CSC 260 – Object Oriented Design

Jhet Birchem

Jason Jenkins

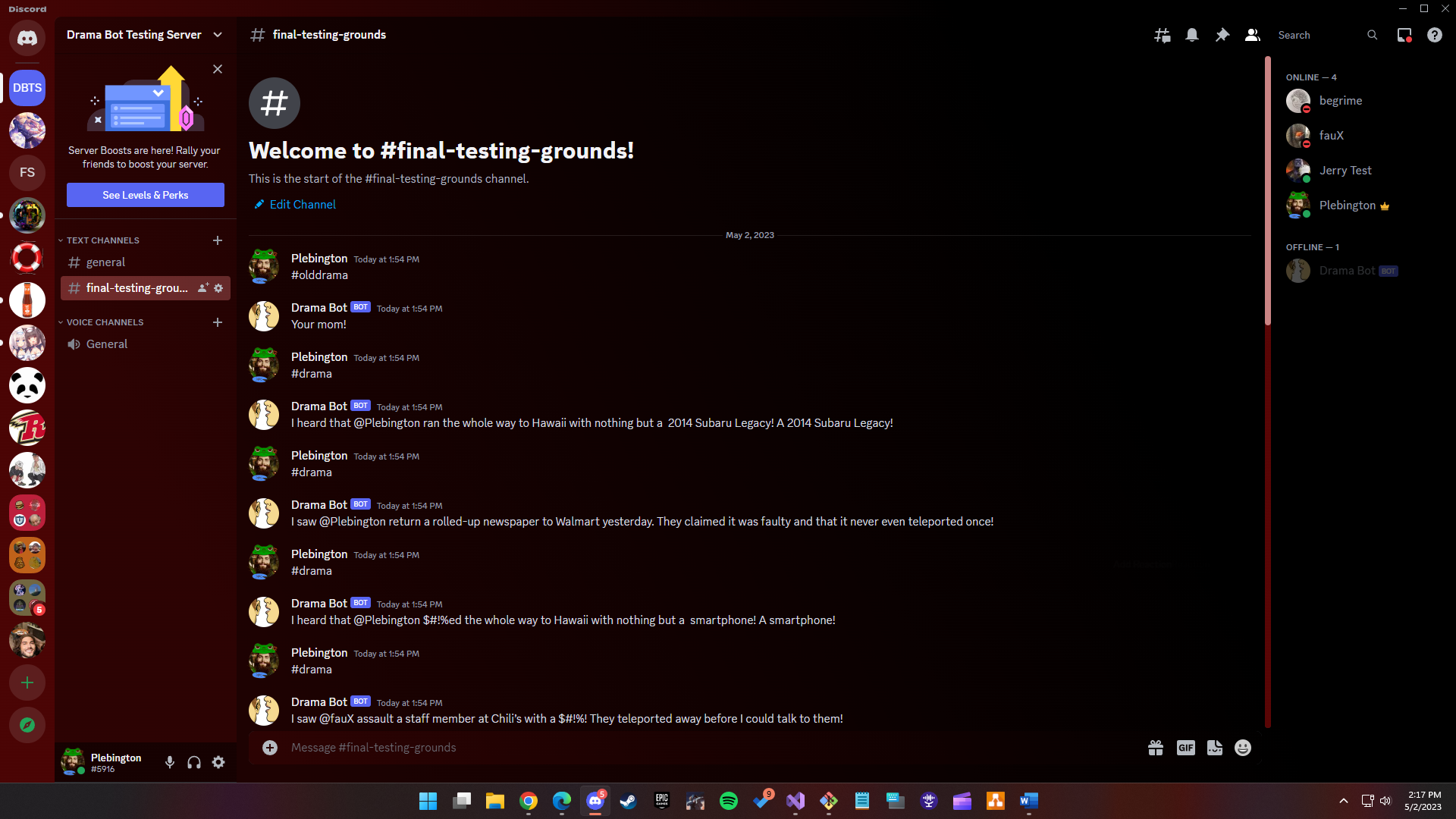
5/2/2023

**Final Project (Discord Drama Bot)**

First of all, before I get too far into anything I wanted to mention that my program utilizes the following NuGet packages extensively, and is not possible without them:

* DSharpPlus (Version 4.4.0)
* DSharpPlus.CommandsNext (Version 4.4.0)
* DSharpPlus.Interactivitiy (Version 4.4.0)

With those credited, onto how I met the project requirements.

* Completeness/Classes and their components – Unique classes aside from the “main” class.
  + Bot.cs
  + DramaCommands.cs
  + OldDramaCommand.cs
  + ConfigJSON.cs
* Appropriate Complexity
  + While I’m not qualified to say if this was appropriate or not, I did find it a challenging feat to figure out how to use someone else’s API for the first time. A.K.A., it was my first time not using basic features or Visual Studio defaults.
* Version Control
  + I used GitHub to push frequent changes I was making to the project. Link: <https://github.com/Jhet-Birchem/DISCORD-BOT-PROJECT-CSC-260>
* Evidence of usage
  + I shared the repository and made it public. Due to Discord’s security with the bot token, it was reset on Discord’s end automatically, meaning that I cannot upload the token.
* Testing
  + Testing included me using 2 different discord accounts in a server to see how the bot worked. This means I was using the #drama command with various amounts of arguments.
  + The results were good. The Mad-Libs style of sentence structure worked well. I’m thankful for my idea of sticking to shorter sentences, as getting the program to output multiple paragraphs could have been a nightmare. Here are some images of me testing the bot out in my testing server:
  + 
  + A screenshot of a computer

    Description automatically generated with medium confidence
  + A screenshot of a computer

    Description automatically generated
* Testing – Continued
  + Initially, my command for #drama only resulted in a childish response of “Your mom!” from the bot. This was to check that the command was working. Then, the hardest part to get to work was the generic #drama command that would pick a random user, a random noun, and a random verb. I eventually got the bot to randomly pick a user on the server and from there it was not hard to do the ability to take in a name as an argument.
  + I was hoping that the @ symbol added to the front of each name would ping the user if they existed within the server. This I was unsure what the API wanted me to do otherwise and decided that just an unhighlighted @ symbol was sufficient for my satisfaction.
  + I also was disappointed in the abilities of randomizing user, noun, and verb selection. Since it was time based, I got more diverse results by adding a delay into the code. This, unfortunately, may never be fixed, as it’s still a pseudo-random way of producing unique results.
* Inheritance
  + Both the OldDramaCommand class and the Drama Commands class inherit from BaseCommandModule for command functionality.
* Encapsulation
  + The Bot class’s first 3 items utilize getters and private setters and are necessities for allowing the program to add functionality and logic to the bot that appears online in the discord server.
* Polymorphism
  + Depending on the number of arguments a user provides when using the #drama command, there is method overloading to take over for a matching argument count (from 0 up to 3).
* Abstraction
  + While not something made by me, the BaseCommandModule interface is a core backbone to making commands for my bot program.
* More Advanced Topics:
  + A config.JSON file is used for a bot token and command prefix.
  + My code is heavily commented, which hopefully allows anyone viewing it to see what each part does and why it’s there. I also used regions to help give clarity in the lengthy drama command class.