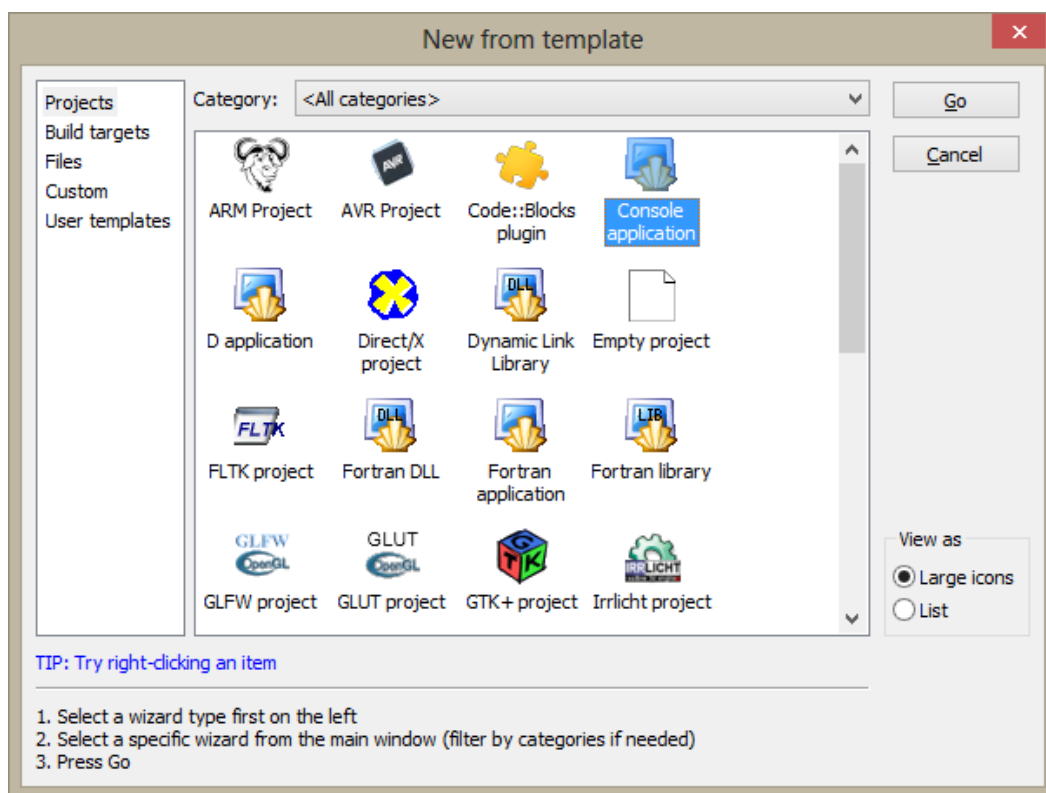
3 PROJECT CREATION IN CODE::BLOCKS

After installing the Code::Blocks to your system, a project can be created in order to start creating a program. A project is a complete collection of the headers, code parts, and some additive elements such as different wrapped up *dll* or *lib* files or even media files such as image and audio. All code writing exercises should be completed in a project structure in order for IDE to compile and link all parts of a software properly.

A new project can be created by following the “File -> New -> Project” path. After that, the “Console Application” choice should be chosen.

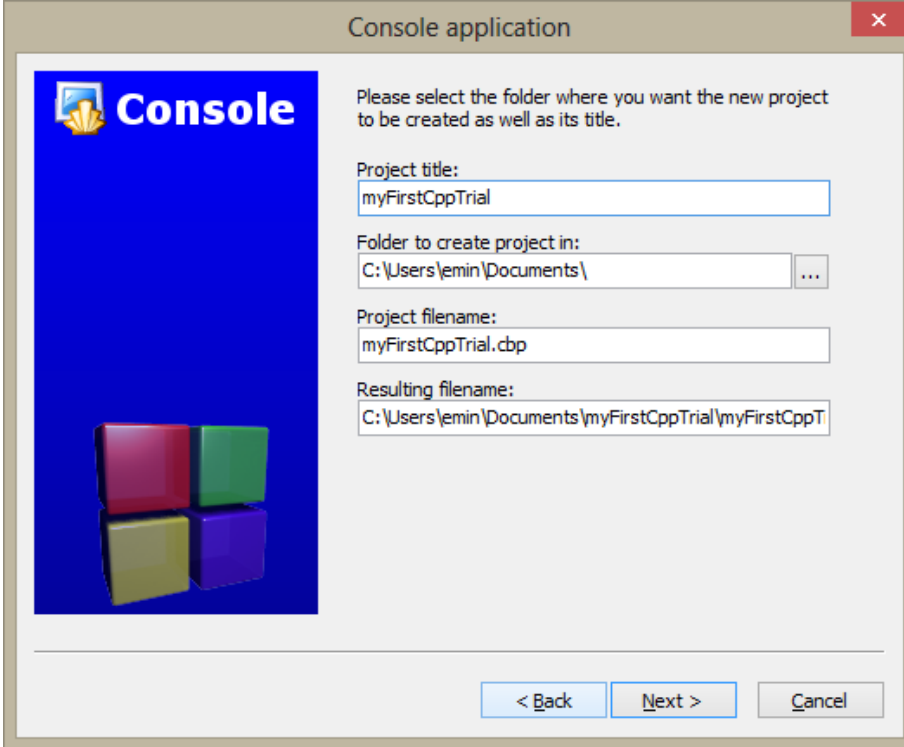


That will bring you a menu with different options. The “Console application” should be chosen.



