# AniMath – Educational Math Game

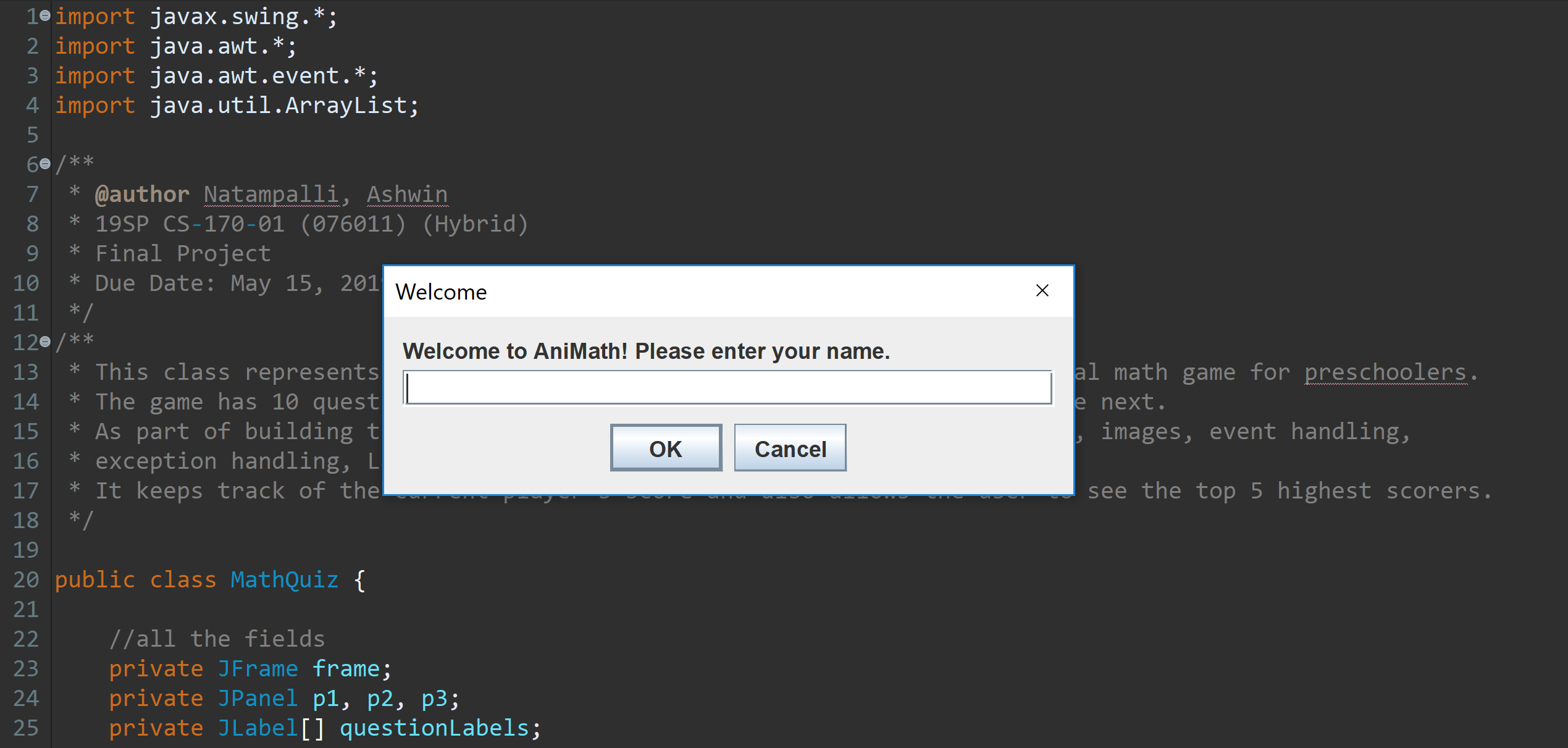
Project by: Ashwin Natampalli

Course: 19SP CS-170-01 (076011) (Hybrid)

