**Project Deliverable 2**

**Personal Software Process (PSP0 and JavaDoc)**

**Points: 50**

**----------------------------------------------------------------------------------------------------**

**Instructions**:

This assignment has to be completed by each student individually. NO COLLABORATION IS ALLOWED.

Submit the following: YourASURiteID-ProjectDeliverable2.zip This compressed folder should contain the following files:

1. Folder called core containing Connect4.java (Game Logic Module)
2. Folder called ui containing Connect4TextConsole.java (Console-based UI to test the game)
3. Folder called docs with JavaDoc documentation files (index.html and all other supporting files such as .css and .js files generated by the tool). Submit the entire folder.
4. ProjectDeliverable2.docx (or pdf) with Completed TimeLog, Design form, DefectLog, and ProjectSummary provided at the end of this assignment description.
5. Make sure to provide responses to reflection questions listed in ProjectDeliverable2 file (this document).
6. A few screen shots showing test results of your working game and answers to reflection questions written inline in this document
7. Readme file (optional: submit if you have any special instructions for testing)

**----------------------------------------------------------------------------------------------------**

**Connect4 Game**:

Connect4 is a 2-player turn-based game played on a vertical board that has seven hollow columns and six rows. Each column has a hole in the upper part of the board, where pieces are introduced. There is a window for every square, so that pieces can be seen from both sides. In short, it´s a vertical board with 42 windows distributed in 6 rows and 7 columns. Both players have a set of 21 thin pieces (like coins); each of them uses a different color. The board is empty at the start of the game. The aim for both players is to make a straight line of four own pieces; the line can be vertical, horizontal or diagonal.

Reference: <https://en.wikipedia.org/wiki/Connect_Four>

**----------------------------------------------------------------------------------------------------**

**Program Requirements**:

Implement a Java-based Connect4 game to be hosted in the ARENA Game system in the future.

In this first version build it as a simple console-based game played by 2 players – Player X and Player O. Ensure that following:

* Make use of good Object-Oriented design
* Provide documentation using Javadoc and appropriate comments in your code.
* Generate HTML documentation using Javadoc tool
* Create 2 packages core and ui.
* Create a separate class for the game logic called Connect4.java and place it in core package
* Create a separate class for the text-based UI called Connect4TextConsole.java and place it in ui package

Create a simple console-based UI as shown in the figures below.

| | | | | | | |

| | | | | | | |

| | | | | | | |

| | | | | | | |

| | | | | | | |

| | | | | | | |

Begin Game.

PlayerX – your turn. Choose a column number from 1-7.

4

| | | | | | | |

| | | | | | | |

| | | | | | | |

| | | | | | | |

| | | | | | | |

| | | |X| | | |

PlayerO – your turn. Choose a column number from 1-7.

1

| | | | | | | |

| | | | | | | |

| | | | | | | |

| | | | | | | |

| | | | | | | |

|O| | |X| | | |

.

.

.

| | | | | | | |

| | | | | | | |

| | | | | |O|X|

| | | |O|O|X|X|

| | | |O|X|X|X|

|O| | |X|X|O|O|

Player X Won the Game

* When the game starts, indicate that it is PlayerX’s turn. Ask the player to choose column number from 1-7.
* Check if the move is valid. If valid move, then show the state of the grid by placing an X at the bottom-most empty row of the specified column. Next check for “WIN” state (i.e., if playerX has an X in four consecutive horizontal, vertical, or diagonal cells of the grid). If it is a “WIN” state inform that PlayerX is the winner and close the game. If not continue the game.
* Indicate that it is PlayerO’s turn. Ask the player to choose column number from 1-7. Continue the game till one of them wins or game results in a draw.

**----------------------------------------------------------------------------------------------------**

**Personal Process:**

Follow a good personal process for implementing this game.

* Please use the time log (provided at the end of this document) to keep track of time spent in each phase of development.
* Please use the defect log (provided at the end of this document) to keep track of defects found and fixed in each phase of development.
* When you are done implementing and testing your program, complete the Project Summary form to summarize your effort and defects. Also answer the reflection questions listed below in Post-mortem phase.

Follow these steps in developing this game:

1. Plan – understand the program specification and get any clarifications needed
2. Design – create a design in the form of a flow chart, break up of classes and methods, class diagram, pseudocode. Provide this design in the PSP design form provided later in the document. Keep track of time spent in this phase and log. Also keep track of any defects found and log them.
3. Code – implement the program. Keep track of time spent in this phase and log. Also keep track of any defects found and log them.
4. Test – Test your program thoroughly and fix bugs found. Keep track of time spent in this phase and log. Also keep track of any defects found and log them.
5. Post Mortem – **Complete the project summary form and answer the following questions**.
6. In which Phase did you spend most of your effort? (look at the time spent in different phases of this assignment to answer this question)
   1. As expected, I spent most of my time in the **code** phase. This is two-fold. One, it’s hard to accurately design a 10,000ft view of a project until I get into the coding aspect. Possibly, this is because I’m a new programmer and two, defects were introduced once coding started, which needed to be fixed.
7. In which Phase did you introduce most number of defects?
   1. **Code** phase. Isn’t this where most defects are introduced?
8. Did you find it useful to follow a systematic process and track your effort and defects?
   1. At this stage of my learning, no. It’s good to see where I am introducing defects, but I do not have the breadth and depth to preemptively know where a defect will be introduced. Debugging and testing are my go-to tools.

