

School of Computing

Graduate Diploma in Information and Communication Technologies

Bachelor of Information and Communication Technologies

BCPR280 – Software Engineering 2

Assignment 1

Semester two 2018

Due date: Friday 22 September 2018

Time: 5:00 pm

Student Name/ID

Ara and its faculty members reserve the right to use electronic means to detect and help prevent plagiarism. Students agree that when submitting this assignment, it may be subject to submission for textual similarity review to Turnitin.com.

Submissions received late will be subject to a penalty of 10% of the student's mark per working day.

This assignment is worth 25% of the total marks for BCPR280.






This assessment has eleven (11) pages including the cover sheet.



Create a Single Page Application using **Vuejs** and **JavaScript ES2018** that meets the requirements of the problem set in either appendix one or appendix two.

The SPA needs to be able to do **all tasks listed in the either appendix one or appendix two**. Use the problem set in appendix one if you scored less than 50% on the practice test. Use the problem set in appendix two if you scored more than 80% on the practice test. Those who scored between 50 and 80 in the practice test can choose which problem set to use.

Marking schedule

-  1. A plan for the work for each task This must include:
 - a) The goal of each iteration.
 - b) The planned tasks in sequence **[planning, analysis, design, coding, testing]**
 - c) A time estimate for each task **[10 minute blocks]**, i.e., which date, when, to do what
 - d) A record of the actual time each task took, i.e., time log (Use tool: <https://www.toggl.com/>)(10 marks)
2. A **design level class diagram** for each task. (10 marks)
3. A plan for how the program feature you are working on will work. By the end of this assignment create **at least one of each** of the following design techniques:
 - UML dynamic diagram (activity, collaboration, state transition, sequence)
 - Use case
 - storyboard
 - Wireframe with tag types, classes/names/ids, attributes, and associated CSS values
 - Pseudocode
 - 'Planning A Complex Algorithm' worksheet
 -  • Spike solution¹.(10 marks)
-  4. Fill in **test planning section** of the test report in plain English for **all functions required** in PSP1 by investigating the files in **spec** folder provided (10 marks)
5. Works programs with all required features and proof it passes **StandardJS** validation (30 marks)
-  6. Run the test to fill in the actual result section in the testing report (10 marks)
7. At least 10 Error logs each recording at least 10 errors (10 marks)
-  8. At least 10 Process Improvement Proposals (10 marks)

¹ <http://www.extremeprogramming.org/rules/spike.html>;
<http://c2.com/cgi/wiki?SpikeSolution>

APPENDIX ONE

Use this problem set if you are not confident/competent at JavaScript.

1. Input a series of letters. End with a "rogue" of a full stop. Output the number of g's entered (there may not be any).
2. Input a "target" letter, followed by a series of letters. End with a "rogue" of a full stop. Output the number of times the target letter was entered in the series.
3. Input a series of numbers. End with a "rogue" of 999. Output the number of **positive numbers** in the series. Output the number of **negative numbers** in the series. Output the number of **zeros** in the series.
4. Input a series of integers. End with a "rogue" of 999. Output the number of times the "next" integer is twice the previous.
5. Input a series of integers. End with a "rogue" of 999. Output the message **"Series in sequence"** if the series is in ascending order or the message **"Series not in sequence"** if the series isn't. Note that if adjacent values are **equal**, the series is still in **ascending order**.

APPENDIX TWO

Use this problem set if you are confident/competent at JavaScript.

1. Write a program to play a number guessing game. The program shall generate a random number between 0 and 99. The USER inputs his/her guess, and the program shall response with "Try higher", "Try lower" or "You got it in n trials" if the guess is correct.
2. Write a program to play a number guessing game. The program shall generate a random number between 0 and 99. The USER inputs his/her guess, and the program shall response with "COLD" if the guess is more than 40 from the target number, "COOL" if the guess is within 20-39 of the target number, "WARM" if the guess is within 10-19 of the target number, "HOT" if the guess is within 1-9 of the target number or "You got it in n trials" if the guess is correct.
3. Write a program to play a number guessing game. The USER mentally selects a number between 0 and 99 and the computer tries to guess it. The computer outputs its guess, and the User response with "Try higher", "Try lower" or "correct". The computer should keep count of the number of guesses. The computer should complain if the USER has lied.
4. Write a program to play a number guessing game. The USER mentally selects a number between 0 and 99 and the computer tries to guess it. The computer outputs its guess, and the User response with "COLD" if the guess is more than 40 from the target number, "COOL" if the guess is within 20-39 of the target number, "WARM" if the guess is within 10-19 of the target number, "HOT" if the guess is within 1-9 of the target number or "correct". The computer should keep count of the number of guesses. The computer should complain if the USER has lied.

