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. Folder called core containing the following:
    a. Connect4.java (Game Logic Module)
    b. Connect4ComputerPlayer.java (logic to play against computer; generate computer moves)
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. ProjectDeliverable3.docx (or pdf) with Completed Time Log, Estimation worksheet, Design form, Defect Log, and Project Summary provided at the end of this assignment description
5. Make sure to provide responses to reflection questions listed in ProjectDeliverable3 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:
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?
        a. I thought it was going to take me an estimated 860 minutes due to how long the first assignment was. This was entirely inaccurate. It took less than one hour, start-to-finish.

    ii. How good was your program size estimate, i.e., was it close to actual?
        a. I was off by 34 lines of code. I have a very hard time estimating the actual lines of code at this stage of my studies.

    iii. In which phase did you introduce most number of defects?
        a. This might not be believable, but I did not introduce any defects. It was a very simple expansion of what was already built and took only a few more methods to complete.

Grading Rubric:
Working game – 15 points

Javadoc Documentation – 5 points
Exception Handling – 5 points
Test Results and Postmortem reflection question responses – 5 points
PSP process – 20 points (4 points each for Time log, Defect log, Estimating Worksheet, Design form, Project Summary)

PSP Time Recording Log

Date	Start	Stop	Interruption Time	Delta Time (m/s)	Phase	Comments
06/03/2020	04:37:13 PM	04:47:06 PM	38 sec	00:03:33	Plan	Interruption: Text
06/03/2020	04:47:14 PM	04:54:03 P		00:06:49	Design	
06/03/2020	04:54:21 PM	05:04:47 PM		00:10:26	Code	
06/03/2020	05:07:05 PM	05:37:55 PM		00:30:50	Code	
06/03/2020	05:38:02 PM	05:40:42 PM		00:02:40	Test	
06/03/2020	05:41:03 PM	05:42:09 PM		00:01:06	Test	
06/04/2020	05:19:23 PM	05:30:00 PM		00:10:37	Design	
06/04/2020	05:31:40 PM	05:37:11 PM		00:05:31	Post Mortem	

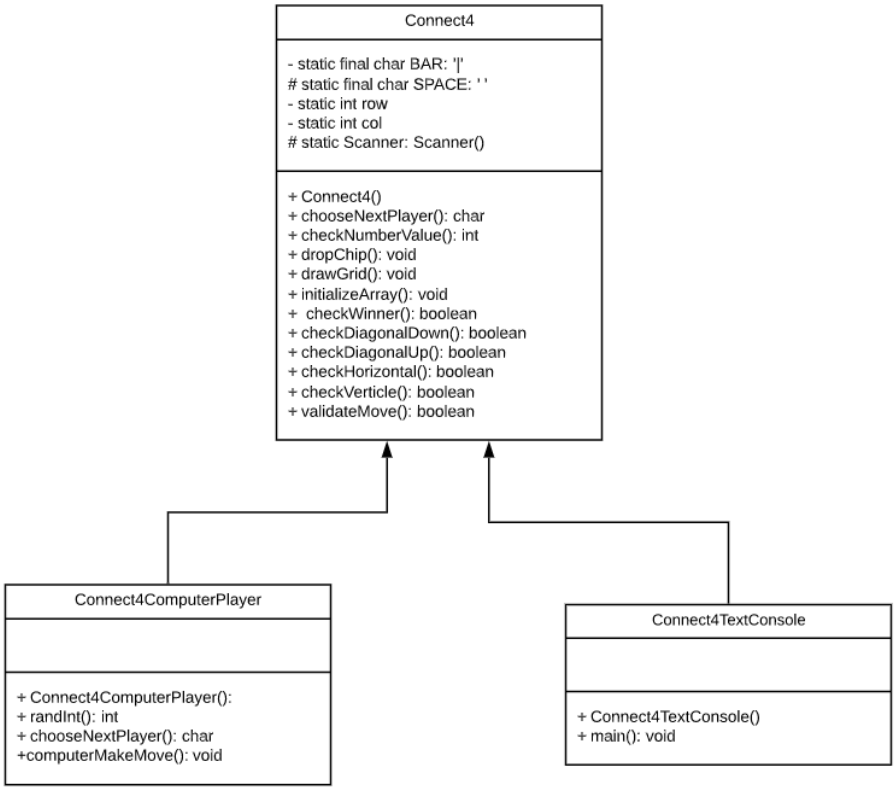
- **Interruption time**: Record any interruption time that was not spent on the task. Write the reason for the interruption in the "Comment" column. If you have several interruptions, record them with plus signs (to remind you to total them).
- **Delta Time**: Enter the clock time you spent on the task, 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 youíve 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)