**--------------------------------------------------------------------------------------------------**

**Grading Rubric**:

Working game – 20 points

Javadoc Documentation – 5 points

Test Results and Postmortem reflection question responses – 5 points

PSP process – 20 points (5 points each for Time log, Defect log, Design form, Project Summary)

**--------------------------------------------------------------------------------------------------**

PSP Time Recording Log

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Date** | **Start** | **Stop** | **Interruption Time** | **Delta**  **Time** | **Phase** | **Comments** |
| 5/25/2020 | 8:11am | 8:55am |  | 44 min | Plan |  |
| 5/25/2020 | 9:31am | 10:05am |  | 34 min | Plan |  |
| 5/25/2020 | 1:41pm | 3:47pm |  | 126 min | Plan |  |
| 5/26/2020 | 6:14am | 8:12am | 12 min | 106 min | Design | Interruption: Wife went to work, made lunch for her |
| 5/26/2020 | 9:33am | 11:12am |  | 99 min | Design |  |
| 5/26/2020 | 12:02pm | 2:45pm |  | 163 min | Design |  |
| 5/26/2020 | 3:55pm | 5:05pm |  | 70 min | Design |  |
| 5/26/2020 | 7:45pm | 10:15pm | 9 min | 141 min | Code | Interruption: Phone call |
| 5/27/2020 | 6:23am | 9:12am |  | 169 min | Code |  |
| 5/27/2020 | 9:33am | 10:15am | 4 min | 38 min | Code | Interruption: Getting coffee |
| 5/27/2020 | 11:03am | 2:55pm |  | 232 min | Code |  |
| 5/27/2020 | 3:09pm | 3:55pm |  | 46 min | Test |  |
| 5/27/2020 | 4:05pm | 4:23pm |  | 18 min | Test |  |
| 5/27/2020 | 4:24pm | 6:09 pm |  | 105 min | Code |  |
| 5/27/2020 | 6:23 pm | 6:55 pm | 8 min | 24 min | Code | Interruption: Wife went to work |
| 5/27/2020 | 7:01pm | 8:25pm |  | 84 min | Code |  |
| 5/27/2020 | 8:25pm | 8:35pm |  | 10 min | Test |  |
| 5/27/2020 | 8:45pm | 9:00pm |  | 15 min | Test |  |
| 5/27/2020 | 9:00pm | 9:48pm | 11 min | 37 min | Code | Interruption: Bathroom break |
| 5/28/2020 | 7:15am | 10:45am |  | 210 min | Code |  |
| 5/28/2020 | 11:47am | 3:45pm |  | 13 min | Code |  |
| 5/28/2020 | 3:45pm | 3:58pm |  | 13 min | Test |  |
| 5/29/2020 | 4:23am | 6:30am | 34 min | 93 min | Code | Interruption: Wife went to work, made lunch for her |
| 5/29/2020 | 6:33am | 7:21am |  | 48 min | Code |  |
| 5/29/2020 | 7:36am | 8:50am | 3 min | 71 min | Code | Interruption: Text |
| 5/29/2020 | 8:50am | 9:03am |  | 13 min | Test |  |
| 5/29/2020 | 10:45am | 11:13am | 16 min | 12 min | Test | Interruption: Phone call and text |
| 5/29/2020 | 1:15pm | 1:23pm |  | 8 min | Test |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |

* **Interruption time**: Record any interruption time that was not spent on the task. Write the reason for the interruption in the "Comment" column. If you have several interruptions, record them with plus signs (to remind you to total them).
* **Delta Time**: Enter the clock time you spent on the task, less the interrupt time.
* **Phase**: Enter the name or other designation of the programming phase being worked on. Example: Design or Code.
* **Comments**: Enter any other pertinent comments that might later remind you of any details or specifics regarding this activity.

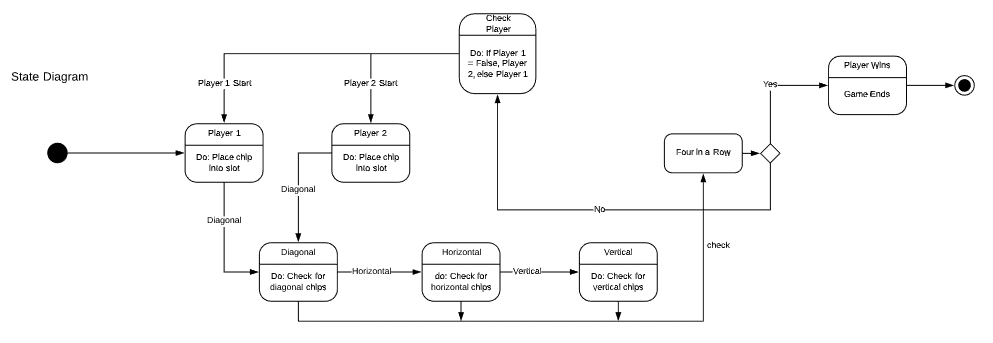
**PSP Design Form**

*Use this form to record whatever you do during the design phase of development. Include notes, class diagrams, flowcharts, formal design notation, or anything else you consider to be part of designing a solution that happens BEFORE you write program source code. Attach additional pages if necessary.*