After you have chosen the console application you should choose “C++” choice (that screen may appear after a “Next” command) and hit the “Next” button. In the next page, a project title is needed

to be entered. Enter your understandable one-word project title (e.g. myFirstCppTrial). After you named your project, hit the “Next” button once more.

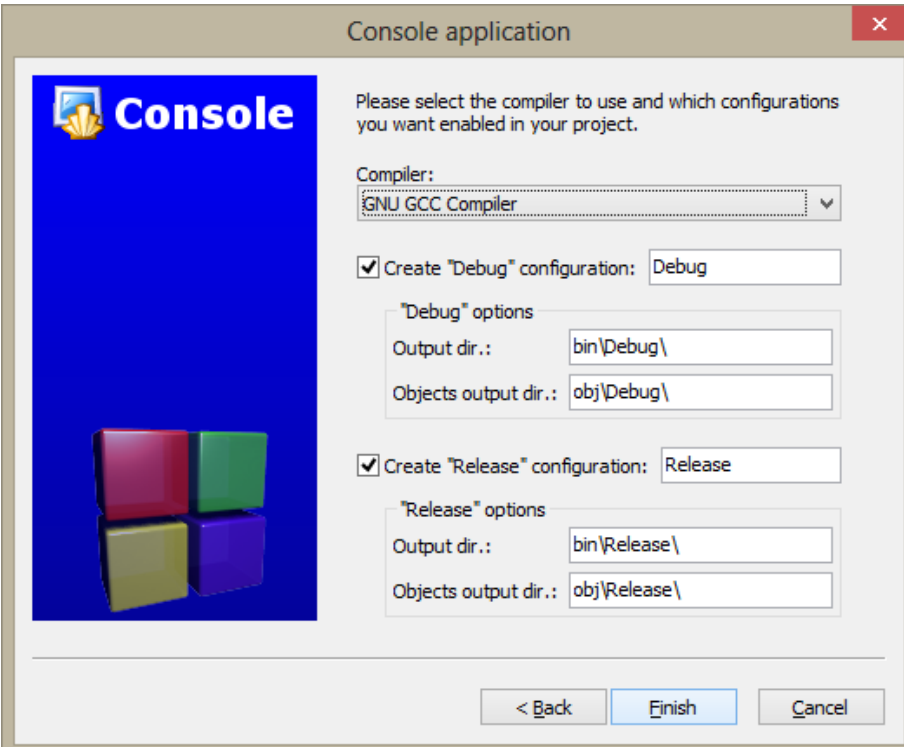


The screenshot shows the 'Console application' wizard window. On the left is a blue panel with the 'Console' logo and a 3D cube graphic. The main area contains the following fields and instructions:

- Instruction: "Please select the folder where you want the new project to be created as well as its title."
- Project title:
- Folder to create project in: ...
- Project filename:
- Resulting filename:

At the bottom are three buttons: "< Back", "Next >", and "Cancel".

Choose the “GNU GCC Compiler” (or leave it as it is if already chosen) and push “Finish” button. Now you have created a new project as your first C++ trial.



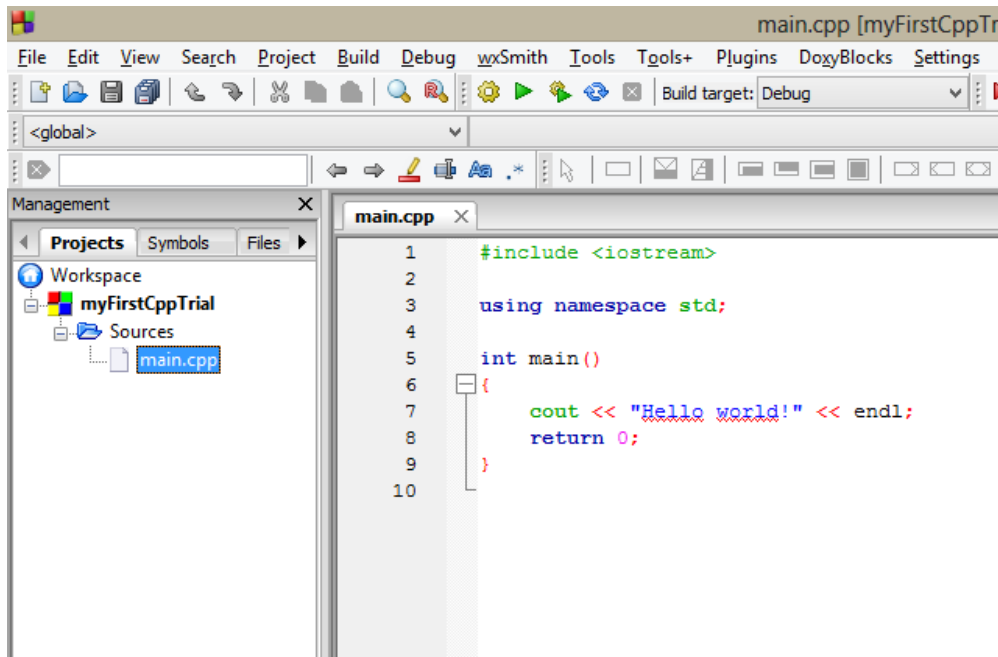
The screenshot shows the 'Console application' wizard window at the second step. The left panel is the same as in the previous screenshot. The main area contains the following fields and instructions:

- Instruction: "Please select the compiler to use and which configurations you want enabled in your project."
- Compiler: (dropdown menu)
- ☒ Create "Debug" configuration:
- "Debug" options:
 - Output dir.:
 - Objects output dir.:
- ☒ Create "Release" configuration:
- "Release" options:
 - Output dir.:
 - Objects output dir.:

At the bottom are three buttons: "< Back", "Finish", and "Cancel".

After the creation, you have you main.cpp and “Hello world!” example. The project creation step ends here. If you need to add (or create) different files such as headers or additional *cpp* files, you should

create or add one by choosing “File -> New -> Empty File” option and confirm for addition to the project for creation. You may also right click the project icon on the left pane and choose “Add files” to add external files to your project.



4 HOW TO WRITE A CLEAR SOURCE CODE?

In order to write a good source code, you need to understand the problem and create an algorithm to solve the standing problem. After an algorithm is created, each step of algorithm is taken as different parts or mostly one or more separate functions in order to support both readability and flexibility.

What is the importance of writing a clean and understandable source code if it does its job well? As many engineering disciplines, we need to use programming for our needs. In this endeavor, there will be different and numerous colleagues to work with. While working with some other team on a certain software development problem, the communication between the developers carry a great importance for the well-being of the work. Even for a single developer, the understandability is needed. You may need your previously written programs and you may not remember the exact way of thinking. Thus, writing a clean and understandable software is an essential behavior.

There are different opinions and debates about the good source writing. To start with, here there are some very basics to keep your source code both understandable and flexible for other applications or additions.

4.1 FORMATTING IN YOUR CODE

In order to keep your source code understandable, first you must start with understandable variable names and class names. Setting your variable names clear will help you when your source code gets more and more complex. You may create or choose your own style of formatting and commenting.

There are different conventions in software development society, which you are NOT asked to learn within the scope of this course. Each of these are created to ease the understandability. If you are

interested in formatting your code in a better way, you may look at <http://google-styleguide.googlecode.com/svn/trunk/cppguide.xml> for Google's C++ Guide, which is used in many open-source projects, and different code styling tips and tricks such as naming convention for your functions and parameters.

4.2 COMMENTING

Commenting is a necessary and helpful act of notation in software development works. Through the supplied comments, a programmer can understand the previous work without analyzing the whole software line by line. Without the proper comments, any source code becomes very hard to follow and understand. The commenting should not be overly done (i.e. comments in every line) but it should be done to explain main functions of the blocks or functions. The code given in Appendix part shows an example of both formatting and commenting in a smaller scale.

Writing a clear source code for your software may be a dull thing to do; however, it is necessary to be a good programmer. Please keep those in your mind while doing your work.

5 HOMEWORK FORMAT

Even though the required format for the homeworks will be mentioned in each homework announcement, the format in general requires a few basic principles. These principles can be mentioned as follows:

Implement a completed work of homework requirements.

Every homework will be given by its fully covered requirements. Each of these requirements are important and will affect the performance of your work, and also your grade. Please pay attention to the requirements and try to implement those as your best.

Try to provide a clear output formatting when you are working on Command Line Interface.

On this course, you will be writing console (i.e. command line) applications. It is a good practice for your application to give its outputs in a formatted form in console. That way, your work are easy to follow, and you can understand much easy only by looking to your output and tell where the mistake is when you are debugging your problem.

Send all documents you are asked for as they are.

Please send all extra documents that you are asked to send besides your project in the form as they are asked. You may be asked to write an extra report or comment to your work besides your source code. Examples to that may include the word documents in 93-97 Word Doc form 'doc', portable document files Adobe 'pdf', or any other document in the form as they asked.

Always clear your build before compressing and sending your work.

