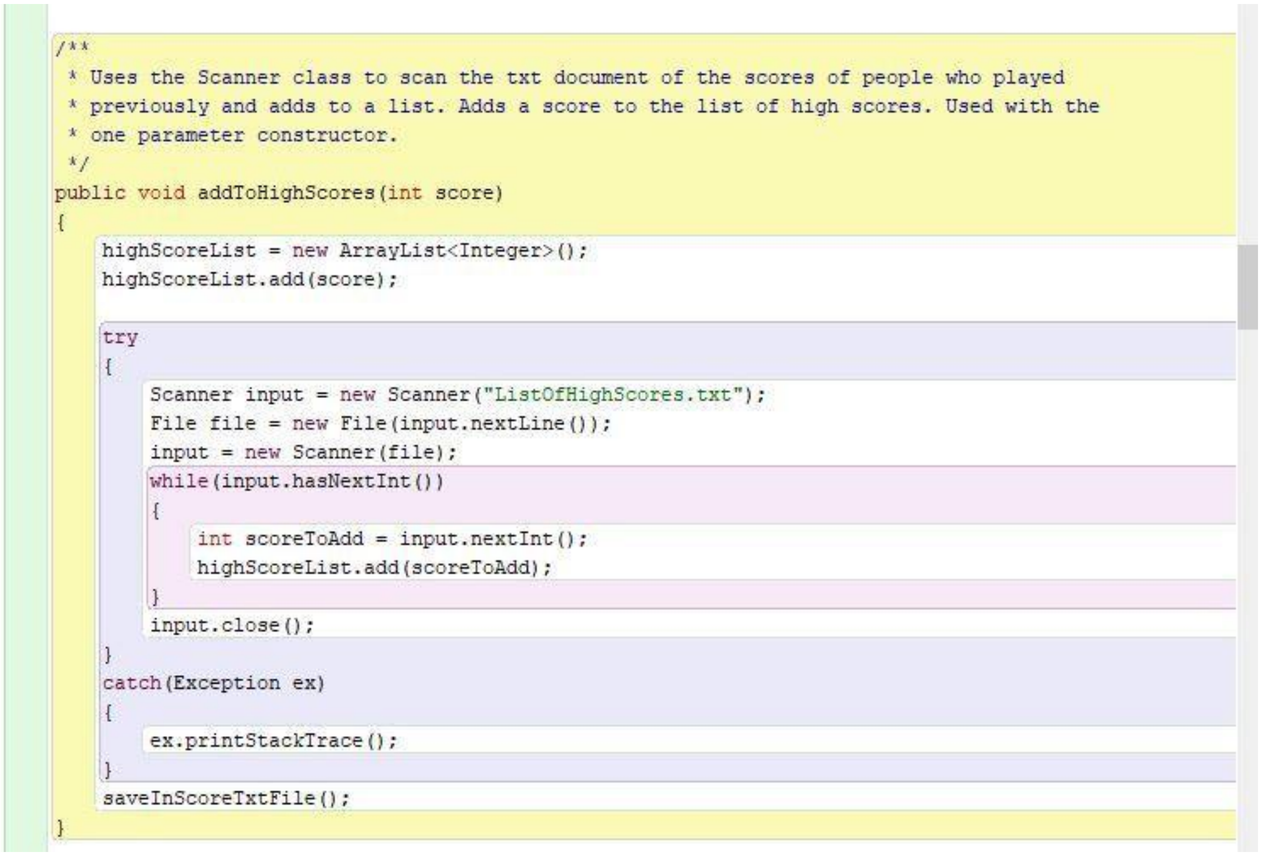
AP Computer Science Principles: 2020 Create Task
Student Sample B

