CME 3202 - Concepts of Programming Languages

Laboratory Worksheet#1

Laboratory Aim

In this laboratory session, you will be introduced to how the lab sessions are carried out with a simple example. Then, compiler and interpreter will be introduced.

Each laboratory worksheet consists of exercises on one or more subtopics of that week's lecture. Each exercise contains tasks to carry out, and questions to answer. You need to implement the tasks by following the steps, and answer the questions; and finally upload your worksheet into the ***"https://online.deu.edu.tr/"*** using the ***"appropriate and correct upload link"***.

repl.it online coding platform will be used for most of the exercises in this course, so you are advised to register the site in https://repl.it/.

The remaining part of this worksheet contains a sample exercise which you need to keep on.

Class Variables vs. Instance Variables

Defining a static variable or a method in a class in C#, C++ and Java makes the variable a *class* variable/method instead of an *instance* variable/method. Class variables are created, and allocated in the memory before the class is first instantiated, and there is only one copy of it; whereas instance variables/methods are created separately for each instance.

Class (static) methods can reach, and modify ONLY class (static) variables; they cannot see instance variables. On the other hand, class (static) variables can be reached, and modified by both class (static) and instance methods.

Exercise 1

Do the following steps, and implement a sample illustrating usage of static keyword in Java.

Step 1

Open https://repl.it , and login if you have already registered in. Create a new repl by clicking +new repl button.

Step 2

Select Java as the programming language. Create main class and add the following lines into main class.

```
    Bicycle firstBicycle= new Bicycle("Bianchi");
    Bicycle secondBicycle= new Bicycle("Bisan");

    System.out.println("Total number of bycles:" +
Bicycle.numberofBicycles);
```

Step 3

Create Bicycle Class and add following lines.

```
    public int gear = 1;          // instance variable
    public String type;           // instance variable
```

```
    static int numberofBicycles = 10;  // class variable

    public Bicycle(String bicycleType){
        numberofBicycles++;
        type = bicycleType;
    }
```

Step 4

Click ***"Run"*** button.

Task 1

Paste the output of the code.

```
Total number of bycles: 12
```

Task 2

Add this code fragment to main function, see the results and explain.

```
secondBicycle.numberofBicycles = 20;

System.out.println("Number of bcycles of secondBicycle:" +
secondBicycle.numberofBicycles);

System.out.println("Total number of bcycle after making changes:" +
Bicycle.numberofBicycles);
```

```
Total number of bycles: 12
Number of bcycles of secondBicycle:20
Total number of bcycle after making changes:20
```

Task 3

Write the required code statement to print the types of two bicycles.

```
System.out.println("First Bicycle type: " + firstBicycle.type); System.out.println("Second Bicycle type: " + secondBicycle.type);
```

Question 1

What is the difference between using "`numberofBicycles`" variable vs. "`type`" variable? How do we access these variables?

```
numberofBicycles: A static variable, shared by all instances, and accessed via the class name
```

type: An instance variable, unique to each object, accessed through the object reference

Task 3

Add the method that is given below to inside of ***"Bicycle"*** class, and run.

```
public static void resetNumberofBicycle(){
      numberofBicycles=0;
      gear = 3;
   }
```

Question 2

Explain the cause of the error which you get.

Static methods belong to the class and do not have access to instance variables (like gear). Hence, you cannot modify or reference instance variables directly in a static method.

```
./src/main/java/Bicycle.java:12: error: class, interface, enum, or record expected
public static void resetNumberofBicycle(){
           ^
./src/main/java/Bicycle.java:14: error: class, interface, enum, or record expected
   gear = 3;
   ^
./src/main/java/Bicycle.java:15: error: class, interface, enum, or record expected
}
```

Question 3

In which occasion, static methods should be used?

Static methods are used when the operation does not depend on instance-specific data. They are ideal for tasks that are common to all instances of the class, such as utility or helper functions, or for resetting shared data.

In this part, you are expected to exercise three different implementation methods in three different programming languages.

Compilation

Compilation is an implementation method in which programs are translated into machine language, and executed directly on the computer.

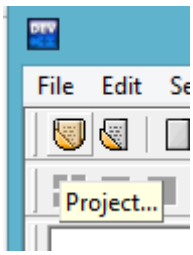
An example of programming languages which is implemented with compiler implementation is C. Apply the following steps, and implement the tasks.