Notes:  
 A state diagram was already developed for Connect4. This state diagram was re-examined for the design of module 2. I found that it correctly corresponded to the development of code.

Dr. Bansal asked that we containerize our projects in two packages; UI and Core. I was unfamiliar with packages and referenced StackOverflow.com:  
<https://stackoverflow.com/questions/3226282/are-there-best-practices-for-java-package-organization>

Previous State Diagram was used extensively when coding.



PSP Defect Recording Log

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Sl. No.** | **Date** | **Defect Type** | **Defect Inject Phase** | **Defect Removal Phase** | **Fix Time** | **Fix Ref** | **Description** |
| 0001 | 5/26/2020 | 100 | Design | Design | 11 min | N/A | Implementation of appropriate methods |
| 0002 | 5/26/2020 | 80 | Code | Code | 8 min | N/A | Incorrect columns resulting in error |
| 0003 | 5/27/2020 | 80 | Code | Code | 13 min | N/A | Initializing array missi |
| 0004 | 5/27/2020 | 80 | Code | Code | 12 min | N/A | No method to check if user places < 1 or > 7 |
| 0005 | 5/27/2020 | 60 | Test | Test | 8 min | N/A | Scanner class not instantiated correctly |
| 0006 | 5/27/2020 | 80 | Code | Code | 22 min | N/A | Incorrect code for diagonal check |
| 0007 | 5/28/2020 | 80 | Code | Code | 13 min | N/A | Method missing for valid move |
| 0008 | 5/28/2020 | 80 | Code | Code | 31 min | N/A | Index out of bounds, row error |
| 0009 | 5/29/2020 | 80 | Test | Test | 4 min | N/A | Splitting of classes not recognized |
| 0010 | 5/29/2020 | 30 | Test | Test | 3 min | N/A | Understanding the use of packages |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |

**Instructions**

* **Defect Type**: Use your best judgment in selecting which defect type applies from list provided below.
* **Defect Inject Phase**: Enter the phase when this defect was injected using your best judgment.
* **Defect Removal Phase**: Enter the phase during which you fixed the defect.
* **Fix Time**: Enter the time that you took to find and fix the defect.
* **Fix Ref**: If you or someone else injected this defect while fixing another defect, record the number of the improperly fixed defect. If you cannot identify the defect number, enter an X. If it is not related to any other defect, enter n/a.
* **Description**: Write a succinct description of the defect that is clear enough to later remind you about the error and help you to remember why you made it.

**PSP Defect Type Standard**

|  |  |  |
| --- | --- | --- |
| **Type Number** | **Type Name** | **Description** |
| 10 | Documentation | Comments, messages |
| 20 | Syntax | Spelling, punctuation, typos, instruction formats |
| 30 | Build, Package | Change management, library, version control |
| 40 | Assignment | Declaration, duplicate names, scope, limits |
| 50 | Interface | Procedure calls and references, I/O, user formats |
| 60 | Checking | Error messages, inadequate checks |
| 70 | Data | Structure, content |
| 80 | Function | Logic, pointers, loops, recursion, computation, function defects |
| 90 | System | Configuration, timing, memory |
| 100 | Environment | Design, compile, test, or other support system problems |

**PSP0 Project Summary**

|  |  |  |
| --- | --- | --- |
| **Time in Phase (minutes)** | **Actual Time**  **(in minutes)** | **% of Total** |
| Planning | 204 | 9.99% |
| Design | 438 | 21.44% |
| Code | 1265 | 61.94% |
| Test | 135 | 6.61% |
| Postmortem | 0 | 0% |
| TOTAL | 2042 |  |

|  |  |  |
| --- | --- | --- |
| **Defects Injected** | **Actual Number of Defects** | **% of Total** |
| Planning | 0 | 0% |
| Design | 8 | 16.32% |
| Code | 22 | 44.89% |
| Test | 19 | 38.77% |
| Postmortem | 0 | 0% |
| TOTAL | 49 |  |

SUMMARY

|  |  |
| --- | --- |
|  | **Actual** |
| Program Size (Lines of Code - LOC) | 296 |
| Productivity (calculated) LOC/Hour | 8.697 |
| Defect Rate (calculated) Defects/KLOC | 165.54 |

* LOC is lines of Code
* KLOC is Kilo lines of code (i.e. 1000 lines)
* 