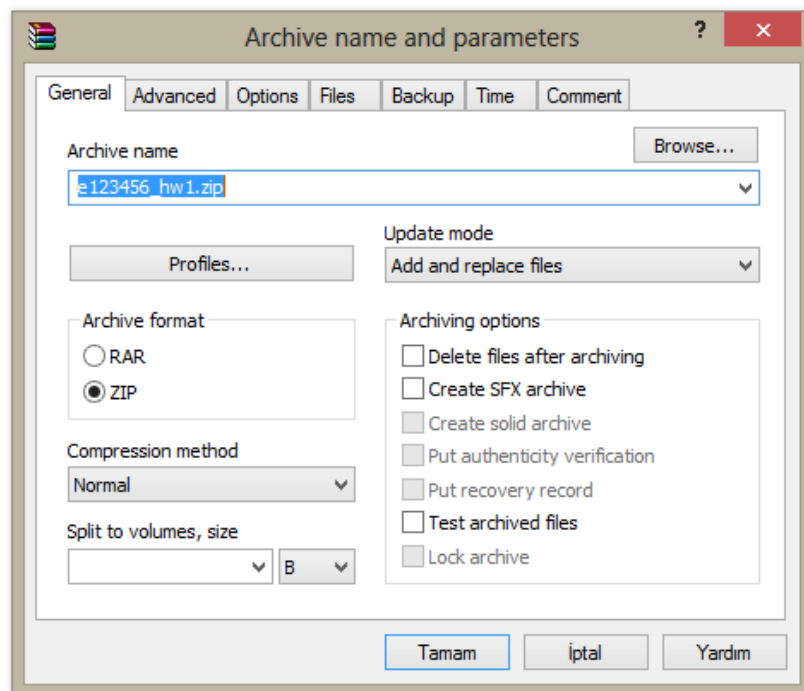
Because we all may use different operating systems, the executables generated due to your build is not required. In addition to that, it is essential to be your source code to be buildable at any platform. Hence your source code will be built after your homework is downloaded.

Because of these reasons, we do not require executables on your homework. To clean your project, please choose “Build->Clean” option before you compress your project folder.

Always send the project folder (and extra documents) in its compressed form with zip file format.

In order to evaluate your homework, we need a complete project contents with their proper relations. So, sending the project folder after compressing that folder in a zip file is required. Find your project folder in your computer (generally it is located under your Documents for Windows users) and compress the whole folder named as you name it at first (for this example it’s “myFirstCppTrial”). If there are extra documents needed, copy those to the same location of your project folder, then **compress all of them in a single** zip file.

myFirstCppTrial	16.10.2013 17:47	Dosya klasörü	
e123456_homework1_report.pdf	16.10.2013 17:48	Adobe Acrobat D...	79 KB
Yeni Microsoft Word Document.docx	16.10.2013 17:47	Microsoft Word D...	0 KB



Name your compressed folder as your student id in e123456 form.

Name your compressed zip file as it is asked in the homework. The naming generally done by “e123456_hwX.zip” where your student number is 123456-7.

Why do we do all these restrictions? Can’t we just write some code and upload it to the Online?

Each one of those which are asked from you is a part of readying you to submit a good, proper, and well documented source code as it is needed to be submitted. Even though you did a perfect job, without proper formatting and documentation, that job will remain as a code piece not a software. You may want to be a programmer or not, either way the importance of delivering a good wrapped holds still.

6 HOMEWORK SENDING PROCEDURE

After you have completed your homework and packed all the things you needed in a zip file, the last thing to do is to send the homework to the METU-Online course page. To do that, you need to first log in to METU-Online and select the course from the left pane.

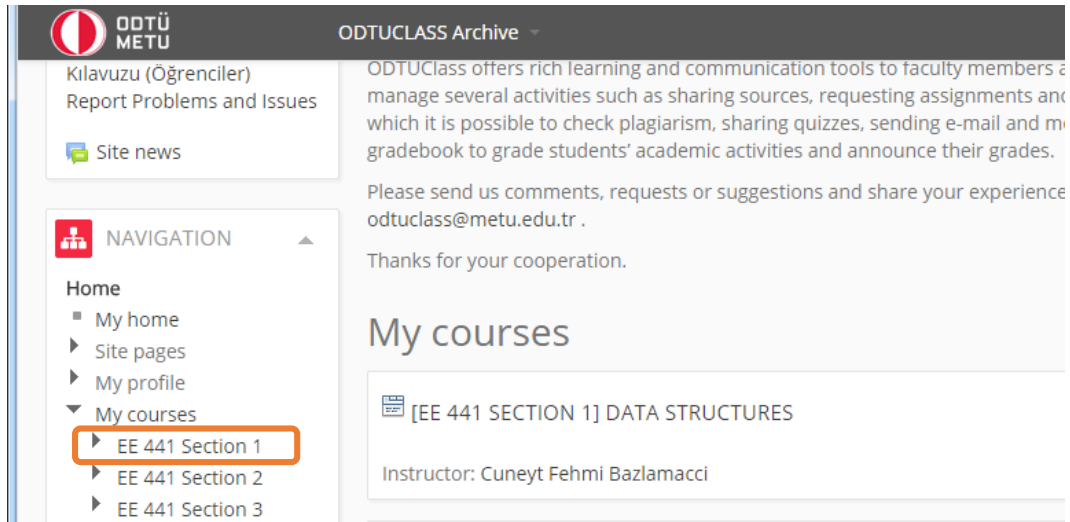


Figure 1 – ODTUClass Dashboard

After you have selected the course, you should find the assignment with relevant title in your course schedule.

