FIT1048 Project Plan – Hackamon!

How to Play

Title Screen

When the game is loaded, the title screen will be the first thing that the user will see. The title screen will contain the welcome text seen above, as well as two options–*"New Game"* and *"Load Game".*

Loading a game will enable the user to load their previously saved progress. There will be three save slots, and each save file will follow the naming convention *"<`PLAYERNAME-SAVESLOTNUMBER`>.txt".*For example, if the name of the player was "Jasmine" and her progress was saved in the third save slot (because there were already two different games saved earlier) the saved file would be called *"`Jasmine-SaveSlot3.txt`".*

Introduction

"After years of training, you finally got the chance to take part in the most prestigious hacking competition in the world–Hackamon. Your whole life, you've been dreaming of becoming the Hackamon Champion, and now you have the chance to become one. But of course, you first need to move up the ranks before you can compete in the champion class."

The text above will be stored in *"`hackamonIntro.txt`"* and will be displayed when starting a new game. The user will then be prompted for a name, after which the following text below which outlines the rules of the games will be shown.

"Hi <NAME>, welcome to Hackamon!

Hackamon is an international hacking competition which attracts the best hackers in the world–of course, since you're here, you must be really good!

You must be pretty excited so let's get right into the rules:

As a hacker, you will have a hacker level and a hacker class. Since you're new you will start as a level 1 hacker, and you will be competing in the Baby class. There are 5 different classes:

- *Baby (Levels 1-2)*
- *Novice (Levels 3-6)*
- *Leader (Levels 7-11)*
- *Elite (Levels 12-14)*
- *Champion (Level 15+)*

To level up you will need to accumulate a certain number of points. This varies between levels, and of course, the higher level you are the more points you need to level up. How do you get points you say? That's simple, you just need to start hacking!

At each hacka-match, your objective is to crack a code consisting of numbers. You will have a certain number of turns allowed to crack a code, and the length of code depends on both your hacker level and your chosen difficulty. At each of your turn, you will be allowed to guess what you think the code is. After your turn, you will then be notified how many digits from your guess is 1) is the right digit and in the right position and how many 2) is the right digit but in the wrong position. The hackamatch will end when you run out of turns or once you correctly hack the code. Depending on how well you solved the code, you will be given points, which you need to level up. These rules will be shown to you at the start of your first hackamatch.

Also, I almost forgot! You can change the difficulty of every hackamatch. Since you're in the Baby Class, you can only play on easy mode. But once you get to novice, you can then play easy, medium, hard and custom. At level 15, you will unlock the extreme difficulty option. Only by beating this extreme difficulty will you then be granted the "Champion" class.

But be careful! Lose 5 hackamatches in a row and you will be automatically demoted 2 levels down. However, win 5 hackamatches in a row and you receive a huge points bonus!

That's all from me for now. If you need HELP at any time at all just write [HELP] on the prompt. To START hacking, write [START]. Or if you want to SAVE, you may write [SAVE]. If you want to exit Hackamon, you may write [EXIT].

Good luck!”

The text above will be stored in *“`hackamonBasicRules.txt`”* and will be shown to the user line by line so that it mimics a person welcoming them to the hackamon competition.

The Hackamon! hub prompt text will then be shown. The hub is where the user is taken when they don’t have an active hackamatch happening.

Welcome to the Hackamon! hub. What do you do now?

[START] - start a new hackamatch
[HELP] - view information
[SAVE] - save game
[EXIT] - exit Hackamon!

My choice: “

Detailed Rules

The following text will be stored in *“`hackamonFullRules.txt`”* and will be shown to the player if they want to read the detailed rules before the start of every hackamatch.

“You have <TURN_NUMBER> tries to hack the passcode, which can consist of any number combination using the digits <ALLOWED_DIGITS>. Not all digits may be used, and repetition is allowed. E.g. one example passcode for the digits 0-3 can be “1200”.

During each turn, you need to input what you think the passcode is WITHOUT SPACES then press enter. For example, “123456”. Once you press enter, you will then be given feedback on your guess.

“–” means that you have guessed a correct digit but you did not place it in the correct position.

“=” means that you have guessed a correct digit and placed it in the correct position.”

You may receive several “–” and “=” or none at all, depending on how well you have guessed the code.

Once the game ends, you receive points which corresponds to how well you played the game. You receive a bonus for taking less turns than the maximum allowed for this game, and you also get bonus points if the difficulty is higher.