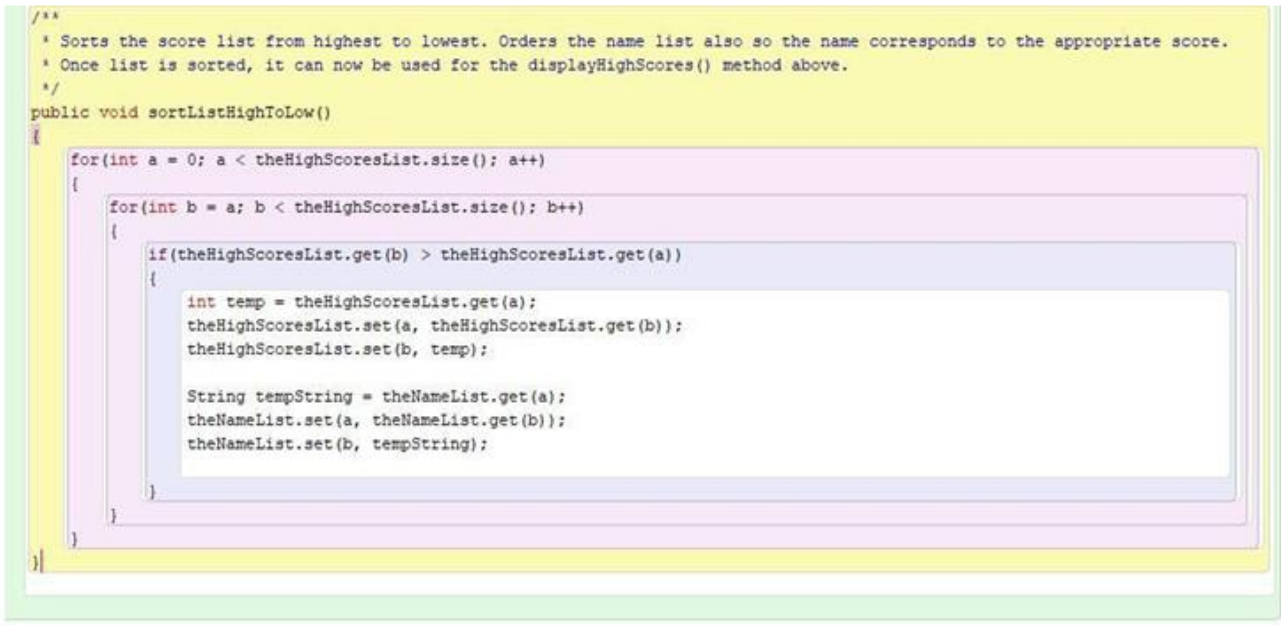
3a)
My Program is called Lee's Typing Game. It is a computer game that allows people to practice their typing. Introducing a variety of sentences, my program helps with improving their typing skills with a score feature that motivates the user to become better and type more quickly and accurately. In the video, the opening screen explains the game's objective. Upon pressing start, on the first sentence, I incorrectly input the sentence and press done, which outputs a penalty on the timer and notifies the user. After I correctly type it and press done, the output is that the timer restarts and the score updates. I then continue playing the typing game. After finishing typing 5 sentences, the game shows the user their score and the ability to add their name to the high score list, with limits to their name.
Then, it demonstrates the unique feature of the high scores.

3b)



```
/**
 * Uses the Scanner class to scan the txt document of the scores of people who played
 * previously and adds to a list. Adds a score to the list of high scores. Used with the
 * one parameter constructor.
 */
public void addToHighScores(int score)
{
    highScoreList = new ArrayList<Integer>();
    highScoreList.add(score);

    try
    {
        Scanner input = new Scanner("ListOfHighScores.txt");
        File file = new File(input.nextLine());
        input = new Scanner(file);
        while(input.hasNextInt())
        {
            int scoreToAdd = input.nextInt();
            highScoreList.add(scoreToAdd);
        }
        input.close();
    }
    catch(Exception ex)
    {
        ex.printStackTrace();
    }
    saveInScoreTxtFile();
}
```



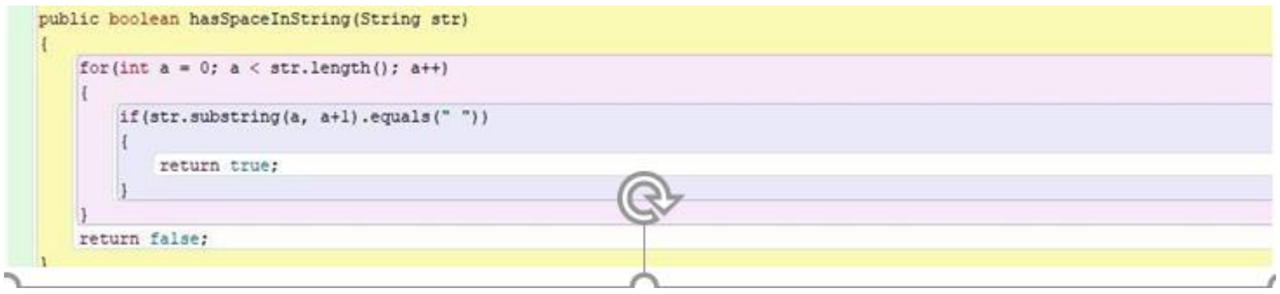
```
/**
 * Sorts the score list from highest to lowest. Orders the name list also so the name corresponds to the appropriate score.
 * Once list is sorted, it can now be used for the displayHighScores() method above.
 */
public void sortListHighToLow()
{
    for(int a = 0; a < theHighScoresList.size(); a++)
    {
        for(int b = a; b < theHighScoresList.size(); b++)
        {
            if(theHighScoresList.get(b) > theHighScoresList.get(a))
            {
                int temp = theHighScoresList.get(a);
                theHighScoresList.set(a, theHighScoresList.get(b));
                theHighScoresList.set(b, temp);

                String tempString = theNameList.get(a);
                theNameList.set(a, theNameList.get(b));
                theNameList.set(b, tempString);

            }
        }
    }
}
```