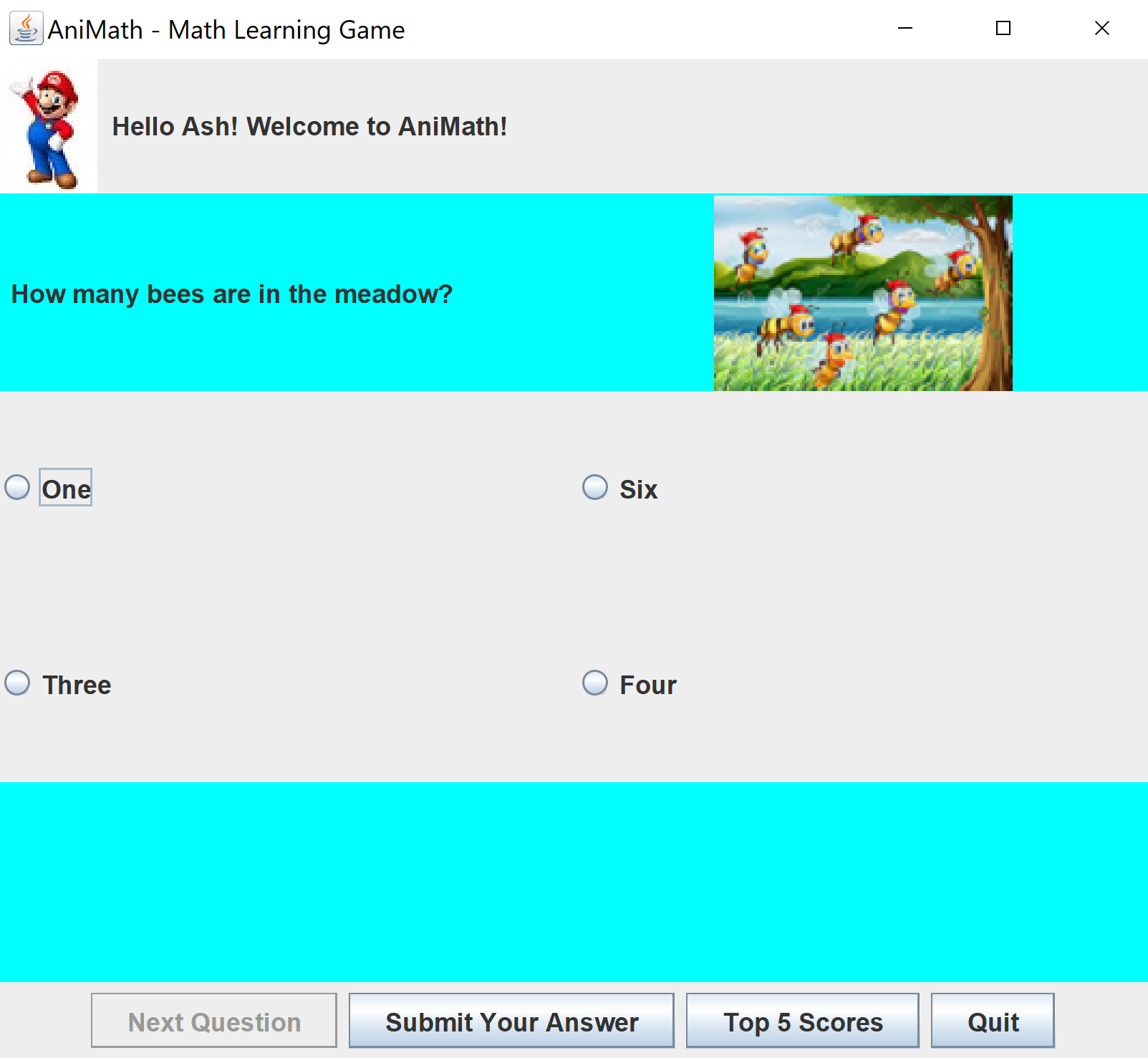
**Description:** The purpose of this project was to create an educational math game for preschoolers using Java. The game, AniMath, is a 10 question quiz that asks the player a number-based question given a picture of an animal/group of animals. Background music is continuously played throughout the game.