If you get stuck at any point in the game, you may type [HINT] to reveal one part of the code. Every hint you ask for will reduce the number of points you earn, and the number of hints you are allowed is 1 less than the length of the passcode.

If you want to save your game, you may write [SAVE].

If you want to forfeit from the current match, you may write [FORFEIT]. This results in an automatic loss with 0 points earned in your tally.

If you want to exit Hackamon, you may write [EXIT].

Finally, if you ever need to view Hackamon! rules again during the game, just write [HELP].”

Help Screen

The following text will be shown when the user asks for help (writes [HELP] in the command line).

“Welcome to HACKAMON! Help Centre.

[1] View hackamatch rules.
[2] View hackamatch difficulty information.
[3] View classes and levelling up information.
[4] View my player stats.
[5] View commands list.
[6] Go back to hub/continue hackamatch.

I would like to: “

Each of the following options will display more information to the user. Option [6] will allow the player to continue their current hackamatch if the state is ACTIVE. The user goes back to the hub otherwise.

Game Development Outline

Game Setup

Upon starting the application file, the user is prompted to either start a new game or load a saved game. If they choose to load a game, they continue where they left off. If they start a new game, the *“`hackamon.txt`”* file is read and displayed to the user.

A player class is then initialised. The application asks for a users name, then stores it in the class. The rest of the attributes are set to a default value. A Hackamon class is also created, whose attributes are also initialised to their default values. The state of the Hackamon class will then be in its “IDLE” state at the moment, because there is no active games happening. This is important so that when saving a file, the save file can know whether to come back to an active hackamatch or not.
“`hackamonBasicRules.txt`” is then read and displayed to the user in the method described above.

The user will then be prompted on what they would like to do. Should the player decide to START a hackamatch, the user is prompted for a difficulty (available difficulty may vary depending on the player’s level) and the player’s response will then be used to instantiate a Hackamatch class. All available difficulties are easy, medium, hard, extreme and custom. Selecting custom will prompt the user for the number of guesses allowed in the hackamatch, the length of the code to be guessed, the number of digits which the code could be made up to (up to 10, i.e. the digits used are from 0-9). Every time a hackamatch is started, a new Hackamatch object will be created so there is no need to clear the variables. We will need to destroy the object later however. A secret code is then generated.

Before every hackamatch starts, the player is prompted whether they would like to view the rules or not. If they say yes, *“hackamonFullRules.txt”* will be displayed. The user can then choose to view this again later at the next hackamatch or through the help centre.

Player’s Turn and Feedback Sample

At the start of each player’s turn, they will see the mastermind board. However, since my theme is a hacking competition, a computer will be displayed instead. The computer will change width/height

depending on hackermatch object attributes. A sample display can be seen on the next page. The turn number will be displayed on the left, the player's past guesses can be viewed as seen below, and the feedback will be displayed on the right in the form of "–" and "=".

