PART 1: PROGRAM SPECIFICATION

- **What will your program look like at the end?**

  The program is a board frame with blocks that are colored in two different colors; light brown and dark brown. There's gonna be a character moving over the blocks and controlled by the player of the game.

  At the top most right corner, there's gonna be an indication of the scores that the player currently has. In addition, there's also gonna be an indication of the number of lives that the player currently has, which are gonna be changing depending on the ability of the player to answer questions correctly or not.

  At the top most left corner of the frame, there is the final goal where the player receives the treasure when the character reaches that final goal.

  Also the design of the program is impressive and good to look at. The main board is surrounded by trees with the goal represented as an arc in blocks.

- **How will the user input work? (e.g., clicking, keys on the keyboard, specifying a file, what have you)**

  The user will use arrow keys to control the direction of the runner. The user should also be able to click letter keys on the keyboard to choose a category from which they'll answer a question. The player uses a mouse to select options from the menu page.

- **How will the program respond?**

  The user will have to move the character across the blocks and once the character stands on a light-brown block, the player will be prompted to choose a category of questions.

For any category chosen, a question will be displayed and the player will have to answer. For every correct answer, the block changes color to green and for any wrong answer, the block changes to red. If the question is left unanswered, the block will remain unchanged until you complete the question at that block.

- **What purpose does it serve? (e.g., is it a game, a productivity tool, or a screen saver?)**

This program is a quiz game and it serves the purpose of educating a player in a fun and enjoyable way about facts in different fields. The main goal of this game is to facilitate learning in a creative way and testing your knowledge on different topics.

PART 2: PROGRAM DESCRIPTION

Description

The player's aim is to answer the questions correctly to gain the maximum score and get the treasure. The categories of questions to be answered are stored in an N-ary tree data structure. Every category node extends from the root node and has 12 questions as its children, and every question node has an array list of nodes which store the answer choices. Initially the correct answer is stored at index 0 before shuffling the choices and getting a new random order. The new order is the one displayed to the player. When the player chooses an answer, it is compared to the correct answer that has been stored in order to determine whether the player is correct or not.

When the player chooses a category, they get a random question from the pool of questions that are randomly generated from a source on the internet.

- **Main**
    - **Holds the frame which contains the game**
    - **Methods**
        - **main(): Public, static, void**
            - **Initializes the game, draws the frame and adds the GamePanel.**
- **GamePanel**
    - **Variables**
        - **Screenwidth, Screenheight, FPS: Public, static, final**
        - **Gamestate: Public, static, Integer**
        - **ChallengeCompleted: Static, Boolean**
        - **gameThread: Thread**
        - **ActionListener: Instance of ActionListener class**
        - **MouseListener: Instance of MouseListenerClass class**
        - **character: Public, static, Instance of Player class**
        - **menu: Instance of Menu class**
    - **Methods**
        - **GamePanel: Constructor, Public,**
            - **Initializes Player and Menu classes**
            - **Calls startgamethread method**
        - **startgamethread(): Public, void**
            - **Initializes Thread to gameThread**
            - **Starts gameThread**
        - **Run: Overrides the run method from the interface Runnable**

        - **Update: Public, void**
            - **Coordinates and updates display of player movements**
        - **paintComponent: Public, void, Parameter(Graphics g)**
        - **draw()**
    - **Classes**
        - **ActionListener**
            - **Implements the KeyListener class**
            - **Overrides keyPressed, keyReleased, keyTyped: All Public, void**
            - **keyPressed**
                - **Displays Menu on press “M”**
                - **Assigns gamestate to 0**
            - **keyReleased**
                - **Assigns the KeyListener Object to the character's keyReleased method.**
        - **MouseListenerClass**
            - **Implements the MouseListener Interface**
            - **mouseClicked, mouseReleased, mouseEntered, mouseExited, mousePressed: Overridden methods from MouseClicked. Public, void.**
            - **mousePressed: changes the display based on clicked options.**
- **Player**
    - **Variables**
        - **menu: Instance of Menu class**

        - **position: Static, Instance of Pair class**
            - **Determines the x and y coordinates of the player.**
        - **speed: public, Integer**
            - **For the player speed.**
        - **uppressed, downpressed, leftpressed, rightpressed, mousepressed: public, boolean**
        - **downA, downB, upA, upB, rightA, rightB, leftA, leftB: public, BufferedImage**
        - **Direction: public, String**
        - **spritetracker: public, Integer**
            - **Initialized to 0.**
        - **id: public, Integer. Initialized to 1**
            - **For the direction of the player**
        - **tileset: Instance of Tiles class**
        - **quizes: Instance of QuestionManager class**
        - **gp: Instance of GamePanel**
    - **Methods**
        - **Player: public, Constructor**
            - **Initiates the menu, tiles, and question classes. Initializes the player position, speed. The method calls getPlayerImage method to draw the image.**
        - **getPlayerImage: public, void**