APPENDIX THREE

DEFECT CLASSIFICATION

<http://www.ipd.uka.de/PSP/Dokumente/DefTyp/defecttypes.html>
created: 1998-09-02, Lutz Prechelt
changed: 1999-01-14, Lutz Prechelt
RCS: \$Id\$

You can either use this standard as is, or adapt it to your needs.
If you adapt it, summarize the changes below.

\$log\$

This defect classification uses three dimensions:

1. The [injection phase](#): When was the defect produced?
2. The [defect type](#) (product-related): What was the structure of the defect itself (or: what sort of repair was required)?
3. The [defect reason](#) (process-related): Why was the defect introduced (or: what sort of mistake led to the defect)?

Defect injection phases

The injection phase of a defect describes *when a defect was introduced*. To be useful, this should not refer to the surface causal event of the defect ("When did I produce the part of the product that I now have to repair?"), but instead should refer to the deep causal event (root cause, RC) of the defect: "When was the event that subsequently made me produce the part of the product that I now have to repair?".

There are three cases:

- **Local:** The Root Cause can be in any one of the development phases of the program in the current time/defect log.
In this case I indicate that phase as the injection phase of the defect, for instance **ds** for design or **cd** for coding.
- **Regional:** The defect can be in some other software produced in the same project (or the same organisation), but not the one in the current time/defect log.
In this case I call the injection phase **pr** (for 'project') or **op** (for 'other project' of my own organisation). Note that the original author of the defective piece of software should be informed of the problem and should log it in the respective time/defect log -- even if s/he does not repair the defect.
- **External:** The defect is in some software I cannot control or repair. In this case I call the injection phase **ex** (for 'external').

Defect Types

This classification describes the *structure* of the defect:

- **IC: Interface Capability.**
The design of an interface is wrong, so that the interface does not provide the functionality that it must provide.
- **IS: Interface Specification.**
The specification of an interface is wrong, so that the parameters involved cannot transfer all of the information required for providing the intended functionality.
This is a less fundamental variant of IC: Only parameters need be added.
- **ID: Interface Description.**
The non-formal part of the description of an interface is incomplete, wrong, or misleading. This is typically diagnosed after an IU.
Note that the description of a variable or class attribute or data structure invariant is also an (internal) interface.
- **II: Interface Implementation.**
Something that I cannot influence does not work as it should.
This defect should never be used when I am the source of the defect. (In principle, this is a special case of ID.)
- **IU: Interface Use.**
An interface was used wrongly, i.e., in such a way as to violate the interface specification.
- **IV: Data Invariant.**
A special case of IU. The interface violation is: not maintaining the invariant of some variable or data structure.
Violating the meaning of a simple variable is a special case of this.
- **MD: Missing Design (of required functionality).**
A certain requirement is covered nowhere in the design.
This is stronger than IC, where the coverage is present, but incomplete.
- **MI: Missing Implementation (of planned functionality).**
A certain part of a design was not implemented.
If the part is small, MC, MA or WA may be more appropriate.
- **ME: Missed Errorhandling.**
An error case was not handled in the program (or not handled properly).
- **MA: Missing Assignment.**
A single variable was not initialized or updated.
Only one statement needs to be added.
- **MC: Missing Call.**
A single method call is missing.
Only one statement needs to be added.
- **WA: Wrong Algorithm.**
The entire logic in a method is wrong and cannot provide the desired functionality.
More than one statement needs to be added or changed.
- **WE: Wrong Expression.**
An expression (in an assignment or method call) computes the wrong value.
Only one expression needs to be changed.
- **WC: Wrong Condition.**
Special case of WE. A boolean expression was wrong.
Only one expression needs to be changed.
- **WN: Wrong Name.**
Special case of WE. Objects or their names were confused. The wrong method, attribute, or variable was used.
Only one name needs to be changed.
- **WT: Wrong Type.**
Two 'similar' types were confused.