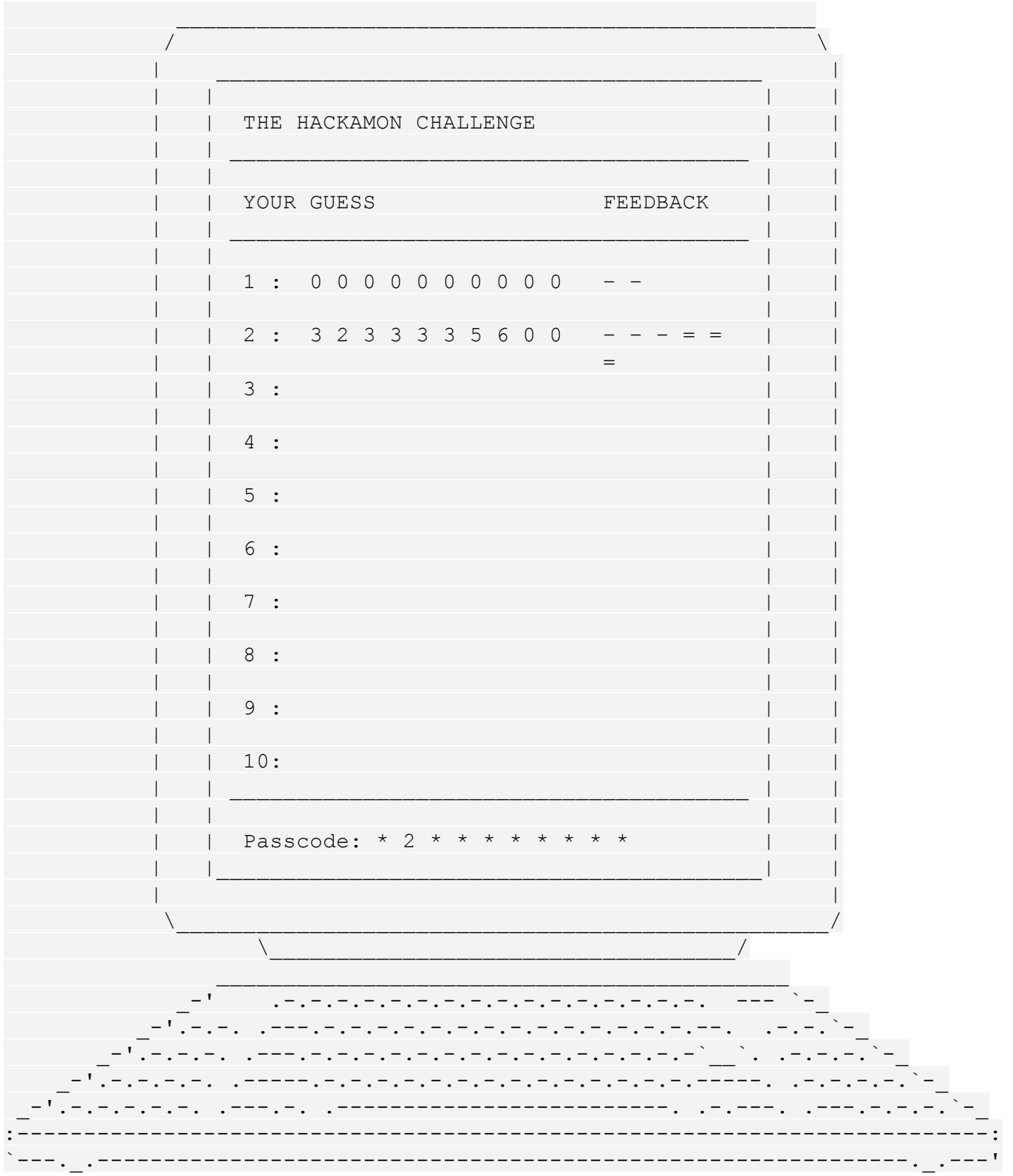
The user will then be prompted on whether they would like to input a command or a guess. They will then be prompted for either a guess or a command, depending on their response. "`PLAYERNAME`" will also be replaced with the player's actual name.

Once the user presses enter, it will then be processed and checked whether they input a valid command or guess.

If it is invalid, the user will be shown an error message and be prompted again as above. If it is valid, hackamatch class will then process it as detailed in the "processing player input section" and then provide feedback as such displayed in the ascii art on the next page. Details from the feedback can be seen through the rules section. A '–' means that the player guessed a correct colour but placed it in the wrong spot. A '=' means that the player guessed both a correct colour and placed it in the correct spot.

This process then repeats for as long as the player has guesses remaining

Go to next page for a sample feedback



```
Do you want to input a guess (passcode) or a command (e.g. HELP)?
Input 0 for guess, 1 for command:
```

IF OPTION 0 DISPLAY: `PLAYERNAME:\> Your guess:`
IF OPTION 1 DISPLAY: `PLAYERNAME:\> Command:`

SAMPLE ERROR MESSAGE: `Error: Invalid input. Try again.`

Processing Player Input

In *Hackamon!*, there are several scenarios in which a player may enter an input:

- ACTIVE STATE: During an active hackamatch
- IDLE/HELP STATE: Outside the hackamatch (before the game has started, after the game has finished, when help screen is brought up.

The user will be prompted for a response in all of these Hakamon! states. Any time the user is prompted for a response, regardless of the state, the player may enter a number of commands. The commands available for use for each state is as follows:

IDLE	ACTIVE	HELP
start	hint	1
save	help	2
help	save	3
exit	forfeit	4
	exit	5
		6

START – Starts a new hackamatch
SAVE – Saves the player's progress/hackamatch
HELP – Goes to help centre menu which has options to view more detailed information
EXIT – Exits Hackamon!
HINT – Reveals 1 digit of the code to the player. Limited to `passcode.length() - 1` uses
FORFEIT – Quits the hackamatch, counts as a loss to the players tally

For the help options, refer to the "Help Screen" section above.

There will also be helper functions in my application file which will make asking for input easier.