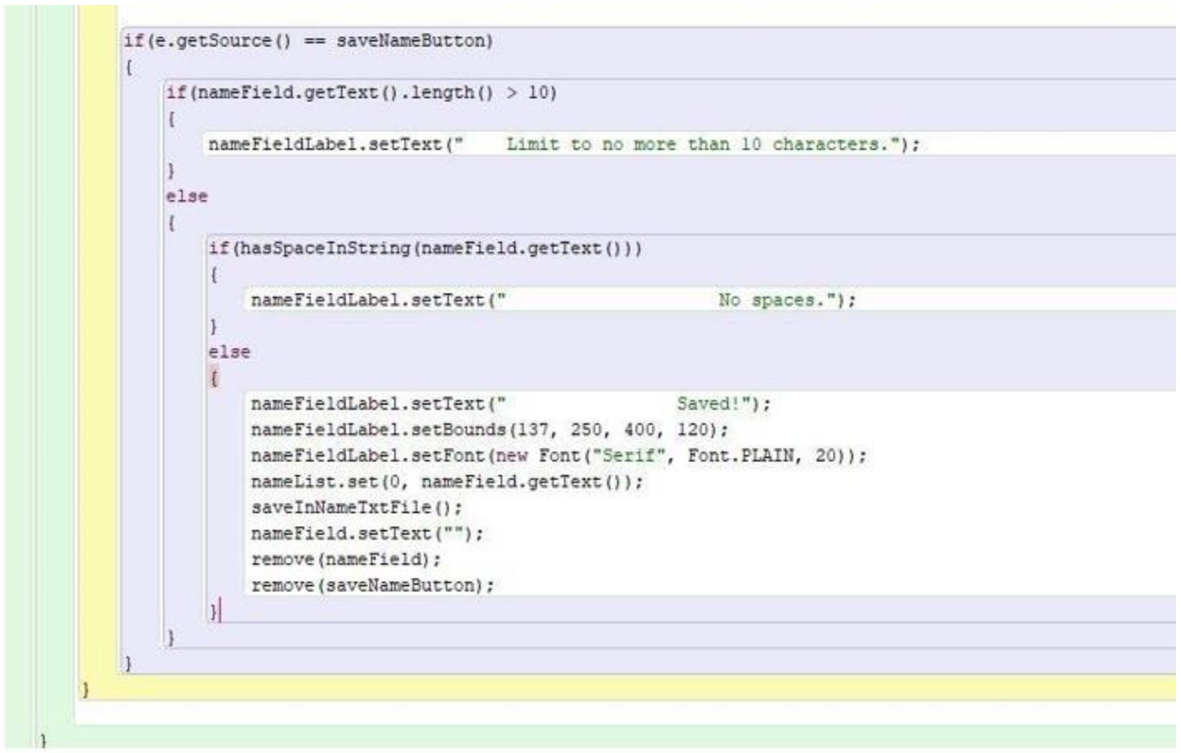
For clarification, these two program segments are in different classes, but they refer to the same list: “highScoreList” and “theHighScoresList”. The first method, addToHighScores(), which is in the GameOverPanel class, adds the user’s final score to a list that represents all the previous scores of people who played the program before. The list then adds the scores of the previous people by scanning the text document that holds the info. Then, the list of all the scores is received by the HighScorePanel class where the scores are sorted from highest to lowest in the method sortListHighToLow() so that it can display the list of high scores accordingly. This list manages complexity in the program because it represents saved data from previous players of the game, which involved the use of the Scanner class and using a text document. This allows the high scores to be sorted more easily and then resaved to the file in a new sorted order. The saved data is processed for displaying the high scores and without this list, there would not be a way to insert a new high score into the file since we don’t know how many separate variables we would need.

3.C.



```
public boolean hasSpaceInString(String str)
{
    for(int a = 0; a < str.length(); a++)
    {
        if(str.substring(a, a+1).equals(" "))
        {
            return true;
        }
    }
    return false;
}
```

Call to the hasSpaceInString procedure



```
            if(e.getSource() == saveNameButton)
            {
                if(nameField.getText().length() > 10)
                {
                    nameFieldLabel.setText("    Limit to no more than 10 characters.");
                }
                else
                {
                    if(hasSpaceInString(nameField.getText()))
                    {
                        nameFieldLabel.setText("                    No spaces.");
                    }
                    else
                    {
                        nameFieldLabel.setText("                Saved!");
                        nameFieldLabel.setBounds(137, 250, 400, 120);
                        nameFieldLabel.setFont(new Font("Serif", Font.PLAIN, 20));
                        nameList.set(0, nameField.getText());
                        saveInNameTxtFile();
                        nameField.setText("");
                        remove(nameField);
                        remove(saveNameButton);
                    }
                }
            }
        }

    }
```

For clarification, the boolean hasSpaceInString() works with the bottom code segment for saveNameButton. This part of the program is used when the user finishes the game and they're on the part where they're saving their name associated with their score, so it can be displayed on the high scores. The code segments for this situation manages the user's input for their score name which includes how long their username is and if there are spaces involved. The code makes sure the user's name is not longer than 10 characters using the String's length() method, and the code hasSpaceInString() finds if there is a space in their name using iteration. Essentially, these are limits to what the user's name can be saved as for their high. score.

3.D.

The first test case for procedure "hasSpaceInString" is to pass a string that has a space in it, such as "Mr Guy". When a string with a space is passed to parameter str, the procedure will return true.

The second test case for procedure "hasSpaceInString" is to pass a string that doesn't have a space in it, such as "Mr.Clean". When a string without a space is passed to parameter str, the procedure will return false.