CS 3360 - Design and Implementation of Programming Languages

PROJECT 2: ASPECT-ORIENTED PROGRAMMING WITH ASPECTJ

(File $Date: 2018/11/09 18:08:20 $)

Due: November 15, 2018

This assignment may be done individually or in pairs. If you work in pair, you need to fill out the contribution form (see the course website).

The purpose of this assignment is to understand basic concepts of aspect-oriented programming (AOP) and have a taste of AOP by writing a small AspectJ program [1].

You are to to develop an AspectJ application for playing Connect Four games. The Connect Four game is a two-player connection game in which the players take turns dropping their discs from the top into a seven-column, six-row vertically suspended grid [Wikipedia]. The pieces fall straight down, occupying the next available space within the column. The objective of the game is to connect four of one's own discs next to each other vertically, horizontally, or diagonally before the opponent.

Download the baseline Java code named c4-base-src.zip from the course website. The Javadoc API specification and an executable jar file are also available from the website. The baseline code lets a single player drop his/her discs in the columns until all the columns are filled, and the game state is shown as a 2D grid of colored discs drawn using the Java 2D graphics API. As depicted by its UML class diagram [2] shown in Figure 1 below, it uses the Model-View-Control

(MVC) metaphor [3] to separate the model classes such as Board and Player from the view and control classes (C4Dialog, BoardPanel, and ColorPlayer).

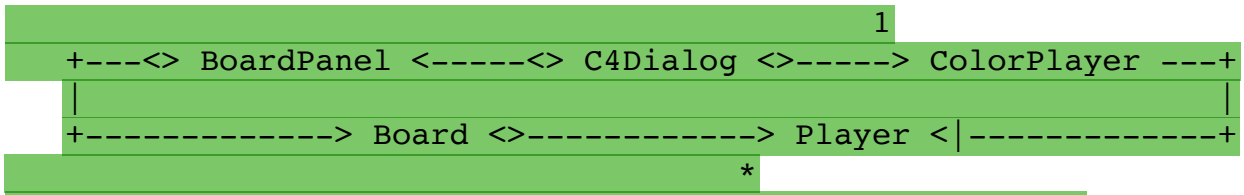

Figure 1. UML class diagram of the baseline code

You are to extend the baseline code to add several new features using AspectJ. DO NOT EDIT THE SOURCE CODE OF THE BASELINE JAVA PROGRAM. Do not duplicate in your AspectJ code the features that are already implemented in the baseline code.

1. (25 points; 64 lines of source code) Write an aspect named pressDisc to improve user experience of the application. The aspect shall highlight the disc to be dropped by showing a different display, e.g., an outlined disc, when the user click it. It shall give an impression that the disc rises up towards the user, or gets depressed away.

   Hint: When a mouse is pressed on a droppable disc, show a raised (or depressed) display of the disc. When a mouse is released, show the regular display.

   Hint: The following methods and classes/interfaces may be useful.
   int BoardPanel.locateSlot(int,int)
   void BoardPanel.drawChecker(Graphics, Color, int, int, int)
   void BoardPanel.addMouseListener(MouseListener)*
   MouseListener interface: mousePressed, mouseReleased
   MouseAdapter class
   *Inherited from Component

2. (25 points; 89 loc) Write an aspect named EndGame to end the game and display the outcome: win or draw (see Problem 4 below for a draw). When the game ends (a sequence of four discs), no more disc can be dropped in the slots.

- Display the game outcome (win or draw) in the message bar.

- Highlight the winning sequence of discs in the board panel.

- Change the behavior of the "new" button to not prompt the user when the game is over.

Hint: The following methods may be useful.

```
void C4Dialog.showMessage(String)
boolean Board.isGameOver()
boolean Board.isFull()
Player Board.playerAt(int, int)
void Board.dropInSlot(int, Player)
boolean Board.isWonBy(Player)
Iterable<Board.Place> Board.winningRow()
void BoardPanel.drawChecker(Graphics, Color, int, int, boolean)
```

3. (25 points; 72 loc) Write an aspect named AddSound to add some sound effect to the application.

- Play a sound when a player drops a disc. Use a distinct sound for each player (see Problem 4 below).

- Play a sound of applause, or cheering, when a player wins (see Problem 2 above).