**Classes:** (please see the documentation within the classes for a detailed description of each of the methods)

* **QuizQuestion** - Represents the quiz questions, the associated answer choices, and the correct answer.
  + **Fields:**  private final String question;
  + private final int correctAnswerIndex;
  + private final String[] answers;
  + **Methods:** String getQuestion()
  + String getCorrectAnswer()
  + String[] getAnswers()
* **QuizCollection** - Represents the collection of QuizQuestions that are displayed to the user. The quiz questions, the associated answer choices, and the correct answer are encapsulated in the QuizQuestion class.
  + **Fields:**  private ArrayList<QuizQuestion> questionList = new ArrayList<QuizQuestion>();
  + **Methods:** ArrayList<QuizQuestion> getQuestions()
  + void addQuizQuestions()
* **PlayerScore** - Represents the player and their score.
  + **Fields:**  private int score;
  + private String name;
  + **Methods:** int getScore()
  + String getName()
* **PlayerScoreComparator** - Implements the Comparator interface, and provides specialized sorting of PlayerScore objects based on the scores.
  + **Methods:** int compare(PlayerScore score1, PlayerScore score2)
* **ScoreManager** - Gets the existing scores, updates the latest score, sorts the top 5 scores, and stores them in a file.
  + **Fields:**  private ArrayList<PlayerScore> scores;
  + private static final String SCORE\_FILE = "src/playerscores.dat"; (file where scores will be saved)
  + **Methods:** void importPlayerScores()
  + void exportPlayerScores()
  + void addScore(String name, int score)
  + ArrayList<PlayerScore> getScores()
  + void sort()
  + String displayTopFivePlayerScores()
* **AudioPlayer** - Generates the music that plays while the game is running.
  + **Fields:**  Long currentFrame;
  + Clip clip;
  + AudioInputStream audioInputStream;
  + static String filePath;
  + **Methods:** void play()
* **MathQuiz** - Represents the game AniMath, which has 10 questions and allows the user to traverse from one to the next. As part of building the user interface, it uses graphics, colors, sounds, images, event handling, exception handling, Layout managers, file I/O, and other techniques. It keeps track of the current player's score and also allows the user to see the top 5 highest scorers.
  + **Fields:**  private JFrame frame;
  + private JPanel p1, p2, p3;
  + private JLabel[] questionLabels;
  + private JRadioButton[] answerRadioButtons;
  + private ButtonGroup[] arrayGroups;
  + private JPanel[] questionPanels;
  + private CardLayout cl;
  + private JLabel lb1, lb2, lb3;
  + private JButton nextBtn, answerBtn, scoreBtn, quitBtn;
  + private int score;
  + String playerName = "";
  + QuizCollection q = null;
  + ArrayList<QuizQuestion> quizQuestions = null;
  + ScoreManager scoreMgr = null;
  + **Methods:** void initialize()
  + void visualize()
  + static boolean isValidName(String name)
  + void main(String[] args)

