

Java VT100 Terminal Emulator Project Plan

Group Three

CMSC 495

Professor Howard

26 March 2017

# Table of Contents:

[Table of Contents:](#_bvp5sfqzd68)

[Description](#_bec1lxx38811)

[Project Requirements](#_sne5qgxgcs8b)

[Ancillary functions](#_bkmroshijcxb)

[Use Scenarios](#_v8lsdnyi0h74)

[Use Case Example](#_8wualuy6nkrb)

[Software requirements](#_vvqed0c1x5k3)

[Hardware Requirements](#_1uswoouvcvy4)

[Sample Input](#_8sopdxpv2f2g)

[Project Milestones](#_4m5xz47z8uza)

[Limits and Exclusions](#_b636wh1r8ouk)

[Hosting/Collaboration](#_ifidz1ziat5y)

[Work Breakdown Structure (WBS):](#_44z67unwxpaq)

[Project Communication Plan](#_kotuym8oytnm)

[Team:](#_7oseauiekdh9)

###### 

# Description

This project is the creation and implementation of a VT100 Terminal Emulator. The project makes use of ANSI control sequence formatted text to display relevant information in the proper location on the screen.

# Project Requirements

This project should allow the user to view ASCII data that is formatted using ANSI control sequences in a manner that reflects these parameters. Furthermore, the following features should be included:

* Detect escape sequences and listen for control words to follow.
* Make use of formatting codes like “m codes” to properly format text to display (i.e. blinking text, bolded, etc).
* Detect the “bell” character commonly used to refresh the screen.
* Allow the user to input data using a 15-key pad which includes 1- 0, directional keys, and Enter.
* Place a grid of transparent buttons over the formatted text to allow the user to interact with displayed objects.
* A drop down list (or similar) should allow the user to connect to the system’s serial ports in order to receive the ASCII data.
* A refresh button should be included to allow the user to clear the screen of leftover characters post page change.

# Ancillary functions

The following functionality is inherited due to the nature of the application:

* Platform independant system serial port enumeration.
* ASCII data streaming through system console.
* Serial input device.

# Use Scenarios

The program’s primary user will likely be using it to control a piece of hardware such as an antenna or other legacy servo control system. These devices often relied on dedicated consoles to interact with the system and as such could greatly benefit from a product that is relatively platform independent and adaptable to evolving technology.

Other users might include those who wish to enumerate their system’s serial ports, those looking to send data across a serial port, and those who are looking to view ASCII data without interacting with it.

# Use Case Example

Pre-condition: The system is booted and the application is accessible to the user.

Post-condition: The system should close smoothly with no errors and release all system resources.

Action profile: User is attempting to control an antenna that he knows uses the ANSI formatted ASCII data to display a control page.

Sequence of events:

1. User opens the application
2. User selects the serial port that the antenna is connected to and clicks “Connect” button.
3. User observes the data displaying on the page and makes a control selection.
4. Antenna moves in the requested direction.
5. Display page is continuously updated with relevant data.
6. User exits the program when finished.

# Software requirements

This program will make use of the Java Simple Serial Connector library - jSSC-2.8.0 Release version (24.01.2014) for enumerating and connecting to the system's serial (COM) ports. Additionally, we will use the freeware RealTerm to send the ANSI formatted data to our program for the purpose of testing.

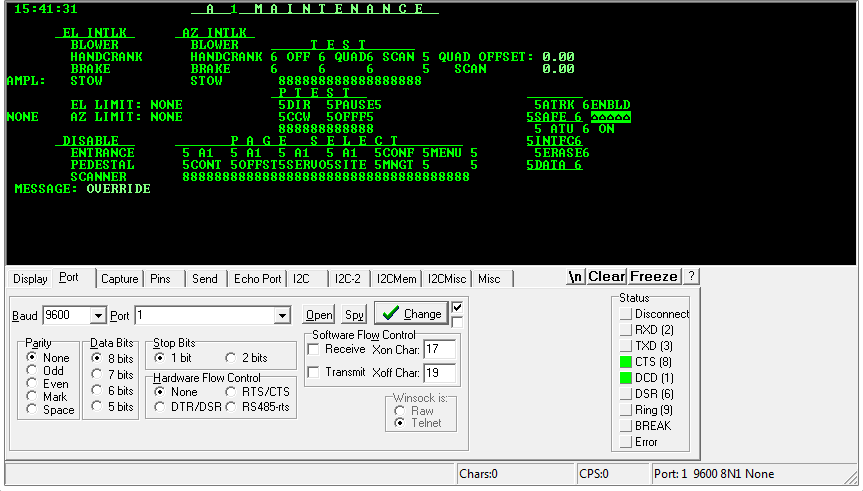
# Hardware Requirements

A computer with some configuration of 2 serial ports is needed for this application. These can be physical COM ports, USB to Serial devices, or simply virtual COM ports. In the real world, the program would be run on any computer (MAC, Linux, Windows) and connected to a device that sends ANSI formatted data in real time. For the purpose of this test, the application will connect to a virtual serial port. At the other end of the virtual connection will be a RealTerm console into which the sample input file will be pasted.

# Sample Input

This is the file we will input into the program. It is a menu page for an antenna controller.