- `int getIntInput(string question)`
- `int getStringInput(string question)`

Based on the player's response, the program will perform the following:

- START – creates a new hackamatch class and initialise all of its necessary variables (see above)
- SAVE – creates new files which stores a stringified version of existing objects. The files will be saved following the naming convention detailed above (<`PLAYERNAME-SAVESLOTNUMER.txt`>)
    - When loading a saved file, the function `void loadSavedFile(int savedFileName)` will initialise all required objects, using the data stored in the saved file.
- HELP – takes the user to the help screen (can be seen in the Help Screen section above), which then gives them a prompt on what they would like more info on
- EXIT – exits the application. If the user does not save, their progress will be lost

- HINT – The hackamatch object will select a random digit of the passcode to be revealed, and will replace one of the asterisks in the display with the actual digit.
- FORFEIT – reveals the passcode to the user, and then deletes the hackamatch object created. A loss will also be added to the player object's "losses" attribute.

End Game Conditions

Every hackamatch can end in one of two ways:

- The player guesses the code correctly within the given number of guesses allowed
- The player does not guess the passcode, and they have run out of guesses

After the player's guess is checked against the passcode, checkGameOver() is called before giving feedback to see if the player has any more turns remaining.

Additional Features

Theme/Board

The theme of my Mastermind game is a hacking competition. The player is a newbie hacker whose dream is to rise up the ranks of Hackamon! (an international hacking competition) and become the Hackamon! champion. During each hackamatch (one round of mastermind), instead of guessing the colour pattern like in mastermind, the player is tasked with hacking the secret passcode to a computer. They have a limited number of guesses, just like how there are a limited number of attempts when you enter the password to your account in real life.

The mastermind board has also been replaced with a computer drawn in ASCII which can be seen above to give a more immersive experience to the player. The board will be displayed with the `displayBoard()` function of the hackamatch class.

Difficulty Selection

During each match, the player can select either easy, medium, hard, extreme or a custom difficulty. The player's class also affects what difficulties the player is able to select.

- Baby class – easy
- Novice class – easy, medium, hard
- Leader/Elite class – easy, medium, hard, custom
- Elite from level 15+ and champion class – easy, medium, hard, extreme, custom

Custom enables the player to set the length of the passcode, the number of digits the passcode is to be selected from, and the number of guesses the player is allowed.

Classes/Levelling Up/Points Scoring

Upon first starting the player is at level 1, and in the baby class. As the user gains more points by playing hackamatches and cracking harder passcodes, they can level up. Your class can rank up as shown below:

- Baby (Levels 1-2)
- Novice (Levels 3-6)
- Leader (Levels 7-11)

- Elite (Levels 12-14)
- Champion (Level 15+)*

The user actually does not automatically become champion upon reaching level 15. At level 15, they unlock the extreme difficulty and only once they are able to solve it will they be granted champion status. If they fail on extreme difficulty they are immediately brought back to the start of level 14, and so they must again work their way up to level 15.

In general, if a user wins 5 games in a row, they are awarded bonus points instead of being promoted to a higher level (like in the brief). However, the bonus points will be more than enough to level up the player which is similar. Furthermore, if they lose 5 games in a row however, they are automatically demoted 2 levels down.

During each hackamatch, the player starts with a certain number of points. They lose points for every hint they use. They can use 1 hint per turn. The total hints in a hackamatch equals the `length of the code - 1`. There is also bonus points awarded at the end of every match for every guess that is leftover/unused. There are also bonus points for winning streaks. The difficulty setting the player chose will also be decide the multiplier value. The final score of the player is their `points obtained in game * multiplier`. The multiplier also decreases with each hint the player uses.

Save/Load Game

Whenever a game is saved, files will either be created or updated and these files will be populated with the data stored in existing objects in the whole game. This will be done by the `saveGame()` () function from the Hackamon class.