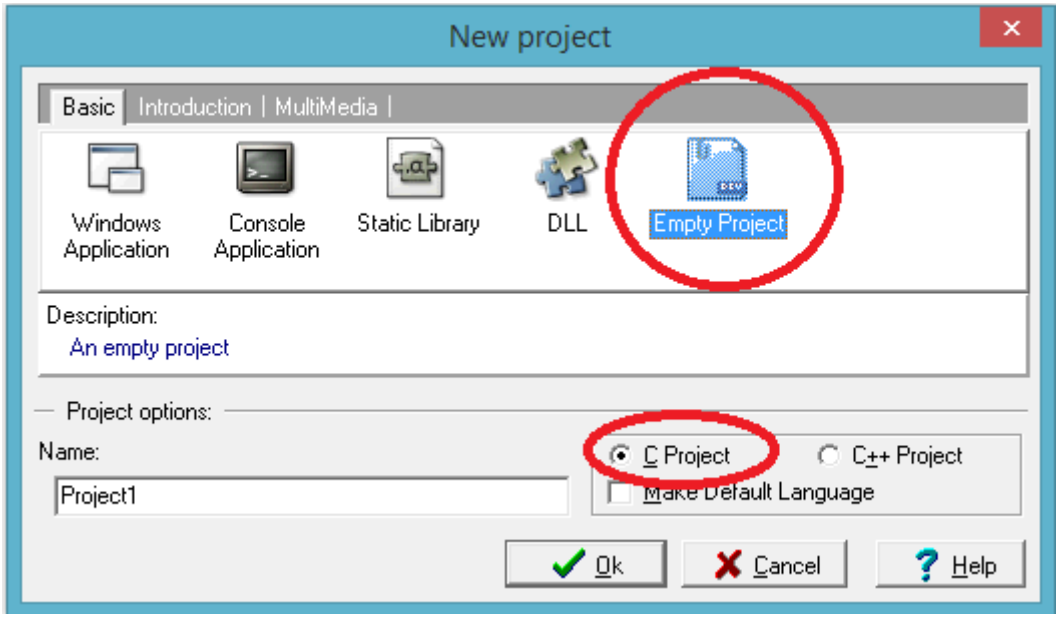
Step 1

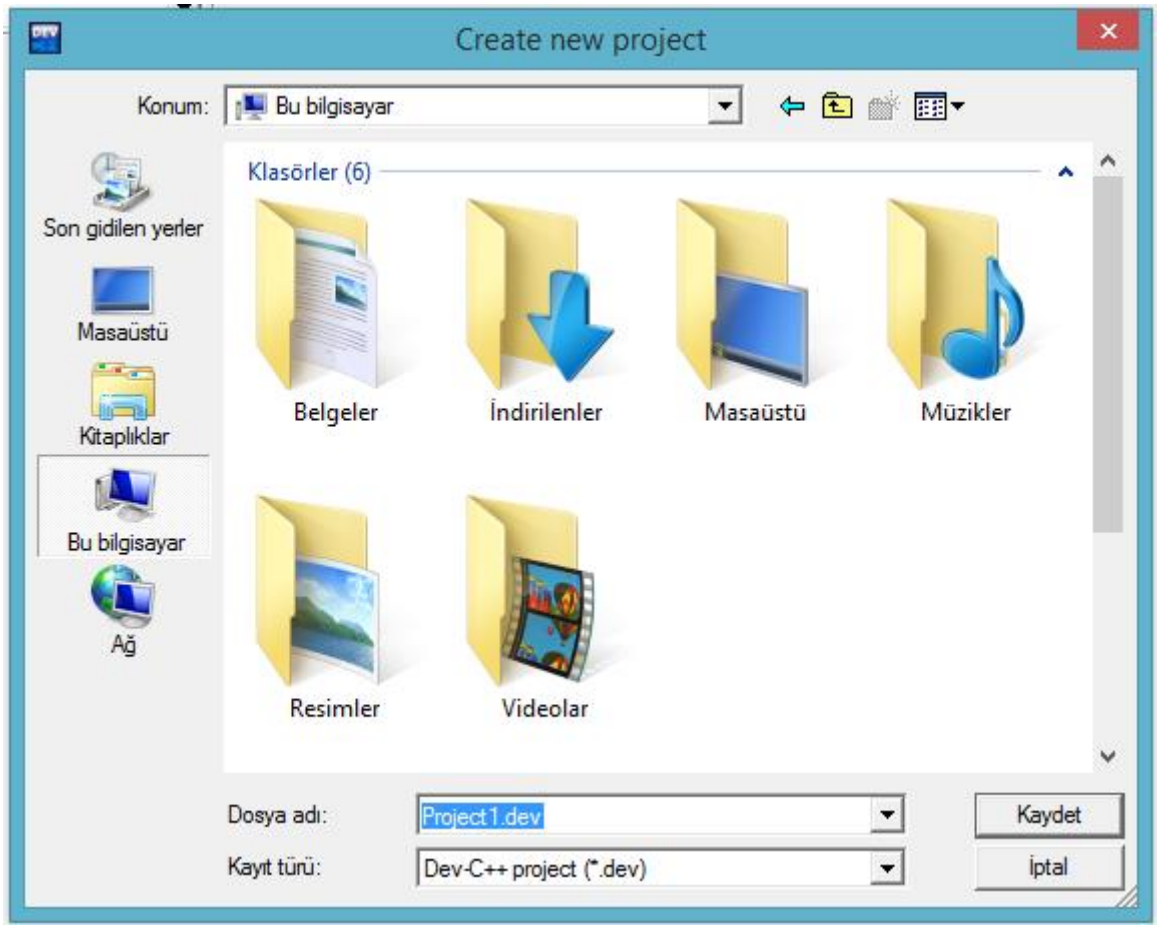
a) Open ***"Dev-C++"*** and click the icon for ***"Project"***.