Hint: You may use the following code snippet to play an audio clip. It assumes that audio files are stored in the directory src/sound.

```
/** Directory where audio files are stored. */
private static final String SOUND_DIR = "/sound/";
```

```
    /** Play the given audio file. Inefficient because a file
will be
     * (re)loaded each time it is played. */
    public static void playAudio(String filename) {
      try {
          AudioInputStream audioIn =
AudioSystem.getAudioInputStream(
        AddSound.class.getResource(SOUND_DIR + filename));
          Clip clip = AudioSystem.getClip();
          clip.open(audioIn);
          clip.start();
      } catch (UnsupportedAudioFileException
              | IOException | LineUnavailableException e) {
          e.printStackTrace();
      }
    }
```

4. (25 points; 74 loc) Write an aspect named AddOpponent to support two-player games. Your aspect shall add a new player, say a "red" player. The two players alternate in dropping a disc of their color into a slot of the board. The winer is the first player to get an unbroken row of four discs. Your aspect shall display the player's turn. With the EndGame aspect (see Problem 2 above), your aspect shall allow users to play complete Connect Four games.

5. (20 bonus points; 263 loc) Write an aspect named AddCheatKey to add a cheat key, say F5, to enable and disable a cheat mode. In the cheat mode, the aspect will hint/warn a winning/loosing row. A winning/loosing row is a row that can win/loose a game with one more move, e.g., an open row of three consecutive discs of the same player, horizontally, vertically, or diagonally. When there exists a winning/loosing row, your aspect should highlight it on the board, say, in a different color.

Bonus: Identify a sequence of four discs with an empty place embedded,
e.g., XOO_OX, and a sequence of three with an empty place embedded
and both ends open, e.g., _O_OO_.

Hint: Use "key bindings" to implement a cheat key. For example, let
the BoardPanel instance to run the following code snippet to
respond to an F5 key click.

```
  ActionMap map = getActionMap();
  int condition = JComponent.WHEN_IN_FOCUSED_WINDOW;
  InputMap inputMap = getInputMap(condition);
  String reload = "Cheat";
  inputMap.put(KeyStroke.getKeyStroke(KeyEvent.VK_F5, 0), reload);
  map.put(reload, new KeyAction(this, reload));
```

And define the KeyAcion class in your aspect as follows.

```
  @SuppressWarnings("serial")
  private static class KeyAction extends AbstractAction {
     private final BoardPanel boardPanel;

     private KeyAction(BoardPanel boardPanel, String command) {
         this.boardPanel = boardPanel;
         putValue(ACTION_COMMAND_KEY, command);
     }

     /** Called when a cheat is requested. */
     public void actionPerformed(ActionEvent event) {
         // code to be run when the cheat (F5) key is pressed.
         // ...
     }
  }
```

Identifying a winning/loosing row is somewhat similar to finding a
winning row (see boolean Board.isWonBy(Player) method).

TESTING

Your code should compile with AspectJ 1.8 or later and run

correctly under Java 1.8 or later version. You should test your programs thoroughly.

WHAT AND HOW TO TURN IN

Submit your program through the Assignment Submission page located in the Homework section of the course website. Your program submission should include the following:

- c4.jar: runnable jar file containing bytecode and support files (e.g., audio clips)
- src directory of source code files
- contribution-form.docx (only for pair work)

The submission page will ask you to zip your program and upload a single zip file. Your zip file should include only a single directory named YourFirstNameLastName containing all your source code files and other support files needed to compile and run your program. DO NOT INCLUDE BYTECODE (.class) FILES. There is a limit on upload file size and the maximum file size is 2MB. You should turn in your programs by 11:59 pm on the due date.

If you work in pair, make only one submission through the Assignment Submission page by specifying both names during the submission; make sure to include the contribution form in your submission.

Make sure that your jar file is indeed executable. You will need to make it self-contained by including the AspectJ runtime library classes that comes with your AspectJ distribution, typically named aspectjrt.jar or org.aspectj.runtime_1.8.6.XXXXXXXX.jar (AJDT).

GRADING

You will be graded, in part, on how clear your code is. Excessively long code will be penalized: don't repeat code in multiple places. Your code should be well documented with Javadoc or AspectJdoc and sensibly indented so it is easy to read.

Be sure your name is in the comments in your code.

REFERENCES

[1] Joe Gradecki, Mastering AspectJ: Aspect-Oriented Programming in Java, Wiley, 2003. Free ebook through UTEP library.

[2] Martina Seidl, et al., UML@Classroom: An Introduction to Object-Oriented Modeling, Springer, 2015. Free ebook through UTEP library.

[3] Holger Gast, How to Use Objects, Addison-Wesley, 2016. Sections 9.1 and 9.2. Ebook available from UTEP library.