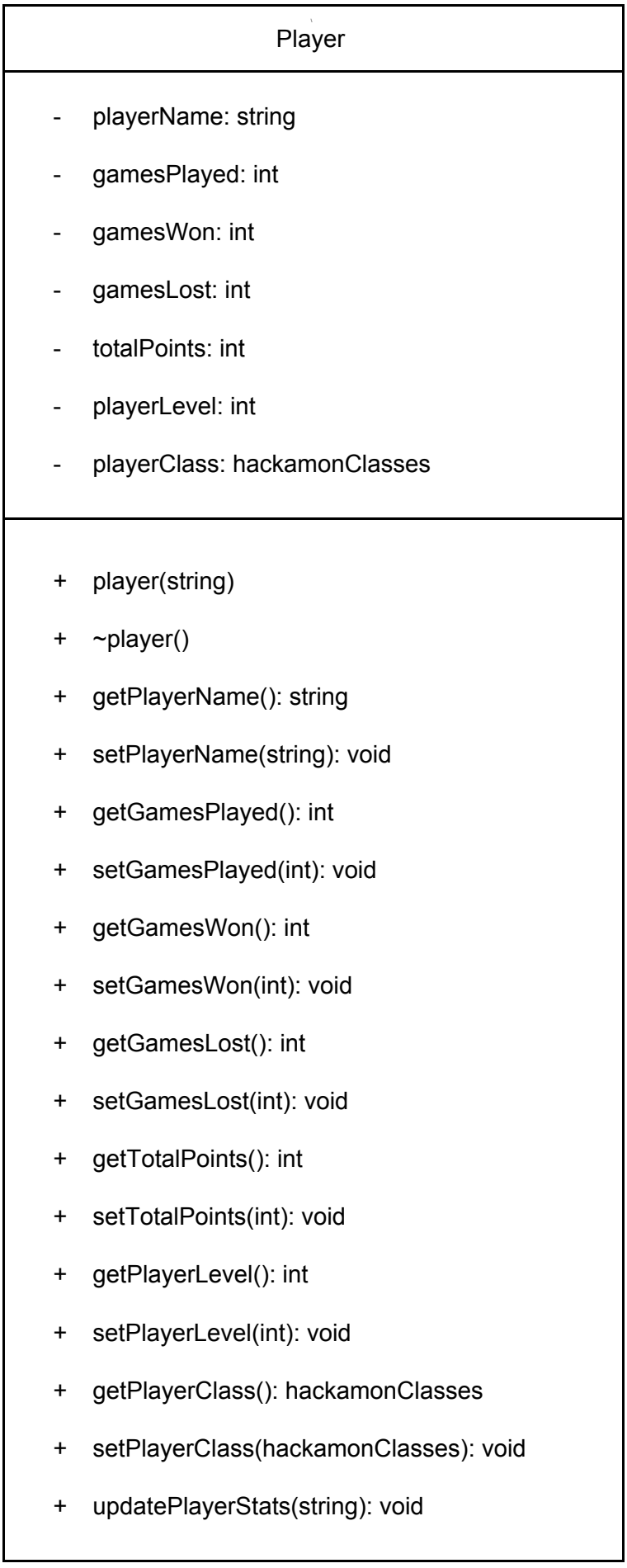
Like mentioned above: Loading a game will enable the user to load their previously saved progress. There will be three save slots, and each save file will follow the naming convention *"<`PLAYERNAME-SAVESLOTNUMBER`>.txt".* For example, if the name of the player was "Jasmine" and her progress was saved in the third save slot (because there were already two different games saved earlier) the saved file would be called *"`Jasmine-SaveSlot3.txt`".* This will be done by the loadGame() function.

Extra Commands

The list of commands are detailed above.

