Beabadoobee Trivia Maze

David Hoang, Faith Capito

BE BOUNDLESS

About me: David Hoang

- Class standing: Junior, age : 21
- Ethnicity: Vietnamese
- From: Tacoma
- Hobbies: Raving, Going to the gym, photography.
- Some Artists I like: Beabadoobee, Wave To Earth, Joji, Daniel Caesar, Cigarettes After Sex, Frank Ocean, and Mac DeMarco.

Something I’m Proud Of:

About Me: Faith Capito

- 23 years old, Junior for CS
- South Puget Sound Community College
  - University of Washington Bothell
    - University of Washington Tacoma
- Hobbies: Multimedia Artist, Dog Training, Reading
- Artists I enjoy: Lamp, Kali Uchis, Gorillaz, $uicideboy$

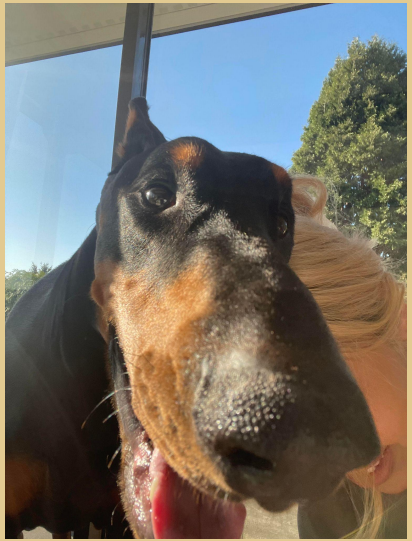
UNIVERSITY of WASHINGTON

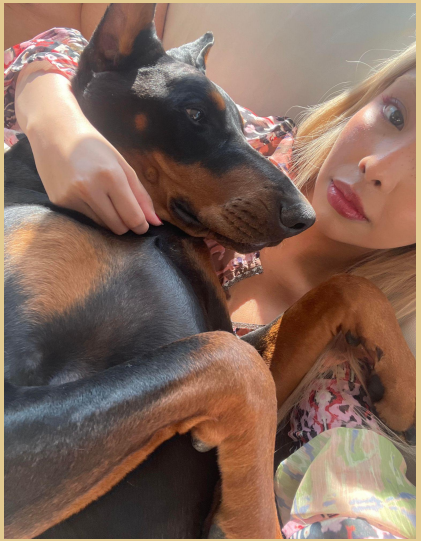
About Me

Happy mom of a Doberman Pinscher named Bruce!

Silly little guy that I have been training for 5 years (he’s a scorpio). Worst personal protection dog ever.

We enjoy walking trails and beaches!





OUR PROJECT

Beabadoobee Trivia Maze is a game revolving around your knowledge of the Indie Rock Artist, Beadbadoobee. It includes a simple, 2D graphical user interface and a fun little character that will travel with you as you traverse through the maze.

To pass through the doors, you will be met with different types of questions: Multiple choice, short answer, and true or false.

GAMEPLAY

The goal is to find your way to the end of the maze through a series of locked doors that can only be passed when a question is answered correctly. If incorrect, the door permanently locks.

Answered doors are traversable for alternate routes.

If all routes are locked off from the exit, you lose!