2. Module Estimates

Module description	Estimated Size
Connect4ComputerPlayer	75

Total Estimated Size: ____75_____

PSP1 Project Summary

Time in Phase (minutes)	Estimated	Actual (m/s)	To Date	To Date %
Planning	20	00:03:33	207	9.85
Design	30	00:06:49	444	21.14
Code	500	00:41:16	1306	62.19
Test	300	00:03:46	138	6.57
Postmortem	10	00:05:31	5	0.23
TOTAL	860	58 min	2100	100

Defects Injected	Estimated	Actual	To Date	To Date %
Planning	***	0	0	0
Design	***	0	8	16.32
Code	***	0	22	44.89
Test	***	0	19	38.77
Postmortem	***	0	0	0
TOTAL	***			100

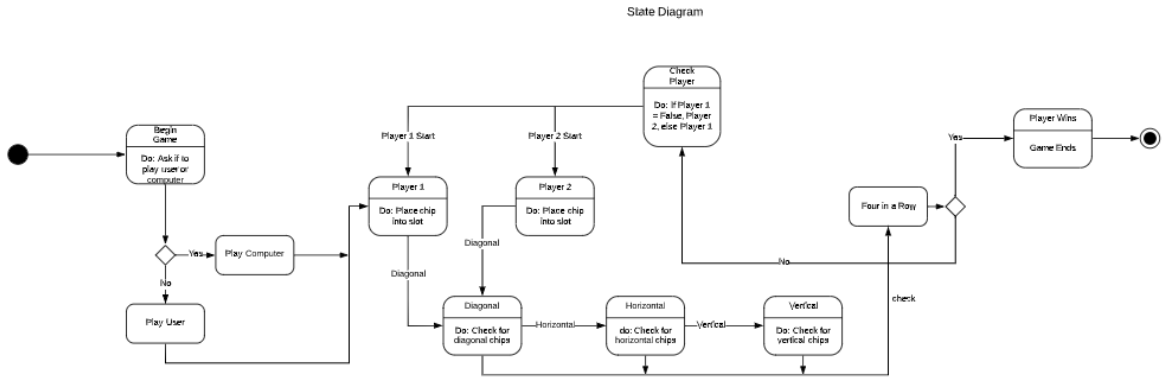
SUMMARY

	Estimated	Actual	To Date
Program Size (LOC)	300	109	405
LOC/Hour	8	109	117
Defects/KLOC	100	0	165.54

- 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.



PSP Defect Recording Log

Sl. No.	Date	Defect Type	Defect Inject Phase	Defect Removal Phase	Fix Time	Fix Ref	Description

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

```
"C:\Program Files\Java\jdk-14.0.1\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\Int
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
Begin Game
Enter 'P' if you want to play against another player; enter 'C' to play against computer
```

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

```
Start game against computer.
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
|X| | | | |O| |
Start game against computer.
It is your turn. Choose a column number from 1 - 7
```

```
5
| | | | | | | |
|O| | |O| | | |
|O|O|O|X|X| | |
|X|X|X|O|X|O| |
|X|O|O|O|X|X|X|
|X|X|X|O|X|O|O|
Player X has won!
```