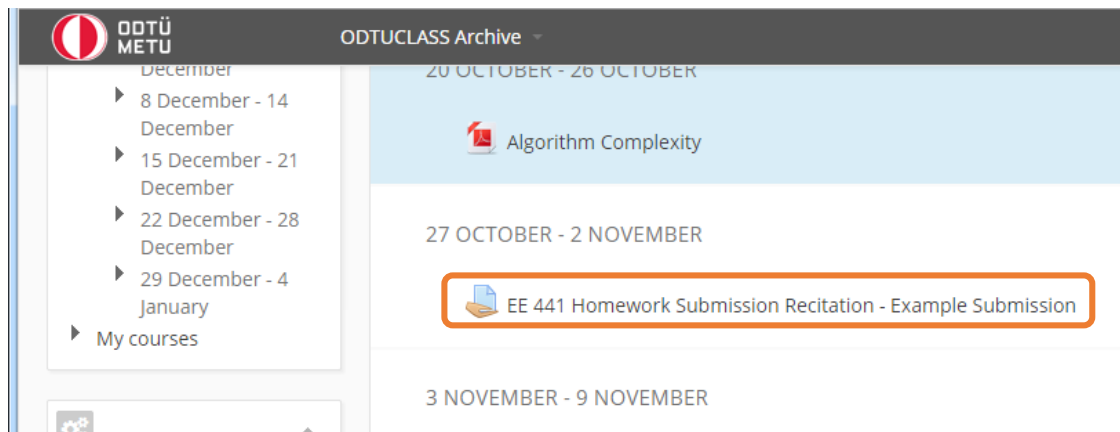


Figure 2 – ODTUClass Course Page, Schedule

After you have located the correct homework name, click the assignment, you will see a menu as shown below:

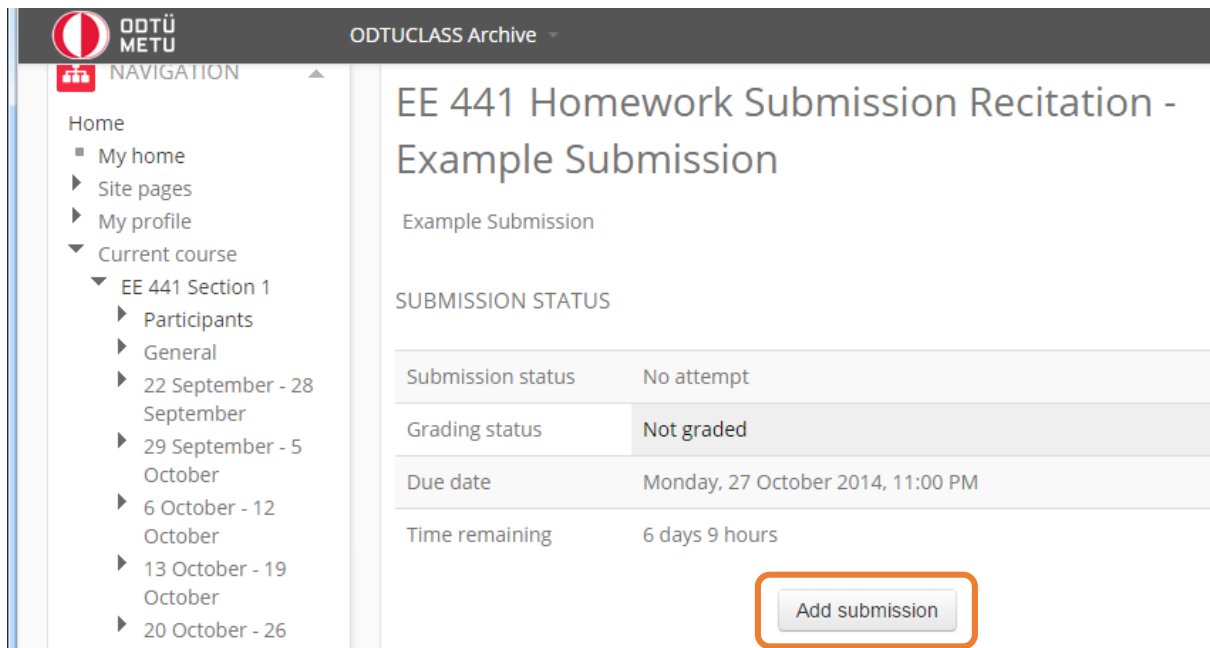


Figure 3 – Homework Submission Screen

Click “Add submission” button to upload your homework file (which is basically compressed folder) in .rar or .zip format. For this operation, you can either drag-and-drop your compressed file, or you may use “File picker” by pressing new file button.

