**Project Deliverable 5**

**Personal Software Process & Quality**

**(PSP2, Network Programming)**

**Points: 50**

**Rock Paper Scissors Game**:

Reference: https://en.wikipedia.org/wiki/Rock\_paper\_scissors

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

**Postmortem Questions**

1. How good was your time and defect estimate for various phases of software development?

My time and defect estimates were approximately the same as the actual time and actual defect. The time estimate was 900 minutes, but the actual time spent was 960 minutes, which was so close. I estimated 40 defects and the actual number of defects injected is 49 defects.

1. How good was your program size estimate, i.e., was it close to actual?

The actual size was close to the estimate size made. The estimate size was 500 LOC while the actual size was 410 LOC

1. How many issues did you find in your code during code review?

4 issues

The functionality describe in the specification is not fully implemented

Some parameters were not presented in the correct order

Client sockets having some issue send moves

The token cell not being displayed on the stage yet

PSP Time Recording Log

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Date** | **Start** | **Stop** | **Interruption Time** | **Delta**  **Time** | **Phase** | **Comments** |
| 11/10/21 | 6:30Pm | 7:30Pm | 0 | 1h | Plan | Understanding the game logic |
| 11/10/21 | 8Pm | 11Pm | 1hr | 2h | Design | Design a networked RPS game |
| 11/11/21 | 4:30Pm | 6:30Pm | 0 | 2h | Design | How will the RPSServer class and RPSClient class be communicating to each other |
| 11/12/21 | 5Pm | 7Pm | 0 | 2h | Code | Implement the functionality of the host server |
| 11/12/21 | 9:30Pm | 11:30Pm | 0 | 2h | code | Implement the functionality of the client |
| 11/15/21 | 6:30Pm | 9:30Pm | 30min | 2h30min | Code Review | Ensuring the specification required has been handed appropriately |
| 11/15/21 | 4:30Pm | 6:30Pm | 0 | 2h | Test | Figure out how to run two clients’ instances at the same time to test the game |
| 11/16/21 | 3:30Pm | 5:30Pm | 0 | 2h | Test | Players turn handled appropriately |
| 11/16/21 | 7:30Pm | 9:30 | 0 | 2h | Postmortem | Filing the logs and complete the code review using PSP2 |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |

**Estimating Worksheet**

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

Text

Description automatically generated

1. Module Estimates

|  |  |
| --- | --- |
| **Module description** | **Estimated Size** |
| RPSServer GUI | 200 |
| RPSClient GUI | 300 |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

Total Estimated Size: \_\_500\_\_\_\_\_\_\_\_\_\_

**PSP2 Project Summary**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Time in Phase (minutes)** | **Estimated** | **Actual** | **To Date** | **To Date %** |
| Planning | 120 min | 60min | 60min | 6.3% |
| Design | 180 min | 240min | 240min | 25% |
| Code | 300 min | 360min | 360min | 38% |
| Code Review | 120 min | 90min | 90min | 9.4% |
| Test | 120 min | 120min | 120min | 13% |
| Postmortem | 60 min | 90min | 90min | 9.4% |
| TOTAL | 900 | 960min | 960min | 100% |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Defects Injected** | **Estimated** | **Actual** | **To Date** | **To Date %** |
| Planning | 5 | 6 | 6 | 12 |
| Design | 8 | 10 | 10 | 20 |
| Code | 12 | 20 | 20 | 41 |
| Code Review | 5 | 4 | 4 | 8.2 |
| Test | 10 | 8 | 8 | 16 |
| Postmortem | 0 | 1 | 1 | 2 |
| TOTAL | 40 | 49 | 49 | 100 |

**SUMMARY**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Estimated** | **Actual** | **To Date** |
| Program Size (LOC) | 500 | 410 | 410 |
| LOC/Hour | 33 | 27 | 27 |
| Defects/KLOC | .04 | .049 | .049 |

* 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.*

Server:

Start the server

Wait for 2 players to join

When both join, read selection of Player 1 (R, P, or S)

Read selection of player 2 (R, P, or S)

Compare the entries and figure out who won/drew

Notify the players

Ask for continuing again

Client: (2 instances running)

Join the Server

On server’s notification, enter my choice (R, P, or S)

