Alan Haverty DT211/3
Network Programming

TxtClock Design Notes

Name: Alan Haverty

Course/Year: DT211/3

Student ID: C12410858

DIT Email: alan.haverty@student.dit.ie

Module: Network Programming – CA1

Document: TxtClock Design Notes

Network Programming CA1 Design Notes

Need to display time as an English sentence in the Format "it is twenty minutes past ten o'clock".
Also want to implement custom values such as “a quarter to” and “half past”.
Need to connect with an NTP Server pool also.

Program takes argument in 24hr time format: hh:mm:ss or hh:mm
If the seconds (ss) are not defined, they are set as 0

If there is no argument defined, the program requests the time from an NTP server, if the NTP server can't be reached after a certain amount of time, the local system time is used instead.

Program Flow:

If the user supplies a time:
Output the time to the console in English.

If the user does not supply a time:
Request a time from an NTP server.
If the NTP server does not reply within 10 seconds, use the machines local system time instead

Validating 24hr Time

Validate a users input using a regular expression
([01]?[0-9]|2[0-3]):([0-5][0-9])(:([0-5][0-9]))?

First group allows a range from 00 - 23
Second group allows a range from 00 - 59
Third optional nested group allows 00 - 59
The third group is nested to make extracting the time easier without having to perform string manipulation on the colon character by skipping group 3 and taking group 4 as the seconds instead.

Program Structure

Use various classes to handle functions like regex validation, time extraction from a string and getting the time from the NTP remote server.

Use a model class to hold the projects time objects, basically a Calendar object wrapped with constructors, getters and setter.

- Implement a method to set the time object to the current system time for the case where the NTP server fails and the local system time is used instead.

Dynamically extract the integer to word data through an external resource file like a text file. Parse the key and values into a hashMap to allow easy conversions.

Logic & algorithm behind which English words to use:

Some Possibilities:

13:20 It is twenty minutes past one o'clock
14:15 It is a quarter past two o'clock
15:30 It is half past three o'clock
15:35 It is twenty-five minutes to four o'clock
16:45 It is a quarter to five o'clock
17:00 It is five o'clock
23:59 It is one minute to twelve o'clock
00:00 It is twelve o'clock

Common Components:

It is [<minute> || <minutes> || <"a quarter">] [<tense> || <no tense>]
[<hour> || <hour + 1> || <1>] o'clock

Conditions on the minute:

If the minute is 0 then:
It is [<hour>] o'clock

if the minute is equal to 15:
It is [<"a quarter">] [<tense = PAST>] [<hour> || <hour-12>] o'clock

if the minute is equal to 30:
It is [<"half">] [<tense = PAST>] [<hour> || <hour-12>] o'clock

If the minute is greater than 0 and less than 31:
It is [<minute> || <minutes>] [<tense = PAST>] [<hour> || <hour-12>] o'clock

If the minute is equal to 45:
It is [<"a quarter">] [<tense = TO>] [<hour + 1> || <hour-12+1> || <"1">] o'clock

If the minute is greater than 30 and less than 60:
It is [<minute> || <minutes>] [<tense = TO>] [<hour + 1> || <hour-12+1> || <"1">] o'clock

Conditions on the hour:

If the tense is PAST or HOUR then the hour is:
[<hour> || <hour-12>]

If the tense is TO then the hour is:
[<hour + 1> || <hour-12+1> || <"1">]

If the tense is TO and the hour is 23:

[<"1">]

Note:
[<hour-12>] indicates a conversion from 24 hours to 12 hour word

Further on mapping:
Using a text file to store the key and values for the time words, they will be mapped into the program using a hashmap (e.g 1 = one, 12 = twelve). This will also include custom key and values such as 101 = a quarter, 102 = half

Update to time message:
Implement seconds in the English sentence, where 10:21:00 (Seconds = 0) outputs a message such as “It is exactly twenty-one minutes past ten”. But 10:21:15 outputs something like “It is twenty-one minutes past ten and fifteen-seconds”.

Possible Tests:
- Test with more than 1 argument
- Test the program fails gracefully with incorrect arguments
- Test the program can run and exit successfully giving a valid argument and also without any arguments
- Test that valid 24hr times are accepted through the validation function
- Test that the expected English sentence is produced for various valid 24hr times
- Test that known invalid 24hr times return false for the validation method
- Test edge cases around hours, the sentence tense and times around midnight (e.g. 23:59, 00:00)
- Test without the text file in correct folder, handle errors and console message
- Test with the correct text file name and location but with incorrect data within
- Verify the NTP time retrieved is correct
- Verify the program runs successfully without an internet connection with no arguments