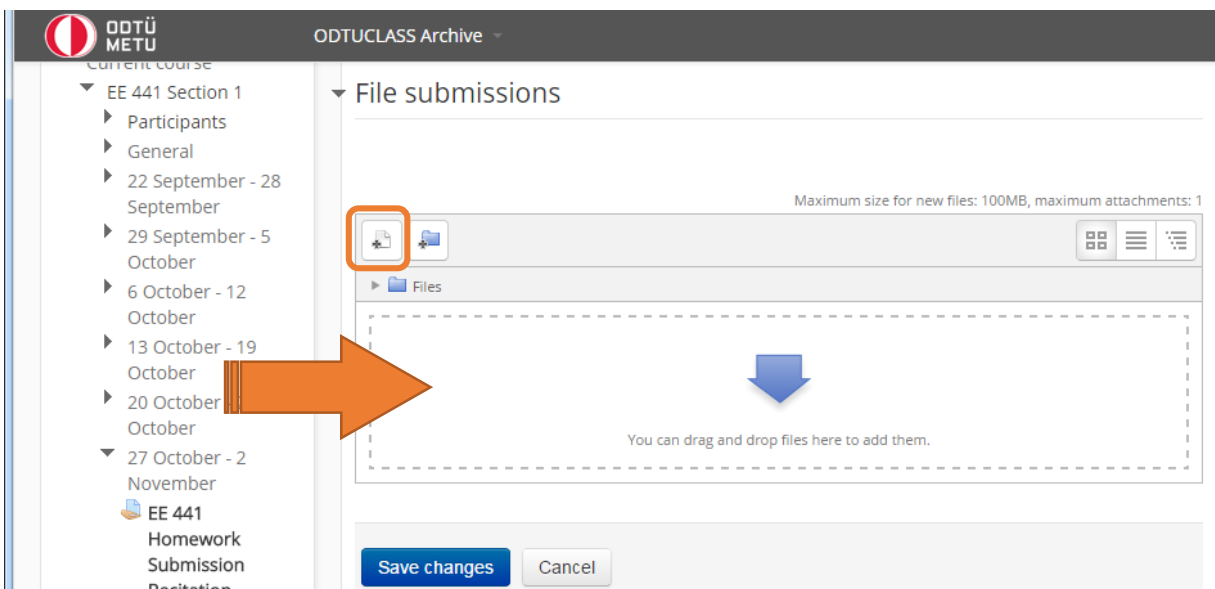


Figure 4 – Add Submission Screen (You may drag-and-drop your file or use blank file button to open File Picker)

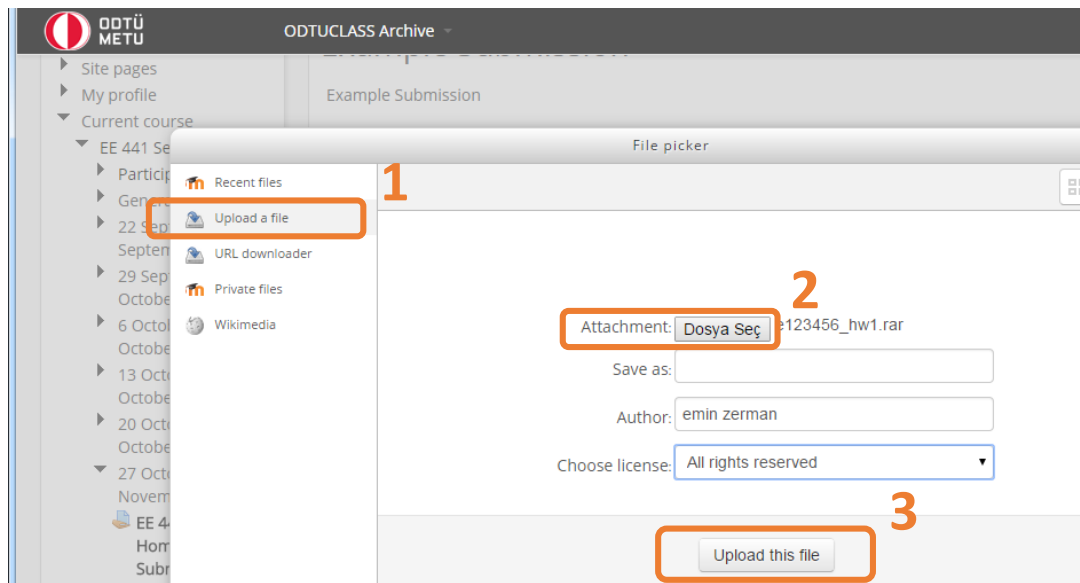


Figure 5 – File Picker Screen (You should first choose “Upload a file” from left menu and “Choose a File”)

While using “File picker”, you should first choose “Upload a file” option from the left menu and “Choose a File” menu. Choose the file from your computer and choose “Upload this file” choice.