b) Select ***"Empty Application"***, tick C file then specify the name of your project and click ***"OK".***

c) Choose the path to create the project. **It's a good idea to create a folder for the project.**

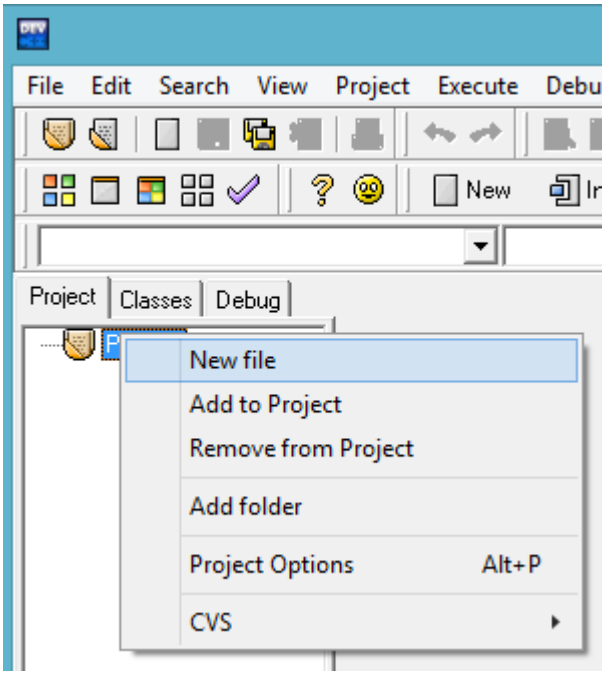


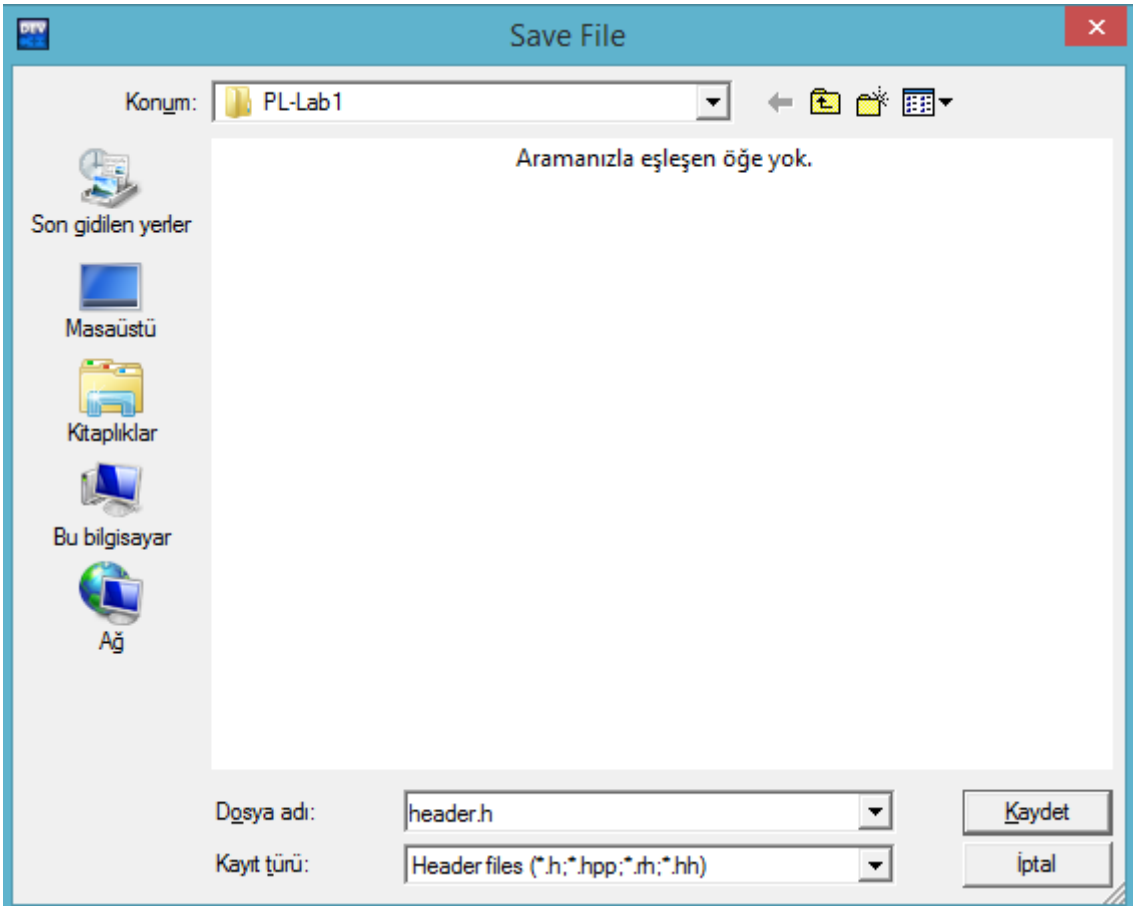




Step 2

Now right-click on the project you have just created in project explorer, and choose **"New File"**.

