CMIS 315 Project 1 Speeding Ticket Fine Calculator

Development Approach Plan

Version 1.0

01/25/2015

Contents

[1. Introduction 1](#_Toc409960318)

[2. Overview 1](#_Toc409960319)

[3. Assumptions/Constraints/Risks 1](#_Toc409960320)

[3.1 Assumptions 1](#_Toc409960321)

[3.2 Constraints 1](#_Toc409960322)

[3.3 Risks 1](#_Toc409960323)

[4. Development Approach 1](#_Toc409960324)

[4.1 Development Methodology 1](#_Toc409960325)

[4.2 Lifecycle Management & Transition Approach 2](#_Toc409960326)

[4.3 Methods & Tools 2](#_Toc409960327)

[4.4 Difficulties Encountered 2](#_Toc409960328)

[4.5 Testing 2](#_Toc409960329)

[5. Weaknesses and Improvements 5](#_Toc409960330)

[5.1 Weaknesses 5](#_Toc409960331)

[5.2 Improvements 6](#_Toc409960332)

List of Tables

Table 1: Development Approach Processes ……………………………………………………2

Table 2: Test Data Test Run 1 ………………………………………………………………….3

Table 3: Test Data Test Run 2 ………………………………………………………………….4

List of Figures

[Figure 1: Screen Shot Test Run 1 3](#_Toc409960060)

[Figure 2: Screen Shot Test Run 2 5](#_Toc409960061)

# Introduction

This Development Approach Plan describes the approach for development of the Speeding Ticket Fine Calculator. This document will outline the development approach, solution design, difficulties encountered, testing strategy, system weaknesses, and possible improvements. This document is written for Professor Omar Zevallos, CMIS 315 instructor.

# Overview

The purpose of this project was to create a program that uses input from the user to calculate total fines for speeding violations. The program should allow the user to enter multiple tickets during a single run of the program and output the total fine calculated from all tickets entered. After the information for each ticket is entered the program should allow the user to input another ticket or exit the program. The program had to make use of a SpeedingFineCalculator class with predefined variables and methods.

# Assumptions/Constraints/Risks

## Assumptions

None

## Constraints

The program must make use of the SpeedingFineCalculator class, with predefined variables and methods, to calculate fines.

## Risks

None

# Development Approach

## Development Methodology

I used the waterfall approach for software development for this project. I concluded this approach at the coding and testing phase, as the software will not be implemented, and operational support will not be needed. I used this methodology because:

1. The project requirements where clearly defined and unambiguous;
2. The requirements for this project are stable and not subject to change;
3. The project had clear objectives and solutions.

I divided the project into 5 parts:

1. SpeedingFineCalculator.h class header
2. Menu.h class header
3. SpeedingFineCalculator.cpp class - calculates the total fine taking the parameters of processing fee, zone, speed limit, and actual speed.
4. Menu.cpp class – contains methods for each menu option as well as input sanitizing methods.
5. Main.cpp – main method for the project, handles the repeating of menu options.

## Lifecycle Management & Transition Approach

I started with the requirements definition phase, as there was no need for an initial investigation. Some of the requirements were defined in the project description; the rest became apparent during the writing of the initial algorithm. After defining all of the requirements I started with the basic system design by hand writing class diagrams and the basic algorithm. Once I had a clear picture of the system, I began to code the classes. Once each class was defined, I coded the main method. Once I had completed all coding I compiled and ran my program in the debugging mode to begin testing.

## Methods & Tools

Table 1: Development Approach Processes

| Process | Tools & Techniques |
| --- | --- |
| Requirements definition | Paper and Pencil |
| Class diagrams | Paper and Pencil |
| Coding | Microsoft Visual Studio 2013 |
| Testing | Microsoft Visual Studio 2013 |
| Approach Document Creation | Microsoft Word 2010, Microsoft Paint |
| Assignment Submission | Microsoft Word 2010, Notepad |

## Difficulties Encountered

I did not encounter any difficulties implementing my solution because of the linear process I used for development. I did not move onto the next phase of development until all concepts of the current phase were clearly defined and understood.

## Testing

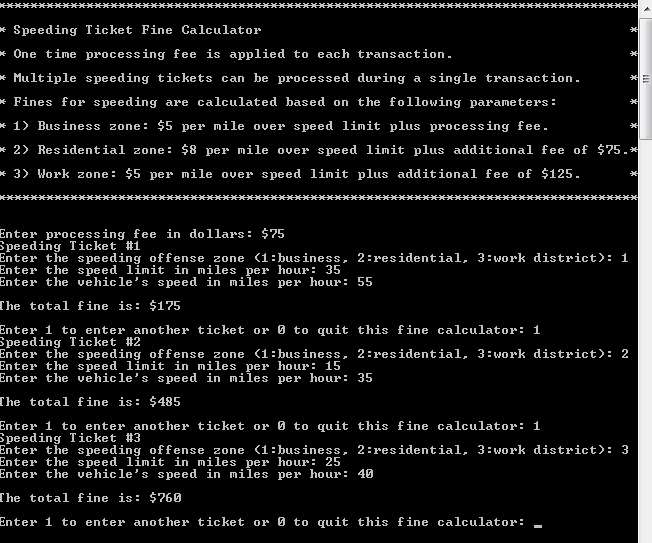
I began testing the program by first compiling the program in Microsoft Visual Studio 2013 and then running in the debugging mode. I entered a processing fee when prompted and then entered data for three speeding tickets choosing each of the three zones to observe the output. The below table shows the test data input along with the expected and actual output.