Find what other player entered and results from the server

Enter choice for continuation

Diagram

Description automatically generated

PSP Defect Recording Log

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Sl. No.** | **Date** | **Defect Type** | **Defect Inject Phase** | **Defect Removal Phase** | **Fix Time** | **Fix Ref** | **Description** |
| 1 | 11/10/21 | 70 | Plan | Plan | 20min |  | Wondering how to set Rock, Paper, Scissors components in simple way |
| 2 | 11/10/21 | 30 | Plan | Design | 15min |  | First had created 3 buttons, then thought of using Cell objects as Panes |
| 3 | 11/11/21 | 100 | Design | Code | 60min |  | Could not run 2 instances of the client in any way. Tried using command line, set paths. |
| 4 | 11/12/21 | 50 | Design | Code |  |  |  |
| 5 | 11/12/21 | 40 | Code | Code | 20min |  | Faced some variable scope issues here and there |
| 6 | 11/15/21 | 20 | Code | Test | 10min |  | Misspelling some variable and function names |
| 7 | 11/15/21 | 80 | Code | Test | 20min |  | SendMove function not performing as required due to inappropriate initialization |
| 8 | 11/16/21 | 60 | Code | Code review | 25min |  | Randomize tokens using the Random function kept on returning the wrong token |
| 9 | 11/16/21 | 70 | Plan | Test |  |  |  |
| 10 | 11/16/21 | 20 | Test | Code review |  |  |  |
|  |  |  |  |  |  |  |  |

**Code Review Checklist – Java**

1. Specification / Design

[The token cell is not displayed on the stage for some reason, haven’t figured how why] Is the functionality described in the specification fully implemented by the code?   
[No] Is there any excess functionality in the code but not described in the specification?

2. Initialization and Declarations

[yes] Are all local and global variables initialized before use?   
[yes] Are variables and class members of the correct type and appropriate mode   
[yes] Are variables declared in the proper scope?   
[yes] Is a constructor called when a new object is desired?   
[yes] Are all needed import statements included?

[yes] Names are simple and if possible short

[No] There are no usages of ‘magic numbers’ (i.e, hard-coded values)

3. General

[yes] Code is easy to understand

[yes] Variable and Methods names are spelt correctly

[No dead code] There is no dead code (i.e., code inaccessible at Runtime)

[yes] Code is not repeated or duplicated

[no empty code] No empty blocks of code

4. Method Calls   
[yes] Are parameters presented in the correct order?   
[yes] Are parameters of the proper type for the method being called?  
[yes] Is the correct method being called, or should it be a different method with a similar name?   
[yes] Are method return values used properly? Cast to the needed type

5. Arrays/Data structures   
[] Are there any off-by-one errors in array indexing?   
[] Can array indexes ever go out-of-bounds?   
[] Is a constructor called when a new array item is desired?

[] Ideal data structures are used

[] Collections are initialized with a specific estimated capacity

6. Object   
[ ] Are all objects (including Strings)  compared with "equals" and not "=="?

[ ] No object exists longer than necessary

[ ] Files/Sockets and other resources if used are properly closed even when an exception occurs in using them

7. Output Format   
[no] Are there any spelling or grammatical errors in displayed output?   
[ ] Is the output formatted correctly in terms of line stepping and spacing?

8. Computation, Comparisons and Assignments   
[] Check order of computation/evaluation, operator precedence and parenthesizing   
[] Can the denominator of a division ever be zero?   
[] Is integer arithmetic, especially division, ever used inappropriately, causing unexpected truncation/rounding?   
[] Check each condition to be sure the proper relational and logical operators are used.   
[] If the test is an error-check, can the error condition actually be legitimate in some cases?   
[] Does the code rely on any implicit type conversions?

9. Exceptions

[yes] Are all relevant exceptions caught?   
[yes] Is the appropriate action taken for each catch block?

[yes] Are all appropriate exceptions thrown?

[yes] Are Catch clauses are fine-grained and catch specific exceptions?

10. Flow of Control

[ No switch case used] In a switch statement is every case terminated by break or return?   
[ No switch case used] Do all switch statements have a default branch?  
[ yes] Check that nested if statements don't have “dangling else” problems.   
[ yes] Are all loops correctly formed, with the appropriate initialization, increment and termination expressions?   
[ yes] Are open-close parentheses and brace pairs properly situated and matched?

11. Files

[yes] Are all files properly declared and opened?   
[ yes] Are all files closed properly, even in the case of an error?   
[yes] Are EOF conditions detected and handled correctly?   
[ yes] Are all file exceptions caught?

12. Documentation

[yes] All methods are commented in clear language.

[yes] Comments exist and describe rationale or reasons for decisions in code

[yes] All public methods/interfaces/contracts are commented describing usage

[yes] All edge cases are described in comments

[yes] All unusual behavior or edge case handling is commented

[yes] Data structures and units of measurement are explained