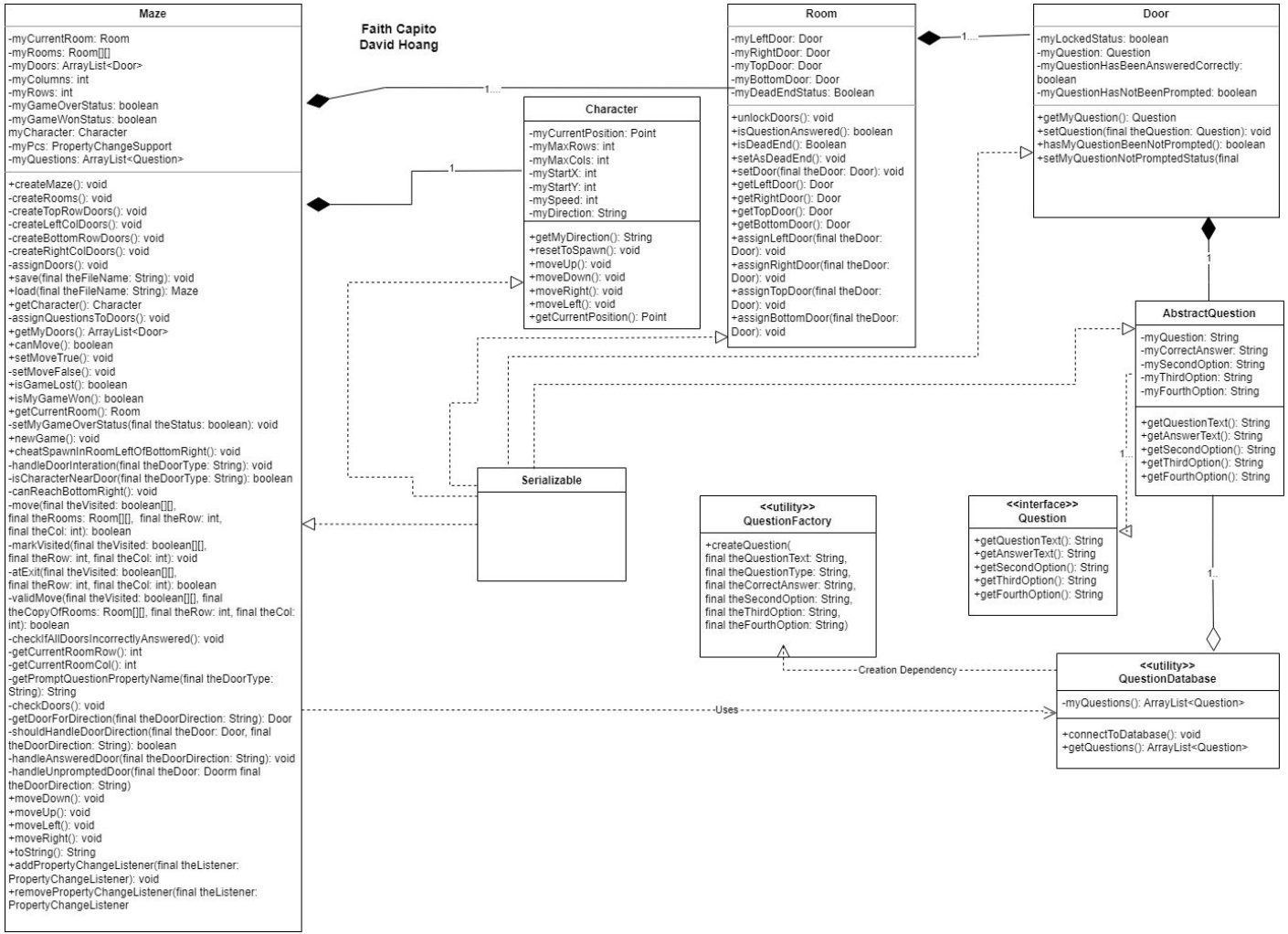
FEATURES

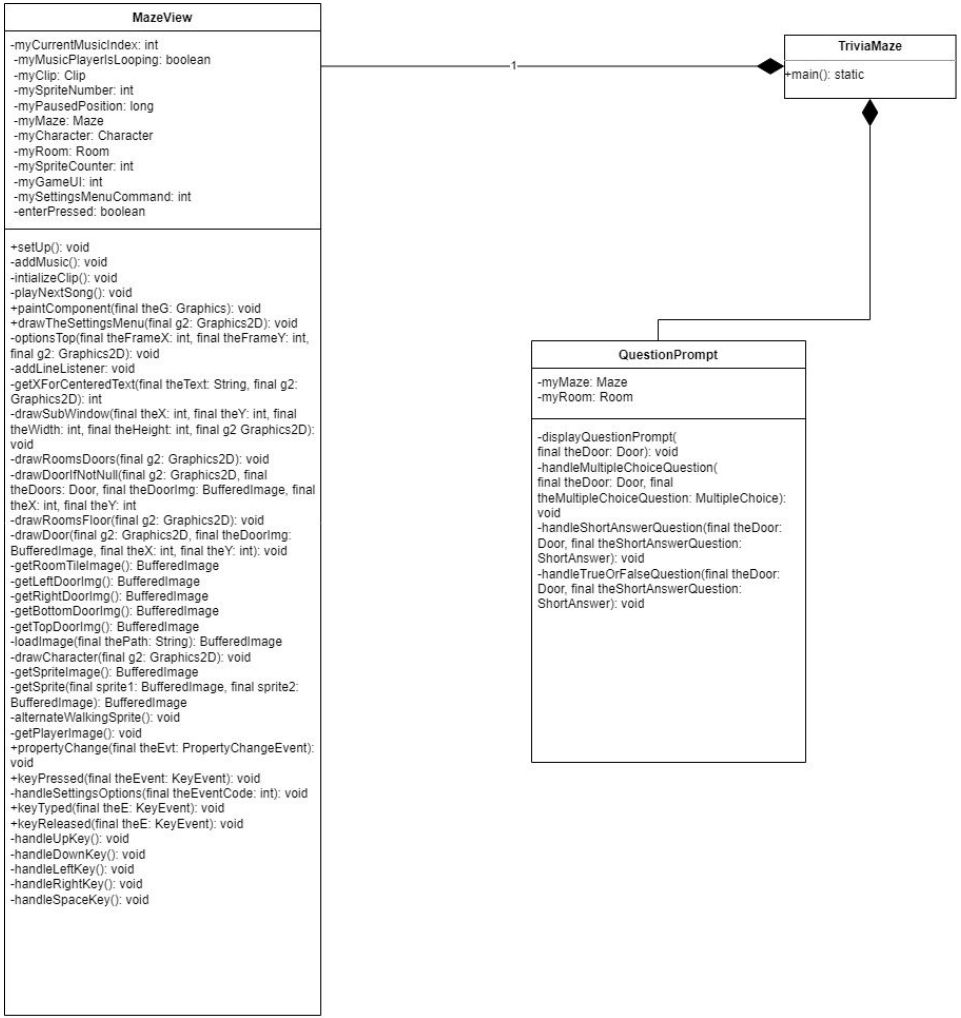
- Wander through the maze using arrow keys or WASD.
- Need a break? Press the spacebar to pause the game!
- If you want to come back to it later, save at any point in the game and load it back up once you're ready to launch.
- Enjoy your gameplay with background music or feel free to change, or pause, the music player to your needs.

DESIGN FEATURES

- Model-View-Controller pattern with GUI using Java Swing.
- Factory Design Pattern
- Abstraction
- Polymorphism
- Inheritance
  - AbstractQuestion
  - Question (Interface)
  - QuestionFactory
  - MultipleChoice
  - ShortAnswer
  - TrueFalse
- Serialization