Table 2: Test Data Test Run 1

| Processing Fee | Zone | Speed Limit | Actual Speed | Expected Output for Total Fine | Actual Output for Total Fine |
| --- | --- | --- | --- | --- | --- |
| 75 | 1 business | 35 | 55 | 175 | 175 |
| - | 2 residential | 15 | 35 | 485 | 485 |
| - | 3 work district | 25 | 45 | 760 | 760 |

The below screen shot shows the output of this run.

Figure 1: Screen Shot Test Run 1

**

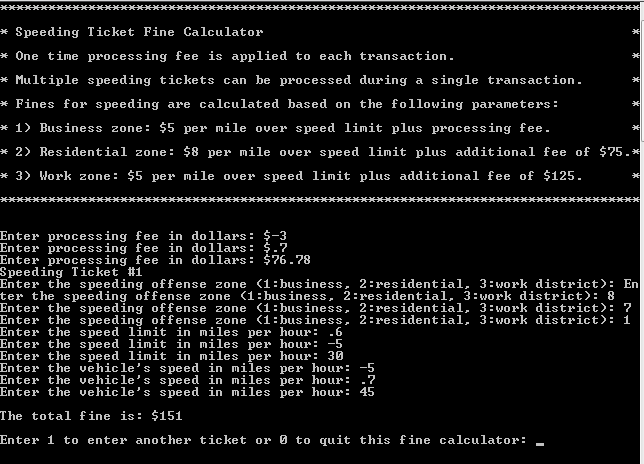
For the second test run I entered several incorrect data types to test my data validation methods. The below table shows the data entered during the second test run with the expected and actual outputs.

Table 3: Test Data Test Run 2

| Processing Fee | Zone | Speed Limit | Actual Speed | Expected Output | Actual Output |
| --- | --- | --- | --- | --- | --- |
| -3 |  |  |  | Prompt user to enter processing fee | Prompt user to enter processing fee |
| .7 |  |  |  | Prompt user to enter processing fee | Prompt user to enter processing fee |
| 76.78 |  |  |  | Prompt user to enter data for Ticket #1 | Prompt user to enter data for Ticket #1 |
|  | 8 |  |  | Prompt user to enter zone | Prompt user to enter zone |
|  | 7 |  |  | Prompt user to enter zone | Prompt user to enter zone |
|  | 1 |  |  | Prompt user to enter speed limit | Prompt user to enter speed limit |
|  |  |  |  | Prompt user to enter speed limit | Prompt user to enter speed limit |
|  |  |  |  | Prompt user to enter speed limit | Prompt user to enter speed limit |
|  |  |  |  | Prompt user to enter vehicle’s speed | Prompt user to enter vehicle’s speed |
|  |  |  | -5 | Prompt user to enter vehicle’s speed | Prompt user to enter vehicle’s speed |
|  |  |  | .7 | Prompt user to enter vehicle’s speed | Prompt user to enter vehicle’s speed |
|  |  |  | 45 | Total fine $151 | Total fine $151 |

The below screen shot shows the output of this run.

Figure 2: Screen Shot Test Run 2

**

# Weaknesses and Improvements

## Weaknesses

The weaknesses that I found in the system are:

1. If the user enters a float value for the processing fee, the value is parsed to integer type and the decimal value is dropped off. This means that processing fees can only be whole number integer values, and if processing fees are ever of float type, than accuracy is lost during the fine calculation.
2. If the user enters a bad data type, the previous input statement is repeated without informing the user what error they made. This may be confusing to some users.

## Improvements

To improve the program I would do the following:

1. Change the data type for processing fee and fine to double type to allow for non-whole number values to be processed.
2. Include error statements to inform the user what error they made and what is the correct data type to enter, and then repeat the input statement.
3. Equations
4. Calculate MPH over speed limit (difference):

Difference = Actual Speed – Speed Limit

1. Calculate speeding fine based on zone multiplier and difference in speed:

Fine = Difference \* Zone multiplier

1. Calculate total fine for offense by adding speeding fine to additional zone fee and processing fee

Total Fine = Fine + Zone Fee + Processing Fee