CPSC 219: Introduction to Computer Science for Multidisciplinary Studies II

Project – Demo 1 Guidelines

Weight: 10%

Demo grading points

- Program is procedural (10)
    - static functions launched from main()
    - data is stored in basic data structures like Arrays/ArrayList/HashSets/HashMaps (not objects), project should not have additional classes/objects outside of using internal java one's
    - proper use of public/private functions
    - proper use of final (constants)
    - proper use of static
- An input menu (5)
    - User has clear messages of current stage of menu (printed to System.out)
    - Menu is well laid out (spaces vertical and horizontal)
    - Can type in input in System.in (terminal)
    - Pick an option for interaction by typing
    - Clear feedback on success of action
    - Multiple choices (input, view stored data, request calculated data)
- Data is stored in data structures (5)
    - Proper use of primitives/objects for the type of data stored (int versus double, Integer versus int)
    - Correct data structure chosen (ex. using arrays for strictly sized collection, ArrayList for flexible data)
    - Use Collections/Arrays properly to interact with stored data for things like printing/etc.
- View data (10)
    - [2] The stored data can be viewed in its regular stored state (ex. a table of entries)
        - Not every user input needs to be visible. For example you could be storing sports statistics. There is no need to record every goal added to be tracked. If you are just incrementing a stored total then we just need to be able to see the stored total has increased.
    - [8] Four new features of data can be accessed. For example, different sorted views of the data. A summation of data features. A prediction of future data. A warning that something was missed or that the rate of something is too low.
- JUnit Testing (out of 5)
    - Unit tests for functions (non-input/non-output/non-menu) functions
- Gitlab Usage (out of 10)

    - Gitlab account exists, private project exists, at least 1 commit, small commits, both partners have a commit, (5) regular commits
- Style/Commenting (out of 5)
    - Name/Date/Tutorial, Functions commented, Javadoc, Inline commenting, doesn't use inline conditionals, limited magic numbers, don't change function names, don't change filenames, etc.
- Partnership penalties
    - During the demo the TA will at times ask different members of your group (partnership) to describe how something works. Each partner should expect to have contributed to unit testing, git commits, commenting, and program functionality. In general the penalties will be
        - -5 Partner is judged to have not contributed in one area (ex. git, junit, commenting, code)
        - -10 Partner didn't contribute in two areas
        - -15 partner didn't contribute in three areas
        - -20 partner didn't contribute in all four areas
    - Contribution penalties are recorded separately for each student. Judgement is made by TA on basis of students being able to explain something about part of code that is being viewed in more detail than just re-iterating the readable syntax.

Example program. I decide to make a CWHL (ice hockey) statistics tracking program. I make a System.in menu that allows me to
Track basic data
1) add a Team,
2) add a player to a team with a name, birthdate, position, and jersey number,
Add additional data
3) add a goal to a player
4) add an assist to a player
5) add a save to a goalie
6) add a shot on goal to a goalie
Output General
7) ask for all players to be printed
Output Special
8) ask for the top 5 goal scorers
9) ask for the top goalie in save percentage
10) recommend line up of 2 defenceman, 1 goalie, and 3 forwards based on being top players
11) list of players over a certain age

I can choose how I store this data. There are many valid ways to use things like ArrayList/Arrays/HashMaps/HashSets to do so.

Maybe an ArrayList of Strings for team names. Then an ArrayList of ArrayLists of String data for personal info about a player. Each player must have a team which is found in my list of teams.

I might store stats for players in a HashMap for goalies and a HashMap for skaters. I might use a key made of their team name and their number on the team to find their data and update it when needed.

To output data I will need to be able to access these data structures and retrieve the data to be output to System.out.