

## Project Deliverable 3

### Personal Software Process (PSP1, JavaDoc, Exception Handling)

**Points: 50**

---

#### Instructions:

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

Submit the following: YourASURiteID-ProjectDeliverable3.zip This compressed folder should contain the following files:

1. Connect4.java (Game Logic Module)
  2. Connect4TextConsole.java (Console-based UI to test the game)
  3. Connect4ComputerPlayer.java (logic to play against computer; generate computer moves)
  4. JavaDoc documentation files (index.html and all other supporting files such as .css and .js files generated by the tool). Submit the entire folder.
  5. Completed Time Log, Estimation worksheet, Design form, Defect Log, and Project Summary provided at the end of this assignment description
  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](https://en.wikipedia.org/wiki/Connect_Four)

---

#### Program Requirements:

To the previously developed Java-based Connect4 game, add a module to “play against the computer”. Create a separate class called Connect4ComputerPlayer.java in the core package that generates the moves for the computer player. The logic to automatically generate computer moves does NOT have to be sophisticated AI algorithm. A naïve algorithm to generate the moves is sufficient for this assignment.

- Continue to make use of good Object-Oriented design
- Provide documentation using Javadoc and appropriate comments in your code.

- Generate HTML documentation using Javadoc tool
- Make sure you provide appropriate Exception Handling throughout the program (in the previously created classes as well)

Sample UI as shown in the figures below.

```
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
```

Begin Game. Enter 'P' if you want to play against another player; enter 'C' to play against computer.

>>C

Start game against computer.

It is your turn. Choose a column number from 1-7.

4

```
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | |X| | | |
```

and so on..

---

### Personal Process:

Follow a good personal process for implementing this game. You will be using PSP1 in this assignment. So, in addition to tracking your effort and defects you will have to estimate the effort and defects for the “play against computer” module as well as adding exception handling to previously created classes.

- 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.
  - estimate the time you are expecting to spend on the new module(s) to be added.
  - estimate the size of the program (only for new code that you will be adding)
  - enter this information in the Estimation columns of the Project Summary form. Use your best guess based on your previous programming experience. You will not be penalized for not having an estimate that is close to the actual. It takes practice to get better at estimation.
  - use the provided estimating worksheet.
2. **Design** – create a design (for the new modules being added) 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 actual columns of the project summary form and answer the following questions.
  - i. How good was your time estimate for various phases of software development?  
My estimations were lower than my actual times. My worst estimation ended up taking twice as long as my estimation.
  - ii. How good was your program size estimate, i.e., was it close to actual?  
My size estimate was about 3 times larger than my actual. I think I originally was going to have a more complex computer player.
  - iii. In which phase did you introduce most number of defects?  
I introduced the most errors in the code phase. I pretty much had the same type of errors as I did in PSP0.

---

#### **Grading Rubric:**

Working game – 15 points

Javadoc Documentation – 5 points

Exception Handling – 5 points

Test Results and Postmortem reflection question responses – 5 points

-----

## PSP Time Recording Log

[illegible]

- **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.

### **PSP1 Informal Size Estimating Procedure**

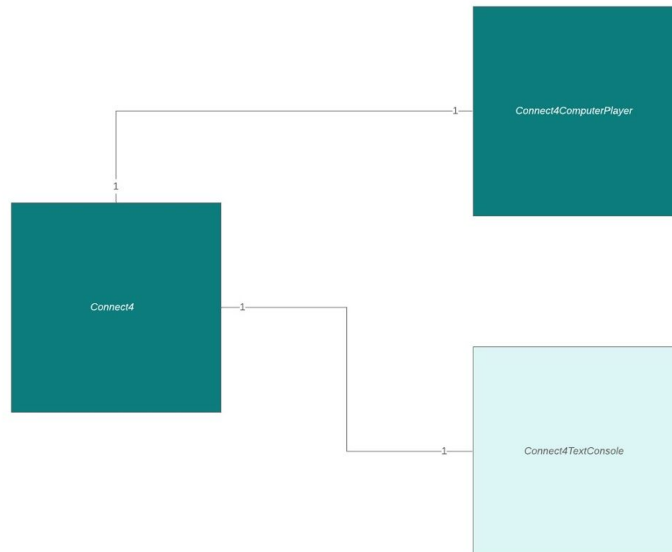
1. Study the requirements.
2. Sketch out a crude design.
3. Decompose the design into “estimatable” chunks.
4. Make a size estimate for each chunk, using a combination of:
  - \* visualization.
  - \* recollection of similar chunks that you’ve previously written
  - \* intuition.
5. Add the sizes of the individual chunks to get a total.