+[0;2p[01;24H[1;4m A M A I N T E N A N C E [m[03;07H[4m EL INTLK [m[03;22H[4m AZ INTLK [m[08;34H[4m P T E S T [m[12;07H[4m DISABLE [m[12;22H[4m P A G E S E L E C T [m[04;09HBLOWER[m[05;09HHANDCRANK[m[06;09HBRAKE[m[07;09HSTOW[m[04;24HBLOWER[m[05;24HHANDCRANK[m[06;24HBRAKE[m[07;24HSTOW[m[13;09HENTRANCE[m[14;09HPEDESTAL[m[15;09HSCANNER[m[09;09HEL LIMIT:[9;30H[0m[10;09HAZ LIMIT:[10;30H[0m[5;55HQUAD OFFSET:[6;57HSCAN AMPL:[1;1H[1m[1;11H[m[16;2HMESSAGE:[16;35H[m[3p[04;34H[4m T E S T [m[05;34H6 OFF 6 QUAD6 SCAN 5[m[06;34H6 6 6 5[m[07;35H888888888888888888[09;35H5DIR 5PAUSE5[10;35H5CCW 5 OFF[m5[11;35H888888888888[13;23H5 A 5 A 5 A 5 A 5CONF 5MENU 5[14;23H5CONT 5OFFST5SERVO5SITE 5MNGT 5 5[15;23H888888888888888888888888888888888888[8;66H[4m [m[9;67H5ATRK 6[m[10;66H[4m5SAFE 6[m[11;67H5 ATU 6[m[12;66H[4m5INTFC6[m[13;67H5ERASE6[m[14;66H[4m5DATA 6[m[2p[2p[01;29H1[13;26H1[13;32H1[13;38H1[13;44H1[3p[5;34H6[5;41H [6;34H6[6;41H [2p[3p[5;34H[7m[5;41H[m[6;34H[7m[6;41H[m[2p[10;36HCCW[10;42H[0;0mOFF[6;44H [06;67H 0.00[05;67H 0.00[4;08H[0;0m[5;08H[0;0m[6;08H[0;0m[7;08H[0;0m[4;23H[0;0m[5;23H[0;0m[6;23H[0;0m[7;23H[0;0m[13;8H[0;0m[14;8H[0;0m[15;8H[1;5m[8;19H[0;0m[9;19H[0;0mNONE [10;19H[0;0mNONE [09;74H[0;0mENBLD[m[10;74H[0;0;7m[m[11;75HON [1;2H15:41:31[16;11H[1;1mOVERRIDE [06;67H 0.00[05;67H 0.00[4;08H[0;0m[5;08H[0;0m[6;08H[0;0m[7;08H[0;0m[4;23H[0;0m[5;23H[0;0m[6;23H[0;0m[7;23H[0;0m[13;8H[0;0m[14;8H[0;0m[15;8H[1;5m[8;19H[0;0m[9;19H[0;0mNONE [10;19H[0;0mNONE [09;74H[0;0mENBLD[m[10;74H[0;0;7m[m

We will parse this message in and make use of the standard ANSI control sequences contained in the text to format it and display it on the screen. This is a rough example of how the text should look:

As this is the output from RealTerm, there are several components missing. Additionally, the digits surrounding the words should be displayed as boxes.

# Project Milestones

This is a list of significant events in the project that occur at a point in time. The milestone schedule shows only major segments of work.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| * Project Plan |  |  | - | Mar 26, 2017 |
| * Create a comprehensive design of the classes and methods and provide pseudocode (or better). |  |  | - | Apr 9, 2017 |
| * Obtain a basic level of functionality |  |  | - | Apr 16, 2017 |
| * Complete Code |  |  | - | Apr 30, 2017 |
| * Complete documentation and commenting within code |  |  | - | May 2, 2017 |
| * Submit Final Project |  |  | - | May 7, 2017 |

# Limits and Exclusions

There is the possibility that this application might be resource intensive due to the fact it will be “painting” the text to a canvas every time new strings are sent. RAM and CPU requirements will become apparent upon further coding/testing.

# Hosting/Collaboration

As required per the syllabus, the code will be hosted in a GitHub repository to allow the users to collaborate on coding and the professor to review the code.

# Work Breakdown Structure (WBS):

Once the scope and deliverables have been identified, the work of the project can be successively subdivided into the smaller and smaller work elements. WBS is the map of the project.

VT100 Project

Phase 1: Research

Phase 2: Implementation

Acquire software solutions

Import JSSC library/discover relevant methods

Construct documentation package

Design class structure

Develop code

Final project submission

# 

# Project Communication Plan

What information need to be collected and when?

Who will review the information?

What methods will be used to gather and store information?

What are the limits, and who has access to certain kind of information?

When will the information be communicated?

How will it be communicated?

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| What information | Target audience | When | Method of communication | Provider | Limits, access to information |
| Coding, GitHub | Team Members | Weekly | Email, GitHub repository | Joseph Bowe | Accessible by team. Currently some trouble getting github to work for everyone. |
| Doc report, Design | Team Member / Developer | Weekly | Email | Adele Janlou, Andrew Rudnev | Accessible by team. Should be secure. |
| Development | Team Member | Weekly | Email | Haley Adams | Accessible by team. |
| Issue report | Team member/ Developer | Weekly | Email | Haley Adams | Accessible by team. Issues shouldn’t be made public. |
| Milestone report | Team Member | Weekly | Email | Adele Janlou, | Accessible to team, investors. |
| Technical report | Team Member/ Developer | Weekly | Email, Google Docs | Andrew Rudnev | Accessible to team. Most important for lead developer. |

# Team:

Joseph Bowe: josephbowe10@gmail.com - Design/Developer

Haley Adams: hladams19@gmail.com - Developer/Lead Tester

Adele Janlou: adele.janlou@gmail.com - Documentation/Design

Andrew Rudnev: andrewrudnev@gmail.com - Developer/Documentation