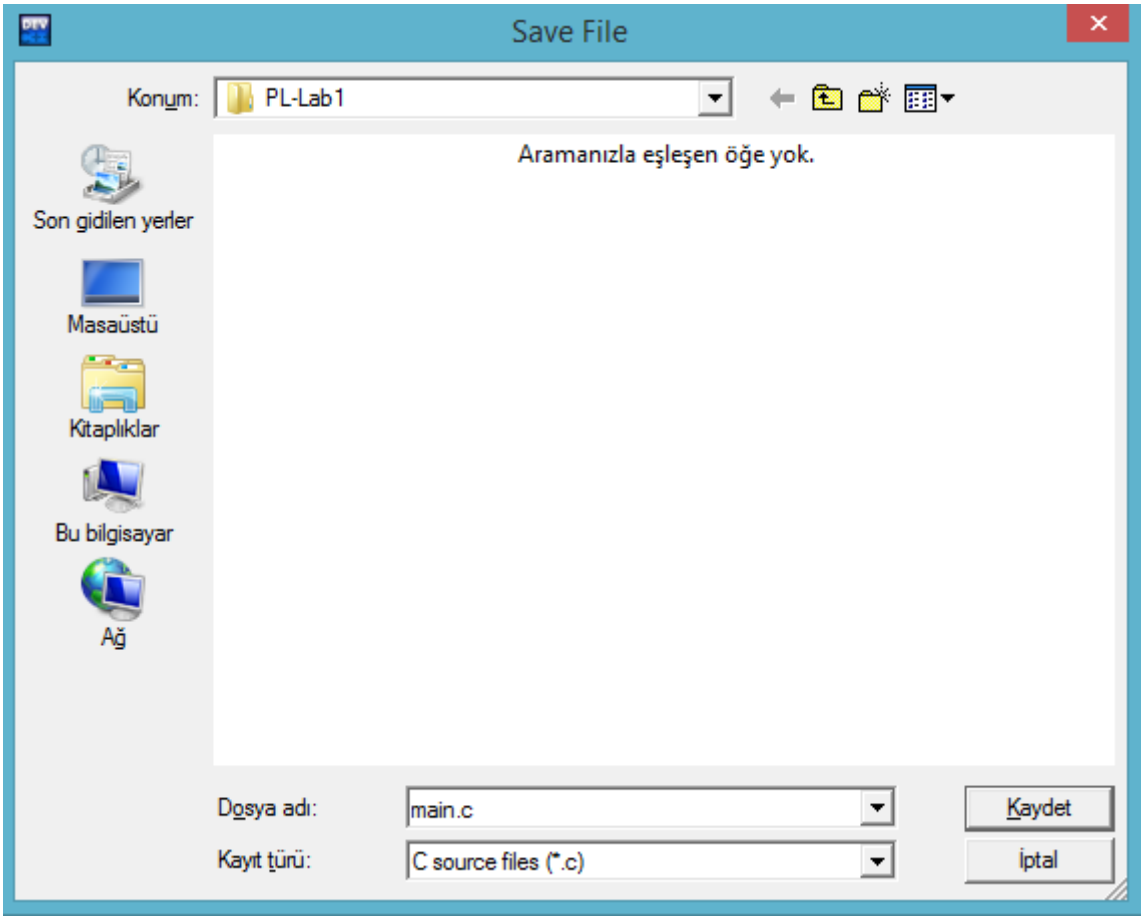




Create header file and write the following code into the header file. ***(between #define HEADER_H and #endif)***

```
#ifndef HEADER_H
#define HEADER_H

void WriteMyString(char*);


#endif
```

Step 3



Create a New File named **"main.c"** and write the following code into this file.

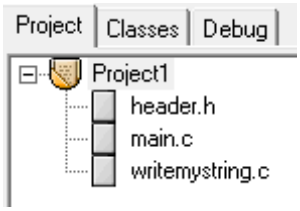
```
#include "header.h"
#include <stdio.h>
int main()
{
        printf("Running your C program...\n");
        WriteMyString("Compiled C");
        printf("Compilation is completed.\n");

        getchar();
        return 0;
}
```

Step 4

Add a new ***"C source file",*** and name it ***"writemystring.c"***. Write the following code into this file.

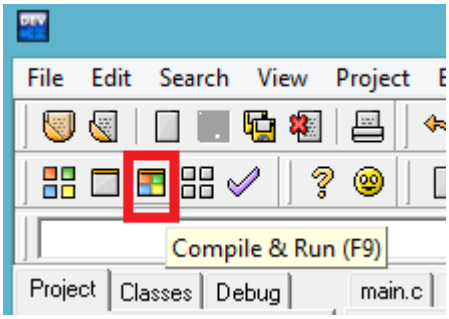
```
#include <stdio.h>

void WriteMyString(char *ThisString)
{
    printf("Incoming String: %s\n", ThisString);
}
```

Now your Project should look like this:



Step 5

Build and Run your code using the appropriate icon.