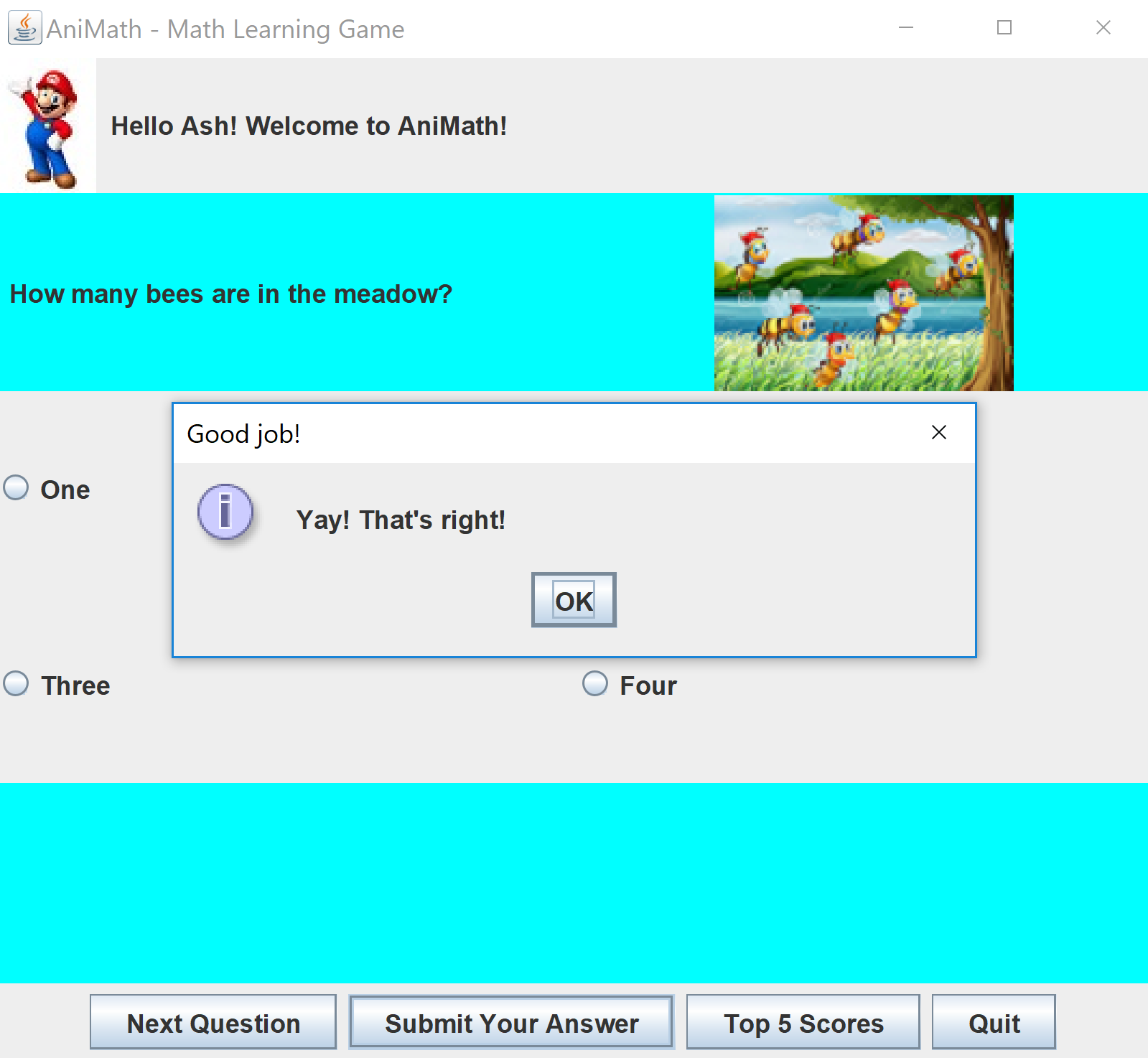
**GAMEPLAY**

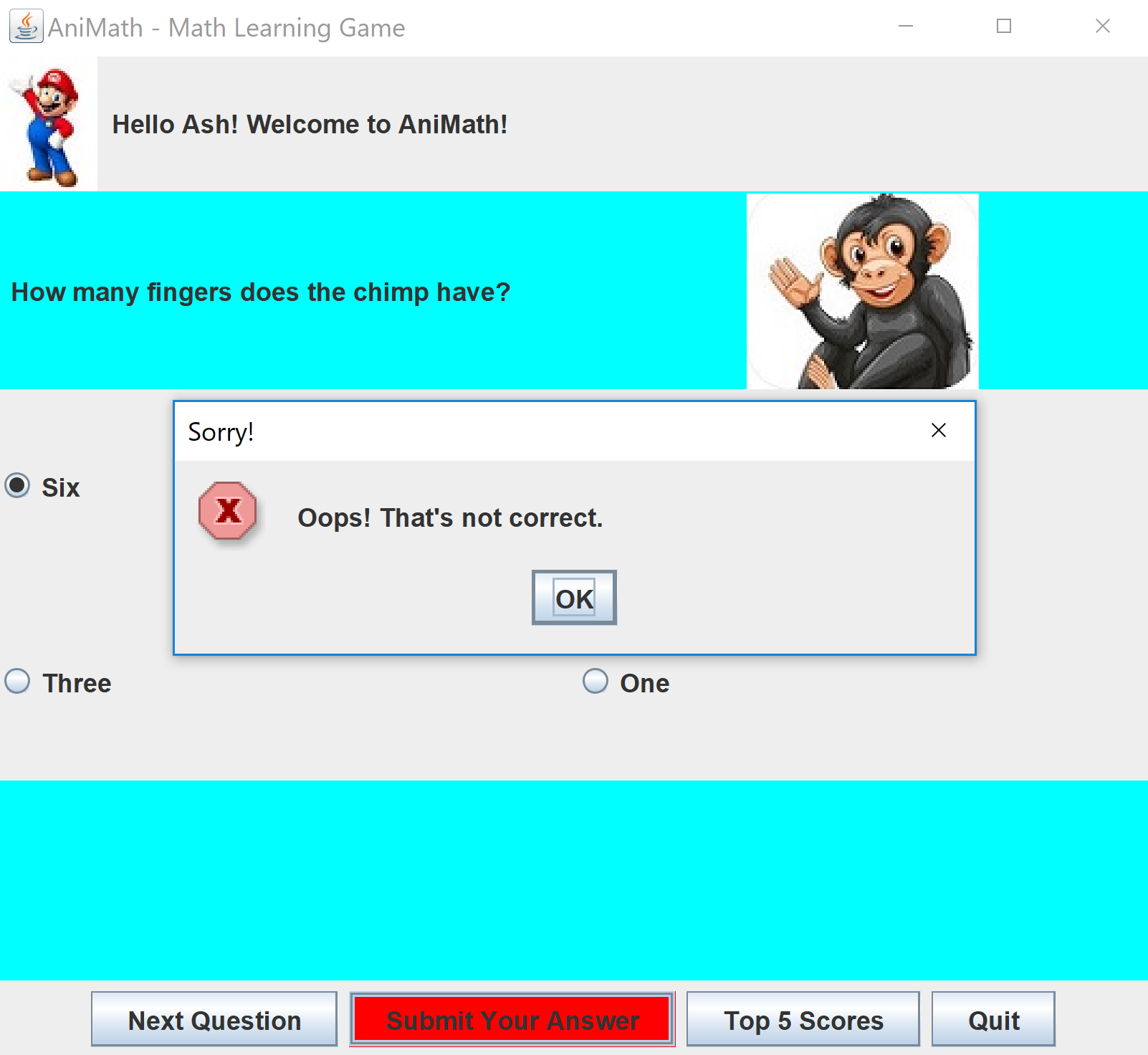
This is the first dialog box that pops up when the MathQuiz class is run. The program validates the name entered.