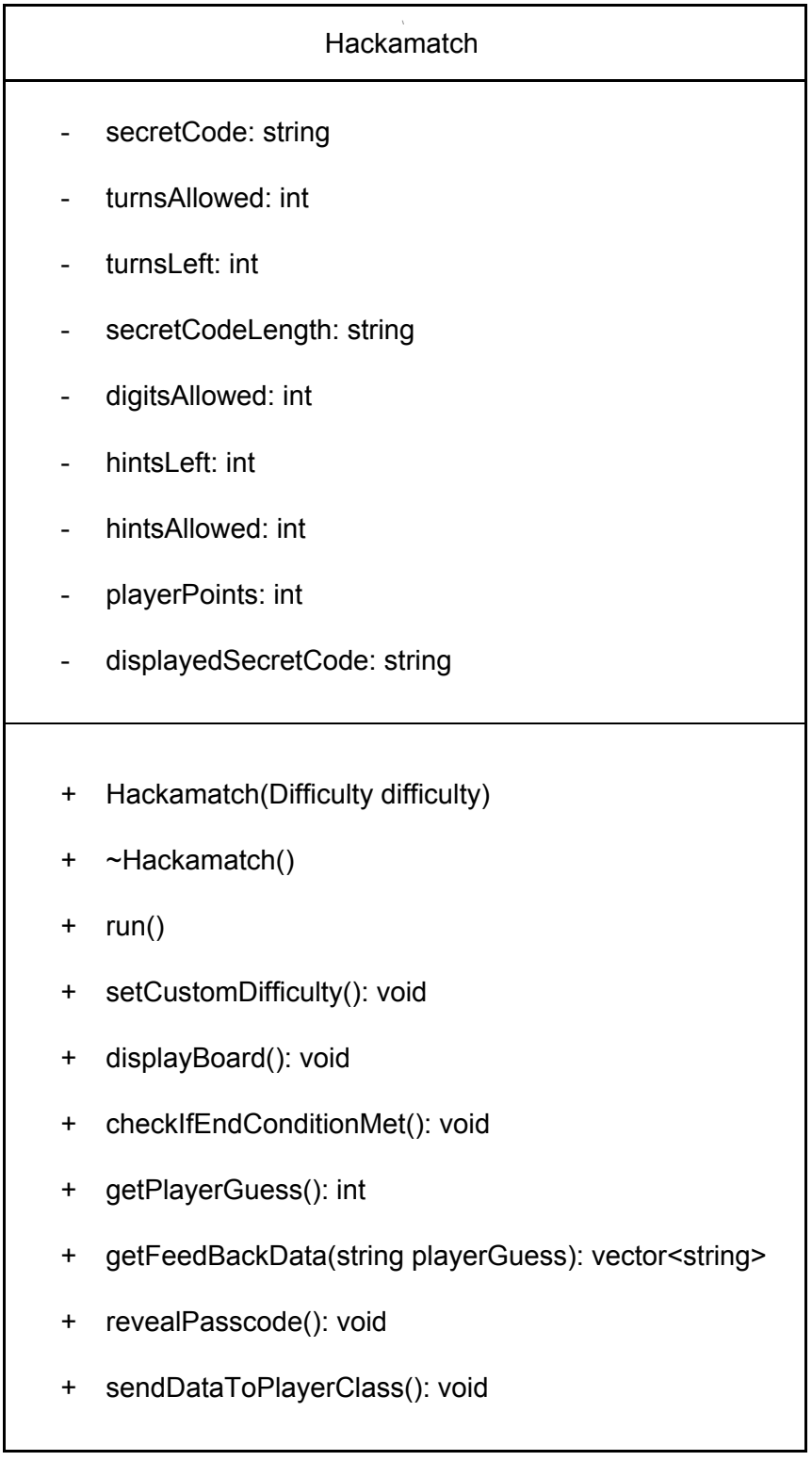
UML Diagrams

<<Enumeration>> hackamonClasses	<<Enumeration>> hackamonStates	<<Enumeration>> difficulty
BABY	IDLE	EASY
NOVICE	ACTIVE	MEDIUM
LEADER	HELP	HARD
ELITE		EXTREME
CHAMPION		CUSTOM


Player
- playerName: string
- gamesPlayed: int
- gamesWon: int
- gamesLost: int
- totalPoints: int
- playerLevel: int
- playerClass: hackamonClasses
+ player(string)
+ ~player()
+ getPlayerName(): string
+ setPlayerName(string): void
+ getGamesPlayed(): int
+ setGamesPlayed(int): void
+ getGamesWon(): int
+ setGamesWon(int): void
+ getGamesLost(): int
+ setGamesLost(int): void
+ getTotalPoints(): int
+ setTotalPoints(int): void
+ getPlayerLevel(): int
+ setPlayerLevel(int): void
+ getPlayerClass(): hackamonClasses
+ setPlayerClass(hackamonClasses): void
+ updatePlayerStats(string): void

Main
None
- displayTextFromFile(string filename): void - getStringInput(string prompt, string errorMessage, vector<string> choices): string - getIntInput(string prompt, int lo, int hi): int - processPlayerInput(string input): void - main(): int

Hackamon
- state: hackamonStates
+ Hackamon() + ~Hackamon() + run() + loadGame(string filename) + saveGame(string filename)


Hackamatch
- secretCode: string
- turnsAllowed: int
- turnsLeft: int
- secretCodeLength: string
- digitsAllowed: int
- hintsLeft: int
- hintsAllowed: int
- playerPoints: int
- displayedSecretCode: string
+ Hackamatch(Difficulty difficulty)
+ ~Hackamatch()
+ run()
+ setCustomDifficulty(): void
+ displayBoard(): void
+ checkIfEndConditionMet(): void
+ getPlayerGuess(): int
+ getFeedBackData(string playerGuess): vector<string>
+ revealPasscode(): void
+ sendDataToPlayerClass(): void