            - **Initiates the downA, downB, upA, upB, rightA, rightB, leftA, leftB to appropriate png images to accurately represent the sprite at different positions/orientations.**
            - **Enclosed in a try/catch block**
        - **keyPressed: public, void**
            - **Handles player movements and sprite presentation.**
        - **keyReleased: public void**
            - **Handle player movements and sprite presentation.**
        - **update: public, void**
            - **Changes player speed and position upon key press**
        - **getX / getY: public, Returns: double**
            - **Returns the player position**
        - **draw: public, void**
            - **Parameter: Graphics object**
            - **Draws the image**
            - **Calls the tileset and questions class draw methods**
- **Pair**
    - **Variables**
        - **x: public, double**
        - **y: public, double**
    - **Methods**
        - **Pair: Public, Constructor**
            - **Parameters: double X (position x), double Y (position y)**

        - **Initializes the x and y variables**

- **Menu**
  - **Methods**
    - **drawMenu: public, void**
      - **Parameter: Graphics object**
      - **Draws the menu panel**
    - **drawHelpScreen: public, void**
      - **Parameter: Graphics object**
      - **Draws the help panel**
- **Tiles**
  - **Variables**
    - **image: public, BufferedImage**
  - **Methods**
    - **Tiles: Public, Constructor**
      - **Creates a new tiles array.**
    - **drawTile: public, void**
      - **Parameter: Graphics object**
      - **Draws the wider background tile image on the panel**
    - **draw: public, void**
      - **Parameter: Graphics object**
      - **Draws individual tiles**
    - **getTileImage: public, void**
      - **References the images to draw the tiles from the sprites folder.**

- **QuestionManager**
    - **Variables**
        - **Count: Public, Integer**
    - **Methods**
        - **Life(): Public, Constructor**
        - **looseLife(): Public**
            - **Calls checkAnswer(). If F, then remove 1 life, else call showHint()**
        - **updateLife(): Public**
            - **Updates and calls the draw method to update remaining lives.**

- **Questions**
    - **Variables**
        - **trial2: Trivia instance**
        - **gp: GamePanel instance**
        - **gamer: Player instance**
        - **lives: static, Integer**
            - **Initialized to 5 and keeps track of player lives left**
        - **score: static, Integer**
            - **Initialized to 0 and keeps track of score**
        - **treasureChest, heart: BufferedImage**
        - **answer: char**
    - **Methods**

- Question: Public, Constructor
    - Parameters: GamePanel gp and Player gamer
- getImages: public, void
    - Gets images
- displayQuestion:
    - Display Question, handle gameover, and player lives run out message
- resetGame: public, void
    - Restarts the panel with a new game
- assignQuestions: public, void
    - Assigns questions to each tile
- keyReleased and keyPressed: public, void, Overridden methods
    - Handle player input.

- **Trivia**
    - **Classes**
        - **Node: Generic,**
            - **To handle nodes for the n-ary tree.**
            - **addChild and removeChild to handle adding and removing sub-nodes.**
            - **data: Generic variable**
            - **children: ArrayList of Generic Nodes**
        - **Question**
            - **Variables**

- **root: Node, String, data = “Root”**
  - **Every category is a children of root**
- **API_URL: final, static, String**
  - **Set to `https://opentdb.com/api.php?`**
- **categoryNode: static, Node, String**
  - **Stores the category before storing it in the n-ary tree**
- **incorrectAns: JSONArray, static**
  - **Stores incorrect choices return from API**
- **dataList: JSONObject, static**
  - **Stores the returned JSON from the API call**
- **numberOfQuestions: static, Integer**
  - **Sent to the API call for the number of questions to return,**
- **rando: static, Integer**
  - **Used in choosing a random question to return from the 12 questions in each category.**

- **Methods**
  - **Question: public, Constructor**
    - **Initializes the categories list and creates list with IDs for appropriate API calls.**
  - **addToTree: public, static, void**

            - **Parameters: String categoryQsn, ArrayList of question answers**
            - **Adds each choice in question answers to appropriate question**
        - **getQuestions: public, static, void**
            - **Parameter: String category**
            - **Makes Http call to API_URL to get questions for given category**
        - **getCategories: public, void, return: ArrayList of strings of categories.**
        - **getQuestionByCategory: public, Parameter: category index, Integer. Return: Node, String of question**
        - **getChoices: Parameter: public, Node questionNode, Return: ArrayList of strings**
        - **checkAnswer: Parameter: public, String, boolean return**
            - **Takes choice and returns boolean after comparison.**
- **Trivia**
    - **Methods**
        - **Trivia: public, Constructor**
            - **Creates new question instance**

        - **Initializes categoriesList to returned list from getCategories method.**
    - **display: public, void**
        - **Handles all displays to the panel**
    - **displayCategories, displayQuestion, displayChoices, displayResult: public, void**
        - **All methods call display with arguments of what's to be displayed.**
        - **All method take the Graphics object as parameter**
    - **displayTriviaPanel: public, void**
        - **Handles calls to the methods above from the Player class.**
        - **Takes Graphic object as parameter**
    - **keyPressed: public, void**
        - **Handles key presses from users and displays questions and categories as appropriate.**