

CSE 2050 – Programming in a Second Language

Fall 2023

Final Project: DMV Driver's Test UI

Total Points: 30

Date Assigned: Monday, Nov 13, 2023

Due Date: Sunday, Dec 2, 2023

Instructions: Please submit your work on Canvas as a zipped file named `cse2050_group_x_final_project.zip`. Make sure to include a `main.py` file as the driver for testing your program and organize all classes into separate files. Do not use global variables, packages outside of the Python Standard Library (except `lxml`), or concepts not covered in class to complete this assignment.

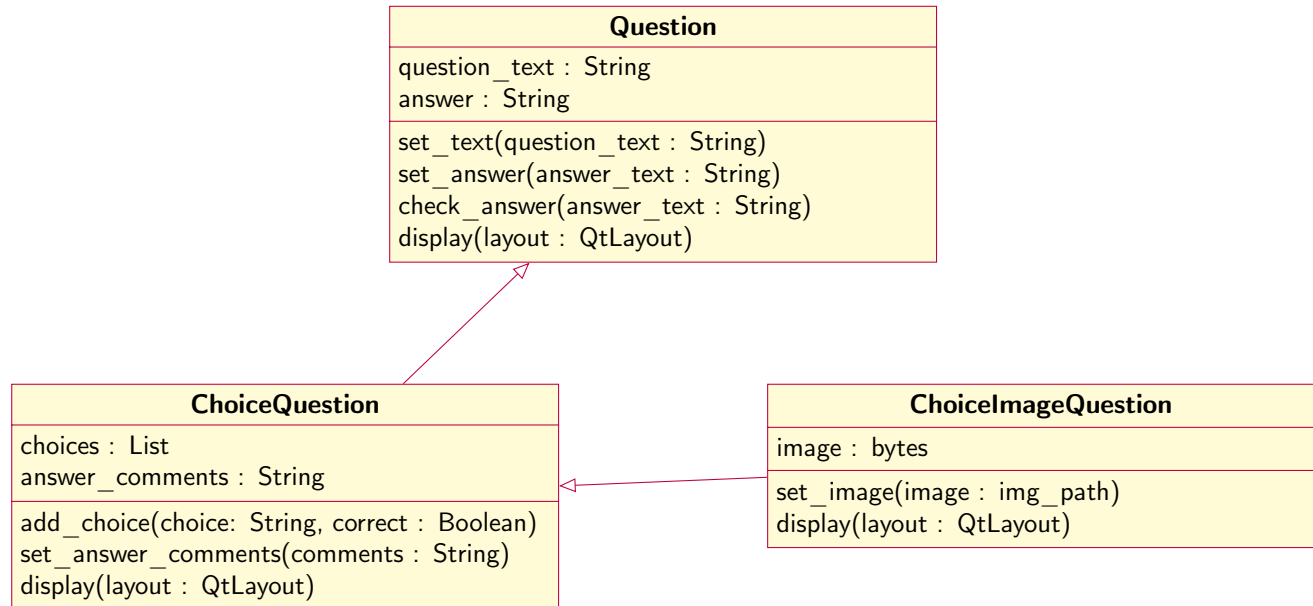
Key Concepts Demonstrated

- Writing GUI programs based on the Objected Oriented Programming (OOP) Paradigm
 - Writing code that uses objects that interact with each other
- Parsing XML files and creating objects from the data
- Solving a real-world problem
- Interpreting UML diagrams and writing code based on the contents

1. (30 points) DMV Driver's Test UI

Given an XML file named `florida_drivers_test.xml` and a folder of images located on Canvas, containing mock questions for a driver's test, which is typically given in Florida, complete the following:

Write an OOP PyQt program based on the following UML class diagram and requirements listed below



Requirements:

Your project should have the following as a minimum:

Question, ChoiceQuestion, ChoiceImageQuestion - (8pts)

These classes should be based on the UML diagram above. The sub-question objects should override the methods from the super class. For example, a choice question without an image should be instantiated as a `ChoiceQuestion` object whereas one with an image should be instantiated as a `ChoiceImageQuestion`.

XMLParser - (10pts)

This class has a method that parses the given XML file and creates question objects based on the question elements in the file.