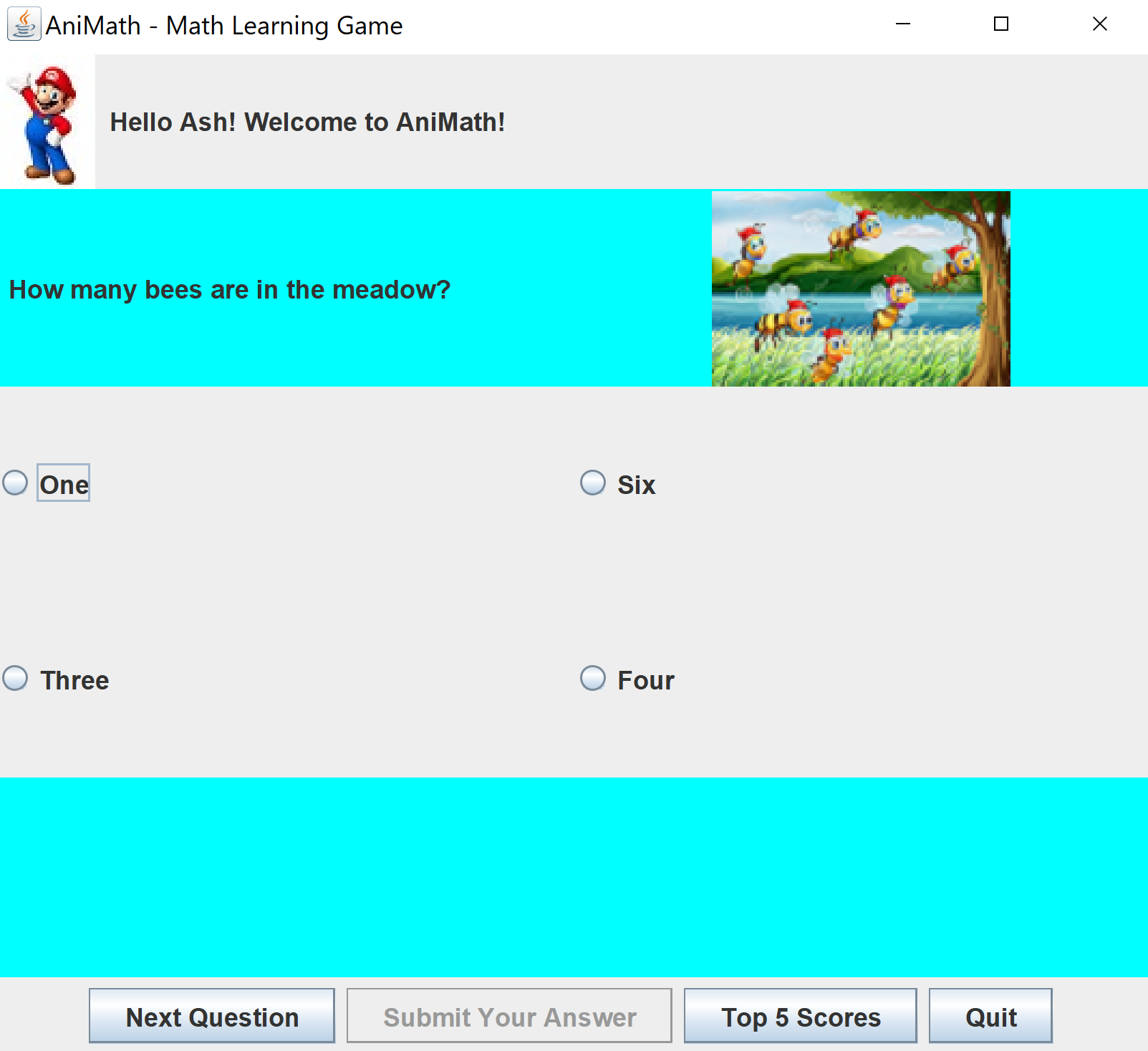
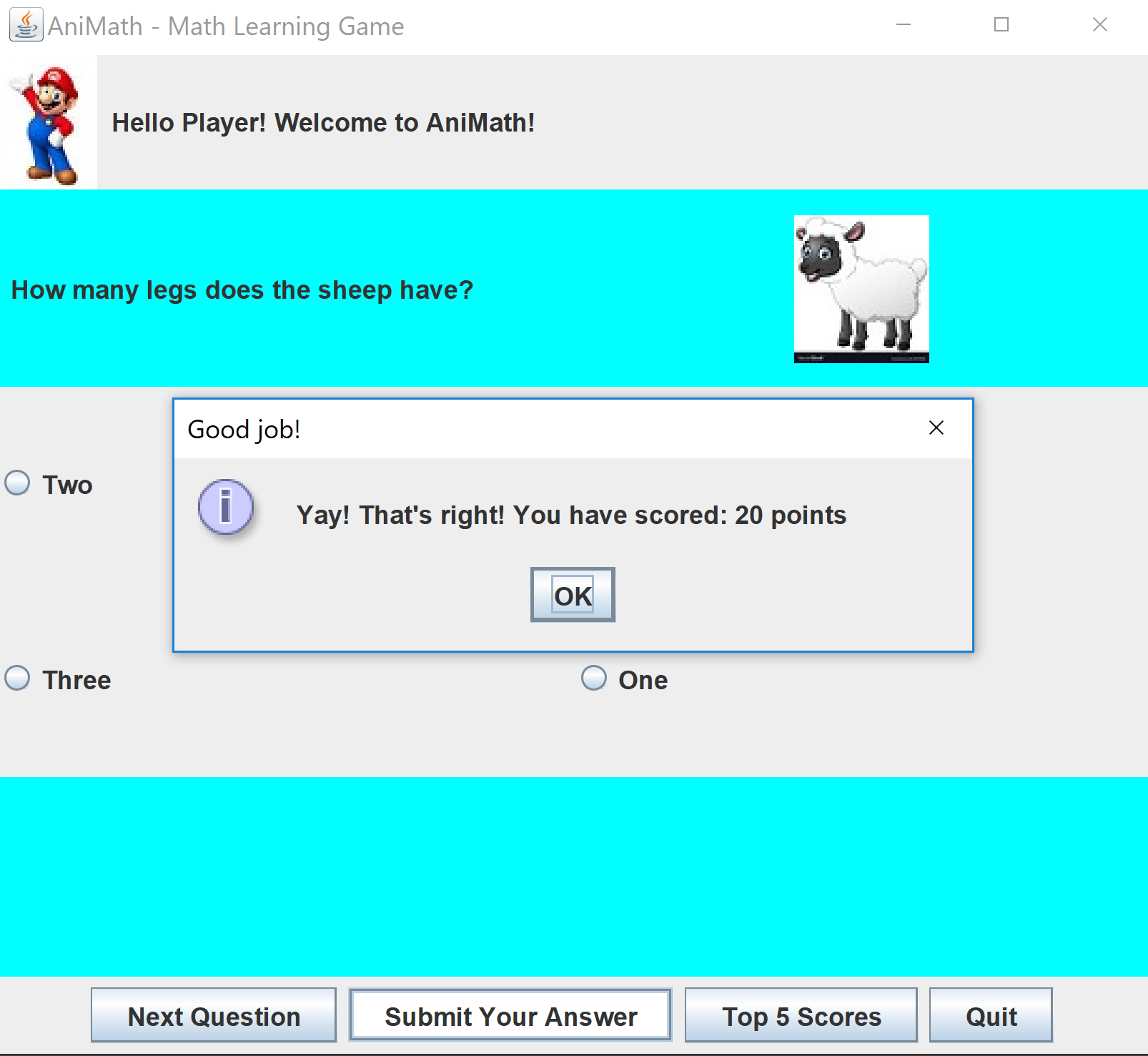