UML DIAGRAM


Faith Capito
David Hoang
Maze
-myCurrentRoom: Room
-myRooms: Room[][]
-myDoors: ArrayList<Door>
-myColumns: int
-myRows: int
-myGameOverStatus: boolean
-myGameWonStatus: boolean
myCharacter: Character
-myPcs: PropertyChangeSupport
-myQuestions: ArrayList<Question>
+createMaze(): void
-createRooms(): void
-createTopRowDoors(): void
-createLeftColDoors(): void
-createBottomRowDoors(): void
-createRightColDoors(): void
-assignDoors(): void
+save(final theFileName: String): void
+load(final theFileName: String): Maze
+getCharacter(): Character
-assignQuestionsToDoors(): void
+getMyDoors(): ArrayList<Door>
+canMove(): boolean
+setMoveTrue(): void
-setMoveFalse(): void
+isGameLost(): boolean
+isMyGameWon(): boolean
+getCurrentRoom(): Room
-setMyGameOverStatus(final theStatus: boolean): void
+newGame(): void
+cheatSpawnInRoomLeftOfBottomRight(): void
-handleDoorInteration(final theDoorType: String): void
-isCharacterNearDoor(final theDoorType: String): boolean
-canReachBottomRight(): void
-move(final theVisited: boolean[][],
final theRooms: Room[][], final theRow: int,
final theCol: int): boolean
-markVisited(final theVisited: boolean[][],
final theRow: int, final theCol: int): void
-atExit(final theVisited: boolean[][],
final theRow: int, final theCol: int): boolean
-validMove(final theVisited: boolean[][], final
theCopyOfRooms: Room[][], final theRow: int, final theCol:
int): boolean
-checkIfAllDoorsIncorrectlyAnswered(): void
-getCurrentRoomRow(): int
-getCurrentRoomCol(): int
-getPromptQuestionPropertyName(final theDoorType:
String): String
-checkDoors(): void
-getDoorForDirection(final theDoorDirection: String): Door
-shouldHandleDoorDirection(final theDoor: Door, final
theDoorDirection: String): boolean
-handleAnsweredDoor(final theDoorDirection: String): void
-handleUnpromptedDoor(final theDoor: Doorm final
theDoorDirection: String)
+moveDown(): void
+moveUp(): void
+moveLeft(): void
+moveRight(): void
+toString(): String
+addPropertyChangeListener(final theListener:
PropertyChangeListener): void
+removePropertyChangeListener(final theListener:
PropertyChangeListener
1
1
Character
-myCurrentPosition: Point
-myMaxRows: int
-myMaxCols: int
-myStartX: int
-myStartY: int
-mySpeed: int
-myDirection: String
+getMyDirection(): String
+resetToSpawn(): void
+moveUp(): void
+moveDown(): void
+moveRight(): void
+moveLeft(): void
+getCurrentPosition(): Point
Room
-myLeftDoor: Door
-myRightDoor: Door
-myTopDoor: Door
-myBottomDoor: Door
-myDeadEndStatus: Boolean
+unlockDoors(): void
+isQuestionAnswered(): boolean
+isDeadEnd(): Boolean
+setAsDeadEnd(): void
+setDoor(final theDoor: Door): void
+getLeftDoor(): Door
+getRightDoor(): Door
+getTopDoor(): Door
+getBottomDoor(): Door
+assignLeftDoor(final theDoor:
Door): void
+assignRightDoor(final theDoor:
Door): void
+assignTopDoor(final theDoor:
Door): void
+assignBottomDoor(final theDoor:
Door): void
1...
Door
-myLockedStatus: boolean
-myQuestion: Question
-myQuestionHasBeenAnsweredCorrectly:
boolean
-myQuestionHasNotBeenPrompted: boolean
+getMyQuestion(): Question
+setQuestion(final theQuestion: Question): void
+hasMyQuestionBeenNotPrompted(): boolean
+setMyQuestionNotPromptedStatus(final
1
AbstractQuestion
-myQuestion: String
-myCorrectAnswer: String
-mySecondOption: String
-myThirdOption: String
-myFourthOption: String
+getQuestionText(): String
+getAnswerText(): String
+getSecondOption(): String
+getThirdOption(): String
+getFourthOption(): String
1..
Serializable
<<utility>>
QuestionFactory
+createQuestion(
final theQuestionText: String,
final theQuestionType: String,
final theCorrectAnswer: String,
final theSecondOption: String,
final theThirdOption: String,
final theFourthOption: String)
<<interface>>
Question
+getQuestionText(): String
+getAnswerText(): String
+getSecondOption(): String
+getThirdOption(): String
+getFourthOption(): String
1..
Creation Dependency
<<utility>>
QuestionDatabase
-myQuestions(): ArrayList<Question>
+connectToDatabase(): void
+getQuestions(): ArrayList<Question>
Uses



UML DIAGRAM

UNIVERSITY *of* WASHINGTON

FUTURE WORK

- Would like to add special items/accessories that the player can find and alter their character with (character customization).
- Add a timer that limits how quickly the player can answer a question before the door locks.
    - Could be part of selecting different difficulty levels.

ON-BRAND STATEMENT

FOR GENERAL USE

> **What defines the students and faculty of the University of Washington? Above all, it's our belief in possibility and our unshakable optimism. It's a connection to others, both near and far. It's a hunger that pushes us to tackle challenges and pursue progress. It's the conviction that together we can create a world of good. And it's our determination to Be Boundless. Join the journey at uw.edu.**

UNIVERSITY *of* WASHINGTON