## **Estimating Worksheet**

1. Conceptual Design (sketch your high-level design here)

## The context of Connect4

Marcus Miller | March 31, 2019



## 2. Module Estimates

Module description	Estimated Size
--------------------	----------------

Plan	50 min
Design	60 min
Code	90 min
Test	30 min
Postmortem	20 min

Total Estimated Size: \_\_250\_\_

### PSP1 Project Summary

Time in Phase (minutes)	Estimated	Actual	To Date	To Date %
Planning	50	38	70	8.67%
Design	60	68	208	25.77%
Code	90	134	314	38.91%
Test	30	50	160	19.83%
Postmortem	20	40	55	6.82%
TOTAL	250	330	807	100%

Defects Injected	Estimated	Actual	To Date	To Date %
Planning	***	0	0	0
Design	***	1	5	50
Code	***	3	5	50
Test	***	0	0	0
Postmortem	***	0	0	0
TOTAL	***	4	10	100

### SUMMARY

	Estimated	Actual	To Date
Program Size (LOC)	350	104	393
LOC/Hour	0.65	18.91	29.22
Defects/KLOC	20.7	38.46	25.45

- LOC is lines of Code

- KLOC is Kilo lines of code (i.e. 1000 lines)

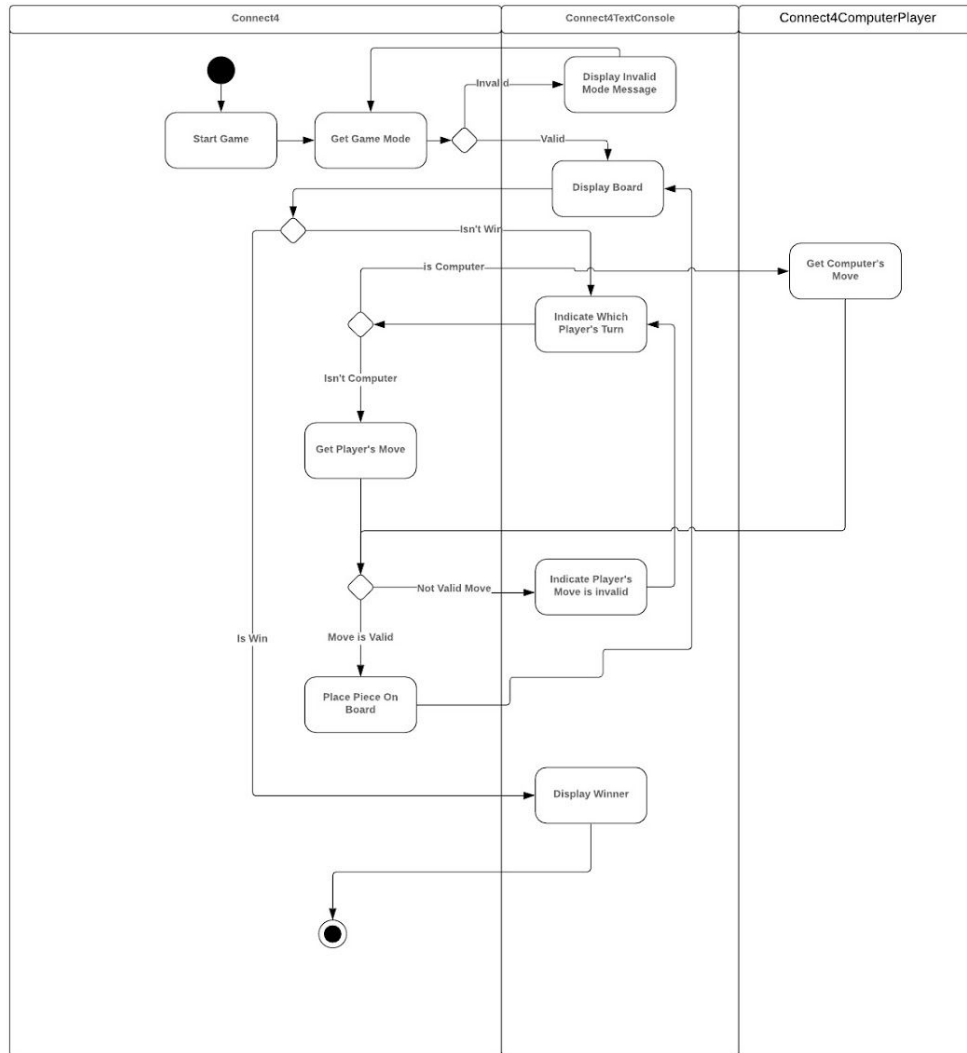
## **PSP Design Form**

*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.*



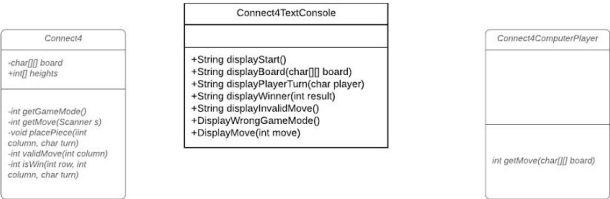
# Connect4 Activity Diagram

Marcus Miller | March 31, 2019



Connect4 Class Diagram

Marcus Miller | March 31, 2019



## PSP Defect Recording Log

[illegible]

## 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