When “OK” is clicked, the game begins. The top panel welcomes the user with the name they entered and the panel below asks the first question, accompanied with a picture. Four choices are provided for each question with one correct answer. Once the player has clicked on an answer, they have the option to submit it, as seen below.

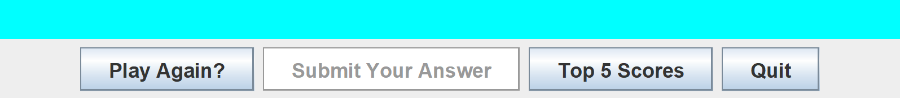
If they answer it correctly, then the following message box appears:



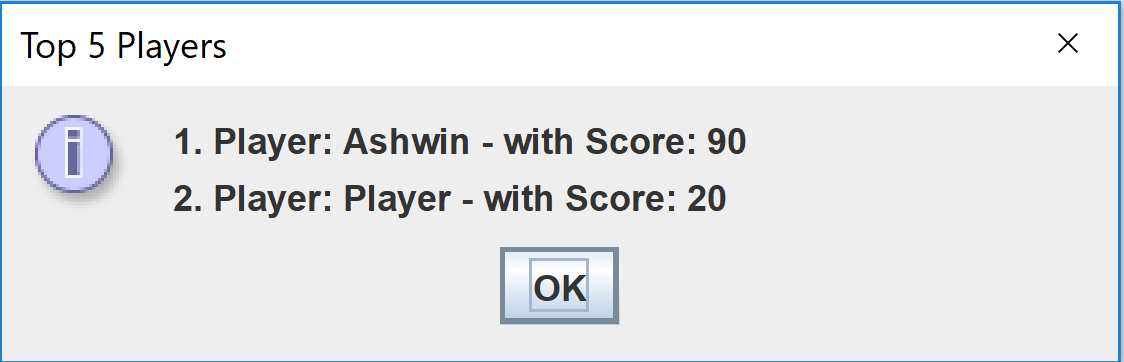
Otherwise:

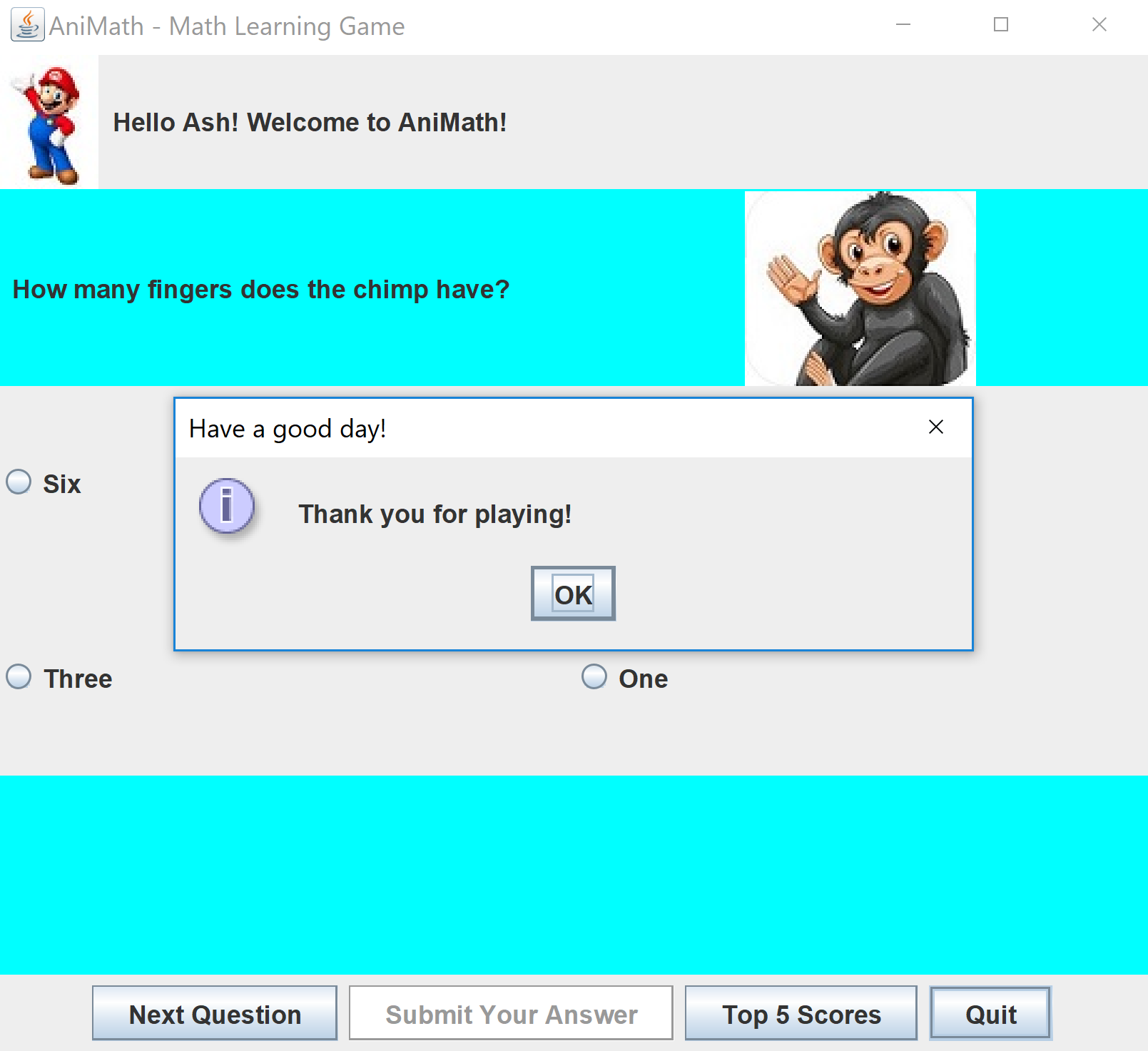
From here, the player can choose to proceed to the next question.

This continues for nine more questions. Once they answer the final question, this message pops up:



The player is given the option to restart, if they wish.

At any point during the game, the player can choose to view the top 5 scores. Once they press the button, the box to the left displays. Once the game is played at least 5 times, the highest 5 scorers and their scores with be displayed in order.



Once the player decides to quit, they can press the button and the following message will appear:

# CLASS DIAGRAM

The class diagram in the Unified Modeling Language (UML) below shows the classes, their attributes, methods and relationship between different classes.

As shown below, the diagram shows relationships between classes, like Association. Some of the examples are:

* MathQuiz has 0-1 association with the QuizCollection. This is true because MathQuiz has a single QuizCollection ArrayList
* QuizCollection has 0-\* association with QuizQuestion. This means that there can be 0 to any number of QuizQuestions in a QuizCollection List
* ScoreManager has 0-\* association with PlayerScore. This means that there can be 0 to any number of scores handled by the ScoreManager

