**Canadian Institute of Technology**

Faculty of Economy  
Department of Business Administration and IT

****

**Morse Code Project**

A project submitted in partial fulfillment   
of the requirements for the **Fundamentals of Programming I course**   
in Ba and IT

**by**

Enio Stavre

Arion Dunisha

**Supervised by**

Evis Plaku

**ABSTRACT**

Perhaps the most famous of all coding schemes is the Morse code, developed by Samuel Morse in 1832 for use with the telegraph system. The Morse code assigns a series of dots and dashes to each letter of the alphabet, each digit, and a few special characters (e.g., period, comma, colon, and semicolon).

Separation between words is indicated by a space or, simply, a dot or dash.

In this project we aim to create an application that is able to translate an English phrase into Morse Code and vice versa. This application will contain 2 classes**, MorseProject** and **Main**.

**MorseProject** contains 2 methods (**EngToMorse** and **MorseToEng)** which are used for our translations, while class **Main** is used to run our application. We will take a closer look at these classes below.

**The Morse Project**

As we mentioned before this project consists of 2 classes, **MorseProject** and **Main.** We will start by taking a look at **MorseProject.**

**MorseProject**

This class is used for translating an English phrase into Morse code and a Morse code into an English phrase. In order to do this, this class will use 2 methods, **EngToMorse** and **MorseToEng.**

**EngToMorse** is our first method which is used to translate an English phrase into Morse code.

It achieves this by first creating a **HashMap,** since it stores keys to a corresponding value. This will let us associate letters, numbers and symbols with their corresponding Morse code. Afterwards we will create a Scanner object which will help us take user input. Then we create 2 String objects **eng** and **eng2. eng** will be usedto take the users input, while **eng2** will turn that input into uppercase in order for our application to read it. We then create String object **result** which will be used to store our translation. Afterwards we create a for loop which is used to get the index of every character in our String and in an iterative way, it translates our English characters into Morse code and then display it.

**MorseToEng** is our second method which is used to translate a Morse code into an English phrase.

It functions similar to our previous method. As above we create a **HashMap** but this time we associate Morse code with the corresponding letters, numbers and symbols. We will create a Scanner object which will help us to take user input. Then we create a String object **eng** used to take the users input and the String object **result** which will be used to store our translation. Afterwards we create a for loop which is used to split the individual Morse codes in our String and in a iterative way, it translates our Morse code into English characters and then display it.

**Main**

Class **Main** is used to run our application and let the user choose what operation he wants to perform.

We first use inheritance in order to use **MorseProject** methods. Afterwards we create a Scanner object in order to get user input. We create a while loop so the user can run the application as many times as he wants. The user will be presented with 3 choices, translate English to Morse Code, translate Morse Code to English or exit the application. If the user inserts 1, method **EngToMorse** will run. If the user inserts 2, method **MorseToEng** will run. If the user inserts a number bigger then 3, the message "Wrong input please try again!" will be displayed since we only have 3 options. At last if the user inserts 3 the application will stop running. We also use exceptions in case the user inserts an invalid input. If for example the input is a String object instead of an Integer the message "Invalid inputs please try again!" will be displayed.