DVMDriverTestUI - (8pts)

This is the main UI that will accept a list of question objects in its constructor and display them one question at a time. See an image of the expected UI in Figure 1 with a `ChoiceImageQuestion`. Figure 2 shows an example layout for a `ChoiceQuestion` and Figure 3 provides tips on how to layout your UI.

When the user clicks on a radio-button, this should disable all radio buttons (`btn.setDisabled(True)`) and display the answer-comments in a widget such as a label below the question section of the UI.

Clicking the *next-question* button should update the UI with a message showing the index of the correct question and the number of correct answers the user has gotten so far. The `check_answer` method, which is part of the question object, should be used to check if a user got a question right, and the UI should be updated accordingly.

main() method - (4pts)

- creates an instance of the `XMLParser` that reads the XML file and stores question objects in a list
- creates an instance of the `DVMDriverTestUI` and displays one question at a time allowing the user to click a *next-question* button to advance to the next question (see image overleaf)

Hints:

- See slides 12 - 20 (UI Design) for a discussion on different concepts related to the `Question` class.
- See slide 77 for code related to adding an image to a PyQt UI
- See slide 80 - 82 for tips on making a quiz UI and clearing a layout in order to add new items

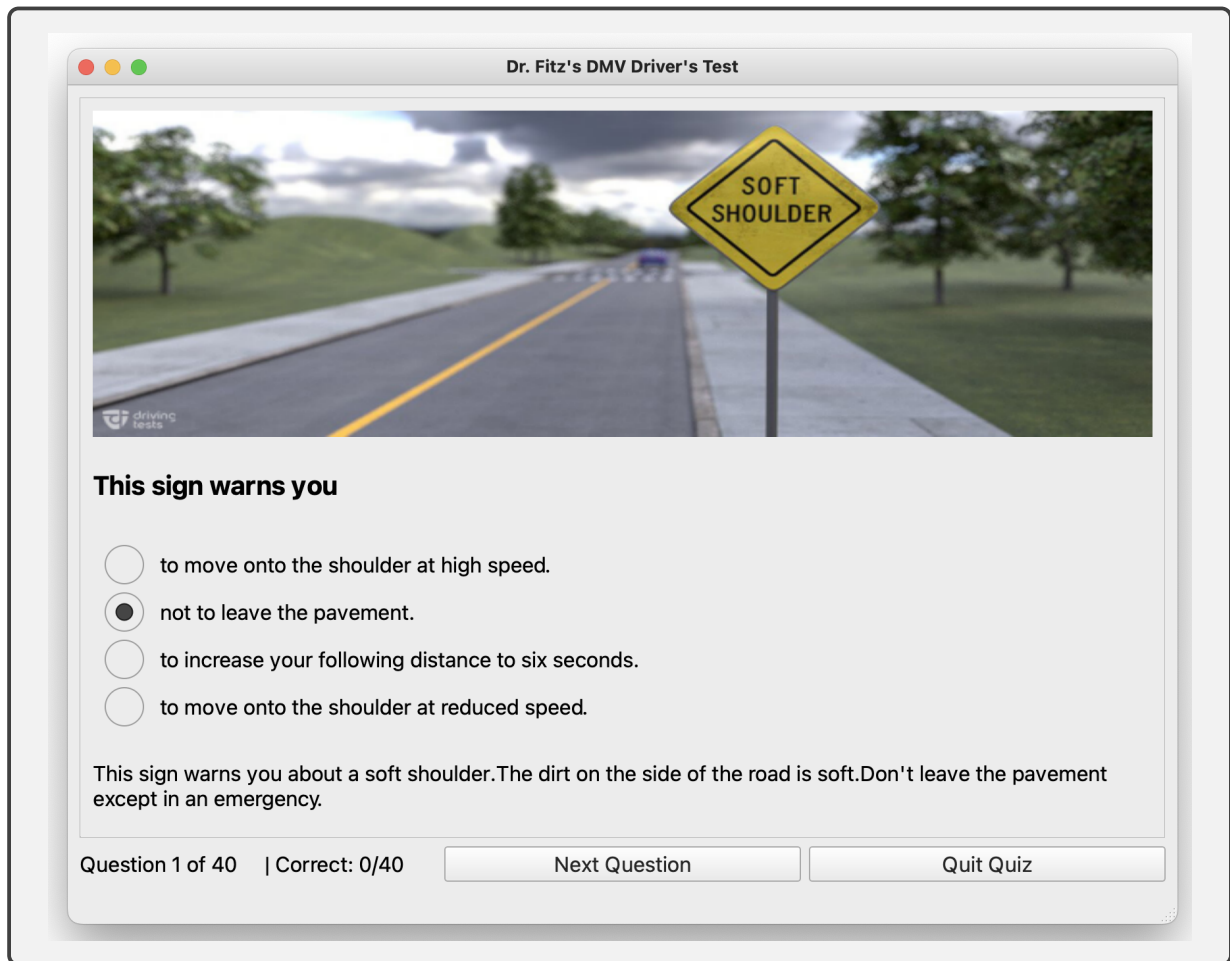


Figure 1: Expected UI for the DMV Driver's Test

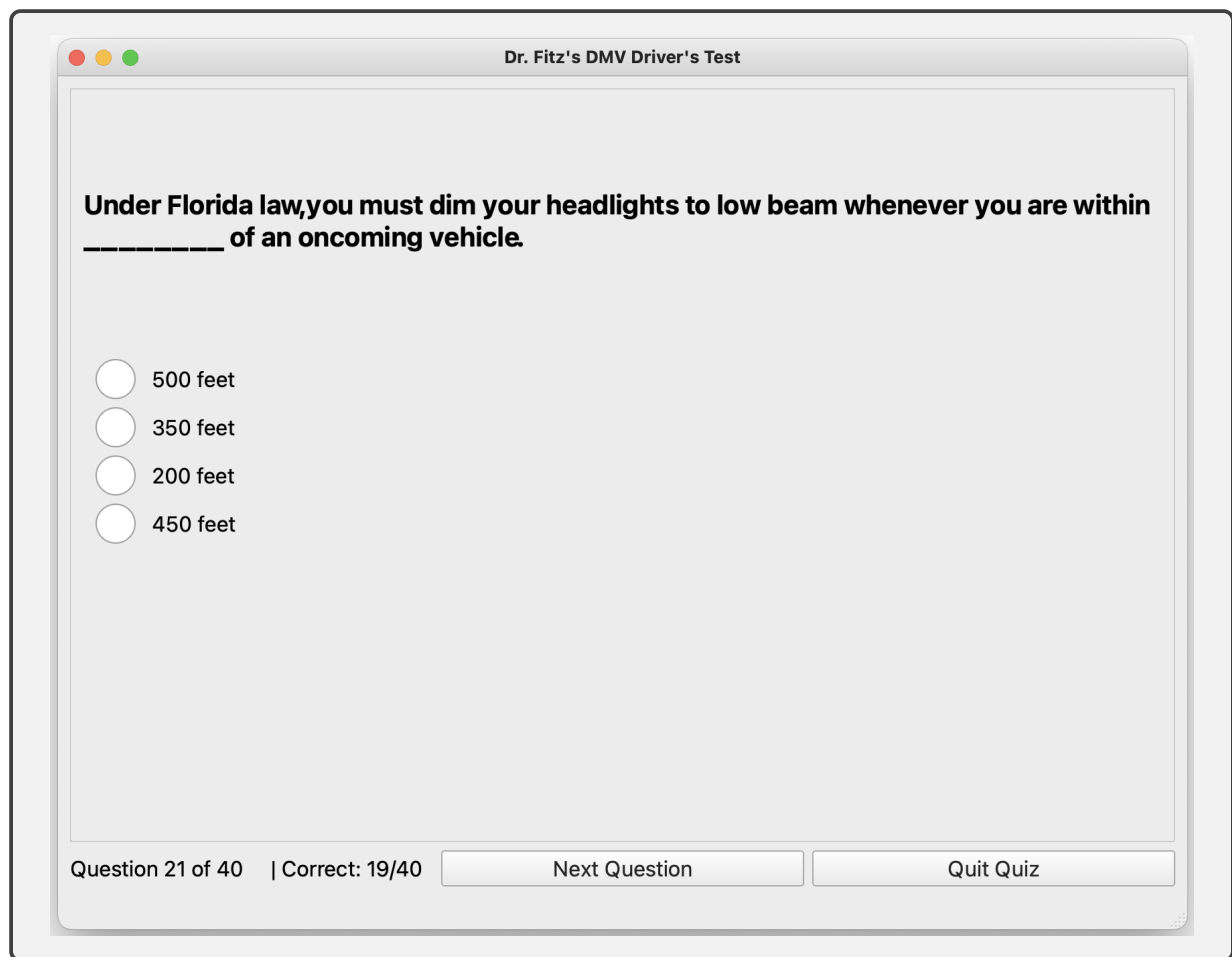


Figure 2: Example DMV Driver's Test UI showing a ChoiceQuestion (i.e., a question without an image)

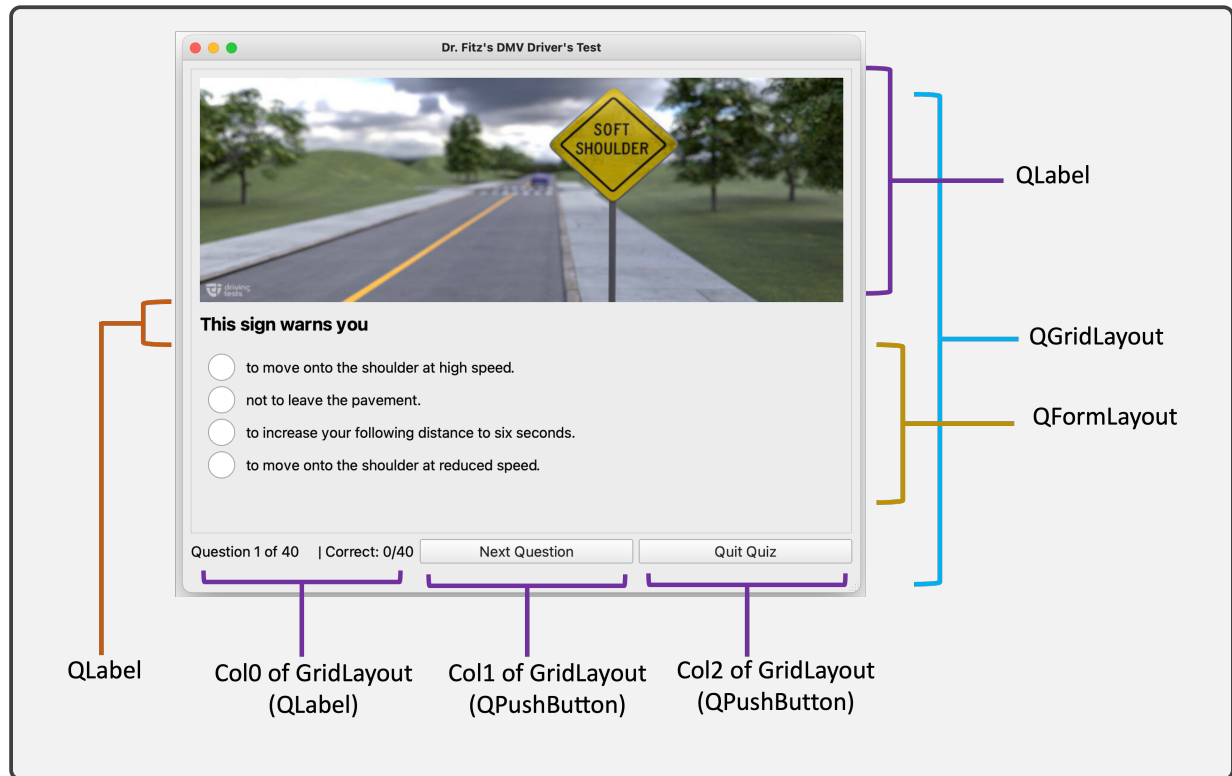


Figure 3: Design Tips for your UI. To set the font-size and weight of a label, use `label.setStyleSheet("font-size:20px;font-weight:600")`