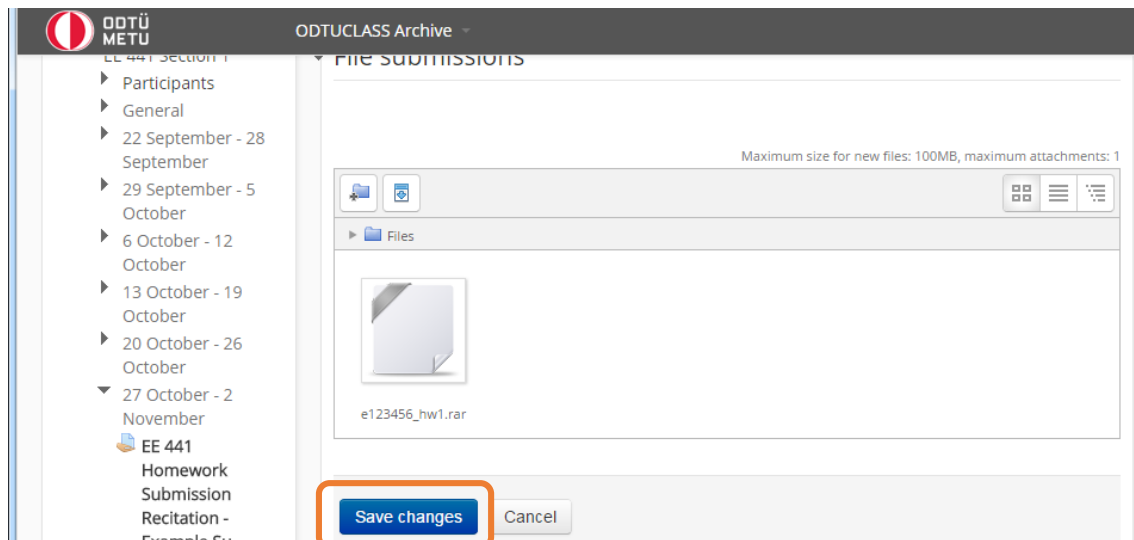


Figure 6 – Add Submission Screen After Upload

Now, you should be able to see your uploaded file. You must press “Save Changes” button in order to complete your submission.

It is strongly encouraged that to check the uploaded document by downloading your own submission as shown in Figure 7. You may also delete your submission if you accidently submit wrong file.

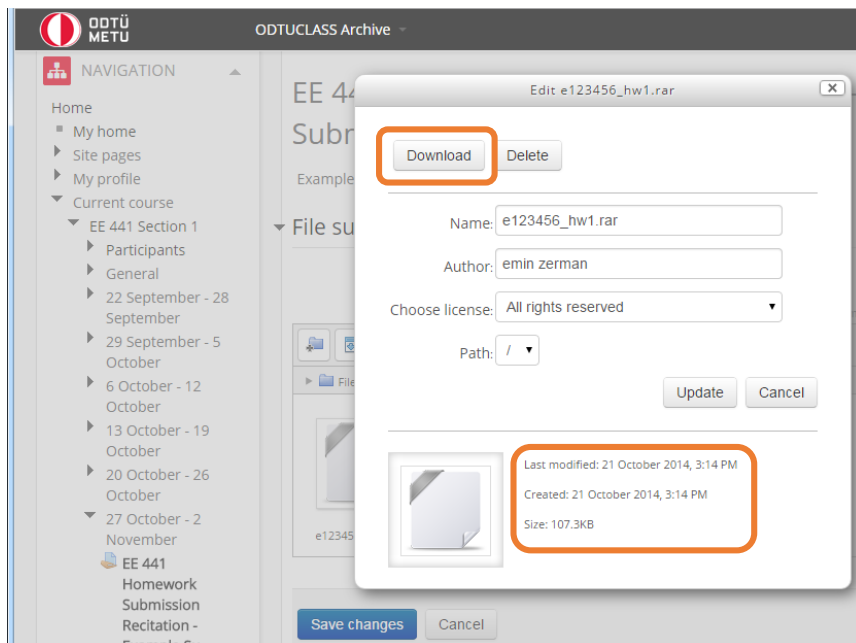


Figure 7 – You should check the uploaded document by downloading your own submission and by checking size

After you have completed submitting your homework, you can see your submission status has been changed to “Submitted for grading”. You also may edit your submission by pressing “Edit submission” button.