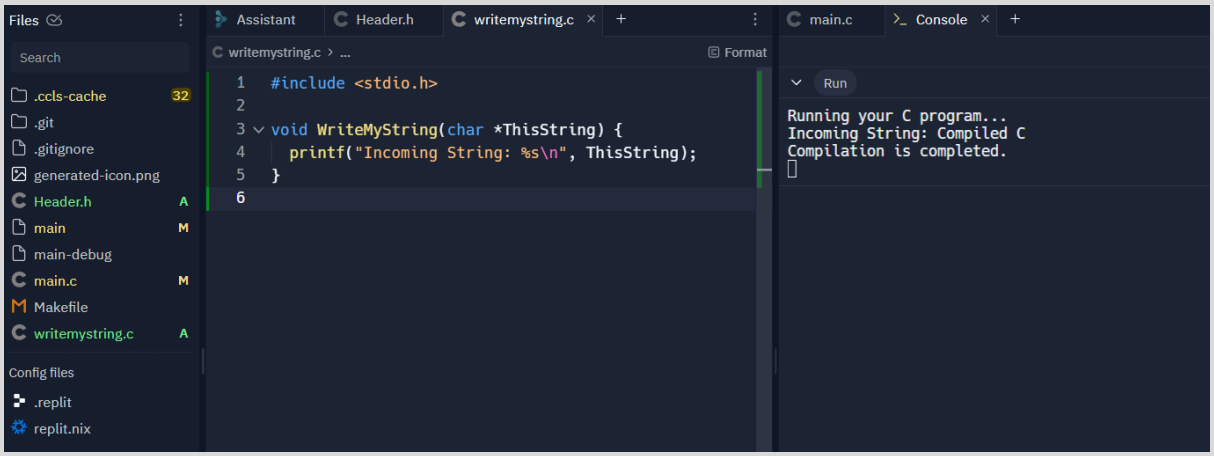
Step 6

Copy and paste your black console screen with the top directory info visible:

Running your C program...

Incoming String: Compiled C

Compilation is completed.



Step 7

Go to the project directory. Write the names of the files in this directory in the box.

Header.h
main
main.c
main-debug
Makefile
writemystring.c

Pure Interpretation

Pure interpretation is an implementation method in which programs are interpreted by another program called an interpreter.

Example 2

An example of programming languages which is implemented with pure interpretation is JavaScript. Apply the following steps, and implement the tasks.

Step 8

Open ***"Notepad++"*** from your ***"Start"*** menu.

Step 9

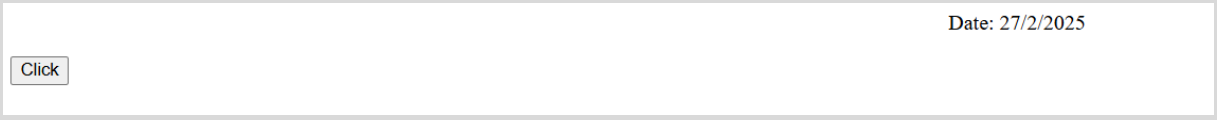
Write the following code into the notepad, and save the document as ***"Jslab.html"***. Select "***Save as Hyper Text Markup Language"***.

```
<html>
<head>
<script type="text/javascript">
function PrintDate() {
  today = new Date();
  document.write('Date: ', today.getDate(),'/', today.getMonth()+1,
'/', today.getFullYear());
}
function MyFunc(txt) {
   document.write(txt);
}
</script>
</head>

<body>
<p align="center">
<script type="text/javascript">
   PrintDate();
</script>
</p>
<form>
<input type="button" value="Click" onClick="MyFunc('JavaScript
Interpreted!');">
</form>
</body>
</html>
```

Step 10

Open the file with an internet browser. Which program does the interpretation operation?

The JavaScript engine within the browser interprets the code.



JavaScript Interpreted!

Hybrid Implementation Systems

Hybrid implementation systems translate high-level programs to an intermediate language designed to allow easy interpretation, and interpret the intermediate code.

An example of programming languages which is implemented with hybrid implementation is Java. Do the following steps, and implement the tasks.

Step 11

Open ***"Eclipse"*** from ***"start"*** menu, and create a ***"Java Project".***

Step 12

Open ***new class creation dialog***. Create a new class with ***public static void main(String[] args) option checked, and named javalab.***

Step 13

Write the following code into ***"main method",*** and run your code.

```
String output="Java bytecode is created";
System.out.println(output);
```

Step 14

Go to the bin directory of your project. What do you need to be able to run the program file?

To run the compiled Java program, you need the Java Virtual Machine (JVM) along with the Java Runtime Environment (JRE).

Step 15

What is **"pure interpretation"** ?

Pure interpretation is a method where the source code is executed directly by an interpreter without a separate compilation step.

Step 16

What is the difference between "***pure interpretation"*** and "***hybrid implementation"***?

Pure Interpretation: The source code is interpreted line-by-line at runtime.

Hybrid Implementation: The source code is first compiled into an intermediate language (bytecode) and then either interpreted or compiled further for execution.