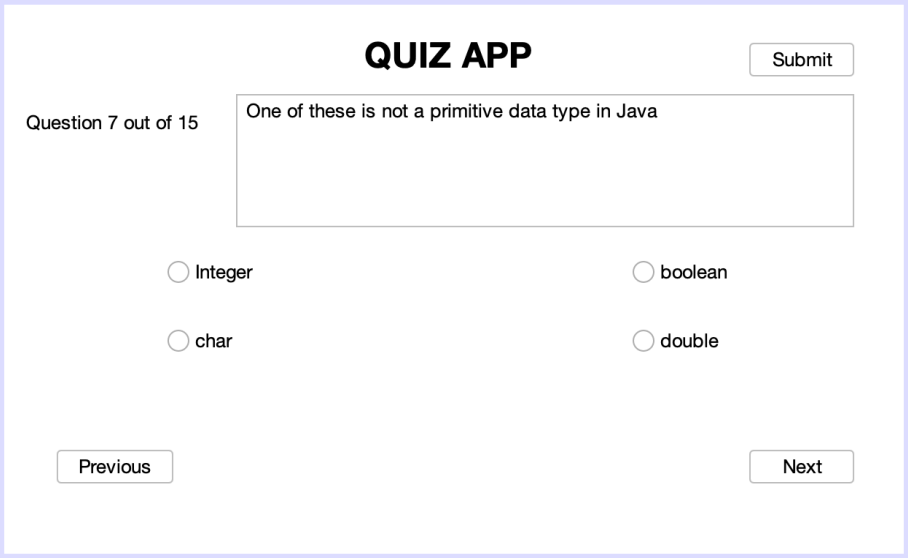
CSCI 218: Programming II (Spring 2025)

Week 8 Lab Activity: Files and IO Streams

Task 1: (Quiz Application)



Create a GUI-based quiz application that presents users with multiple-choice questions, allows them to select their answers, and provides feedback on their performance. The application should have the following components:

- A label to display the question.
- (4) Radio buttons for each answer option.
- "Next" and "Previous" buttons that allow users to navigate between questions.
- "Submit" button to submit the answers and display the user's score.

Additional requirements:

1. Before the start of the quiz, the question set should be loaded from a text file and the name of the user should be requested for.
2. At the end of the quiz, calculate the user's score based on the number of correct answers and display the user's score using a label and a progress bar.
3. Save the user's score and name to a file to track quiz history. Additionally, implement functionality to display a chart representing the quiz history.

4. Provide a feedback option to display the questions, users' choice and the correct answers
5. Hints can be included for enhanced functionality for each question.
6. Randomize the order of questions to prevent predictability.

Task 2: (Developing an Application for Manipulating Subtitles)

- **Introduction**

  Subtitles are captions displayed at the bottom of a cinema or television screen that translate or transcribe the dialogue or events taking place in the video (see Figure 1 for an example). Subtitles can be hard-coded into the video stream or come as a separate file. The goal of this project is to develop an application for manipulating subtitle files.

- **Data Format**

  There are many file formats for representing subtitles, but arguably the simplest of these formats is the SubRip format, also known as SRT format because of its file extension. An SRT file is a text file consisting of a sequence of subtitles, each having the following structure:

  1. A sequential number which starts at 1.
  2. The start and end times when the subtitle should appear on screen. The time is specified in the format: HH:MM:SS,ms (the last field is an integer value in milliseconds). The display intervals of subtitles must not overlap.
  3. The text of the subtitle. Line breaks are allowed.
  4. An empty line indicating the start of a new subtitle (the last subtitle is not followed by a an empty line).

  The following shows sample subtitles from the movie The Chosen Season 1: Link

```
758
00:51:12,110 --> 00:51:15,071
Who are you? How do you know my name?

759
00:51:15,155 --> 00:51:17,824
"Thus says the Lord who created you."

760
00:51:20,202 --> 00:51:22,788
"And He who formed you."

761
00:51:25,081 --> 00:51:26,500
"Fear not…

762
00:51:29,586 --> 00:51:31,880
for I have redeemed you."

763
00:51:34,007 --> 00:51:36,009
"I have called you by name."

764
00:51:38,178 --> 00:51:42,224
"You… are mine."
```

- **Requirements**

  In this phase, you are required to implement the following classes and interfaces (**This specification must under no circumstances be modified**):

// Interface representing time

```
public interface Time {

    int getHH();
    int getMM();
    int getSS();
    int getMS();
    void setHH(int hh);
    void setMM(int mm);
    void setSS(int ss);
    void setMS(int ms);

}
```

// This interface represents a single subtitle.

```
public interface Subtitle {
// Return the start time of the Subtitle.
Time getStartTime();

    // Return the end time of the Subtitle.
    Time getEndTime();

// Return the subtitle text.
    String getText();

// Set the start time of the Subtitle.
    void setStartTime(Time startTime);

// Set the end time of the Subtitle.
void setEndTime(Time endTime);

// Set the subtitle text.
    void setText(String text);

}
```

```
// This interface represents a subtitle sequence.

 public interface SubtitleSeq {

   // Add a subtitle.
void addSubtitle(Subtitle st);

// Return all subtitles in their chronological order.
List<Subtitle> getSubtitles();

/**
      * Return the subtitle displayed at the specified time, null if no
* subtitle is displayed.
      * @param time
      * @return
      */
Subtitle getSubtitle(Time time);

/**
      *   Return, in chronological order, all subtitles displayed between the
      *   specified start and end times. The first element of this list is the
      *   subtitle of which the display interval contains or otherwise comes
      *   Immediately after startTime. The last element of this list is the
      *   subtitle of which the display interval contains or otherwise comes
      *   immediately before endTime.
      * @param startTime
      * @param endTime
      * @return
      */
List<Subtitle> getSubtitles(Time startTime, Time endTime);

// Return, in chronological order, all subtitles containing str as a //
sub-string in their text.
List<Subtitle> getSubtitles(String str);

// Remove all subtitles containing str as a sub-string in their text.
void remove(String str);

// Replace str1 with str2 in all subtitles.
void replace(String str1, String str2);

/**
      *    Shift the subtitles by offseting their start/end times with the
specified
      *  offset (in milliseconds). The value offset can be positive or negative.
      *   Negative time is not allowed and must be replaced with 0. If the end
time
      *  becomes 0, the subtitle must be removed.
      * @param offset
      */
void shift(int offset);

/**
```

```
      * Cut all subtitles between the specified start and end times. The first
subtitle to be removed is the one for which the display interval contains or
otherwise comes immediately after startTime. The last subtitle to be removed is
the one for which the display interval contains or otherwise comes immediately
before endTime. The start and end times of all subtitles must be adjusted to
reflect the new time. */
void cut(Time startTime, Time endTime);
}
```

```
public class SubtitleSeqFactory {
// Return an empty subtitles sequence
public static SubtitleSeq getSubtitleSeq() { }

   // Load a subtitle sequence from an SRT file. If the file does not exist or
// is corrupted (incorrect format), null is returned.
public static SubtitleSeq loadSubtitleSeq(String fileName) {
}
}
```

The specification given above (class and interface names, and method signatures) must not be modified. Any change to the specification results in compilation errors and would count for zero grade.