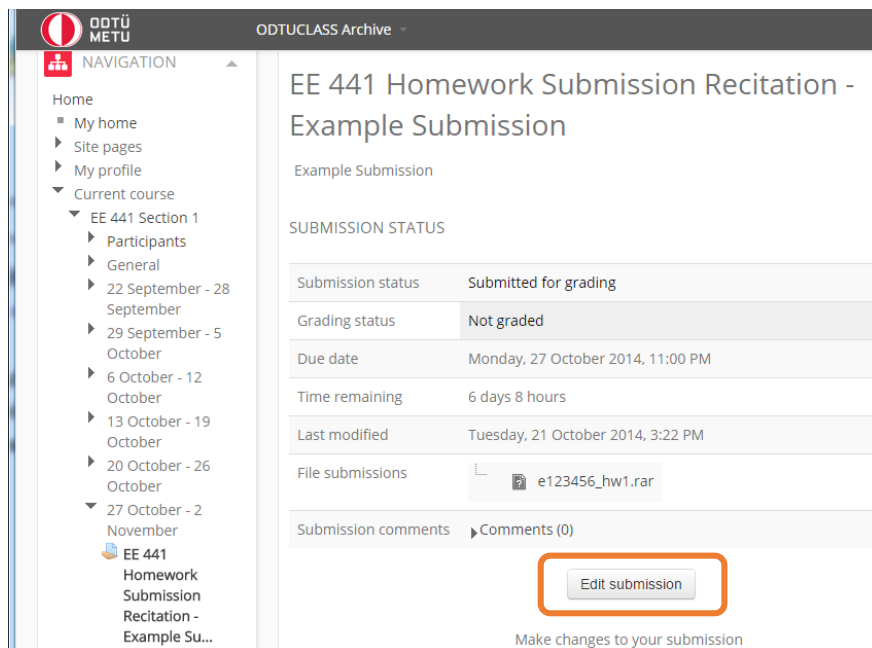


Figure 8 – Assignment Screen, After Submission

7 APPENDIX – SAMPLE CODE

Please write an understandable source code while writing your software. The sample code for a clear formatting is given below:

```
//=====
// Program:    Sample Software
// File:       ee441_sample.cpp
// Function:    Main (complete program listing in this file)
// Description: Find the required fee for a reservation program
// Author:     Ali Yazar (aliyazar@abc.com)
// Notes:      This is an introductory, sample program.
// Revisions:   1.00 11/11/2011
//=====

// -----Header Files-----
#include "ClassStructure.h";
#include "PreWritten.cpp" //Used for some other functions

// -----Global Variables-----
int exitter = 0;

// -----Functions-----
void welcome(){
    cout << endl << "Welcome to The Program " << endl;
    cout << "-----" << endl;
    cout << "Choose your option:" << endl;
    cout << "1- If you are a customer of ABC Company" << endl;
    cout << "2- If you are not a customer" << endl;
    cout << "3- If you are Turkish, press 3!" << endl;
    cout << "4- Exit the program" << endl;
}

void welcome2(){
    cout << endl << "Welcome Back! " << endl;
    cout << "-----" << endl;
    cout << "Choose your option:" << endl;
    cout << "1- Enter with your customer number" << endl;
    cout << "2- Enter with your ID number" << endl;
    cout << "3- Exit" << endl;
}

int computeRequiredFee(ClassStructure* classPtr, int workDays){

    return classPtr->computeFee(workDays); // This is used for the easy reading of main program
}

// -----Main Function-----
void main(){

    ClassStructure* FirstObject=NULL;
    char choice, choice2;           //will be used to save integer value corresponding to user input
    string name;
    int customerNumber, idNumber;   // variable names such as custNo can be not understandable
    int workingDays, result;

    // -----User Interface-----
    while (exitter==0){             //User interface loops until user wants to exit
        welcome();                  //Entrance menu is displayed
        cin>>choice;                //User enters his/her choice via keyboard
        switch(choice){             //The operation corresponding to user input is done

            case '1':                //If customer.
```

```
welcome2(); // Second welcome menu
cin >> choice2;
switch(choice2){
    case '1':
        // If customer number is given
        cout << "Please enter your customer number : " << endl;
        cin >> customerNumber;
        break;

    case '2':
        // If ID number is given
        cout << "Please enter your ID number"
        cin >> idNumber;
        customerNumber = findCustomerNumber(idNumber);
        // findCustomerNumber function is from PreWritten.cpp
        break;

    case '3':
        cout << "Exiting to main menu!" << endl;
        continue;

    default:
        // If none given
        cout << "\t\tERROR:You entered an invalid command! Start again!"
        continue;
}
// Find customer and working days
FirstObject = findCustomer(customerNumber);
workingDays = findWorkDays('1',FirstObject);
result = computeRequiredFee(FirstObject, workingDays);
respond(result);
break;

case '2': // If not a customer
    cout << "Please enter your work days:" << endl;
    cin >> workingDays;
    result = workingDays * 3;
    respond(result);
    break;

case '3': //If Turkish
    cout << "Uzgunuz Turkce menu hizmetimiz bu numarada saglanmamaktadır. Turkce
    menuye erismek icin 441 BORG numarali telefonu arayiniz." << endl;
    exitter = 1;
    break;

case '4': // Exit from the program
    exitter = 1;
    break;

default:
    cout << "\t\tERROR:You entered an invalid command!" << endl;
    break;
}
// ----- End of User Interface -----
}
// ----- End of Main Function -----
```