IC, IS, ID, MD are typically related to design.

IU, IV, MI, MA, WE, WC, WN are typically related to implementation.

After some defect data have been collected one may adapt the above classification to one's own needs. Such changes should be upwards compatible so as not to render old data useless. This means that the only changes are typically *clarification* of the definition of a defect type and *extension* of the scheme to include

additional defect types. Old defect types may be marked obsolete but should not be removed from the standard. Still, as a rule, changes to the defect classification should be made only rarely.

Defect Reasons

This classification describes *why the defect was introduced*:

- **om:** Omission.
I forgot something that I knew I had to do.
- **ig:** Ignorance.
I forgot something, because I did not know I had to do it.
- **cm:** Commission.
I did something wrong, although I knew in principle how to do it right.
- **ty:** Typo.
I did something trivial wrong, although I knew exactly how to do it right.
- **kn:** Knowledge.
I did something wrong, because I lacked the general knowledge (i.e., the education) how to do it right.
- **in:** Information.
I did something wrong, because I lacked the specific knowledge how to do it right or had received misleading information about how to do it. This refers to a problem with communication.
- **ex:** External.
I did nothing wrong. The problem was somewhere else and the defect was introduced by some other person.

[Lutz Prechelt](#), prechelt@ira.uka.de, Last modified: Wed Apr 14 09:39:03 MET DST 1999

Student
Program
Instructor

Page 7

APPENDIC FIVE

Process Improvement Proposal (PIP)

Student	_____	Date	_____
Program	_____	Program #	_____
Instructor	_____	Language	_____

Problem Description

Briefly describe the problems that you encountered.

Proposal Description

Briefly describe the process improvements that you propose.

Other Notes and Comments

Note any other comments or observations that describe your experiences or improvement ideas.

APPENDIX SIX

Test Report Template

Student	_____	Date	_____
Program	_____	Program #	_____
Instructor	_____	Language	_____

Test Name/Number	_____
Test Objective	_____
Test Description	_____

Test Conditions	_____

Expected Results	_____

Actual Results	_____

Test Name/Number	_____
Test Objective	_____
Test Description	_____

Test Conditions	_____

Expected Results	_____

Actual Results	_____

Test Report Template Instructions

General	<ul style="list-style-type: none">- Expand this table or use multiple copies as needed.- Report all the tests that were successfully run.- Be as brief and concise as possible.
Header	<ul style="list-style-type: none">- Enter your name and the date.- Enter the program name and number.- Enter the instructor's name and the programming language you are using.
Test Name/Number	Uniquely identify each test for each program. <ul style="list-style-type: none">- the same tests with different data- the same data with different tests
Test Objective	<ul style="list-style-type: none">- Briefly describe the objective of the test.
Test Description	Describe each test's data and procedures in sufficient detail to facilitate its later use as a regression test.
Test Conditions	<ul style="list-style-type: none">- List any special configuration, timing, fix, or other conditions of the test. When multiple tests are run with different parameters or under varying conditions, separately list each.
Expected Results	<ul style="list-style-type: none">- List the results that the test should produce if it runs properly.
Actual Results	List the results that were actually produced.

Name _____

ITERATION PLAN

<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
--------------------------	--------------------------	--------------------------	--------------------------	--------------------------	--------------------------	--------------------------	--------------------------	--------------------------	--------------------------	--------------------------

- | | | | | | | | | | | |
|-----------------------|-----------------------|-----------------------|-----------------------|-----------------------|-----------------------|-----------------------|-----------------------|-----------------------|-----------------------|--|
| <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | goal |
| <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | planned tasks in sequence |
| <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | time estimate for each task [10 minute blocks] |
| <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | record of the actual time |

DESIGN LEVEL CLASS DIAGRAM

<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
--------------------------	--------------------------	--------------------------	--------------------------	--------------------------	--------------------------	--------------------------	--------------------------	--------------------------	--------------------------	--------------------------

PLAN FOR HOW FEATURE WORKS.

<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
--------------------------	--------------------------	--------------------------	--------------------------	--------------------------	--------------------------	--------------------------	--------------------------	--------------------------	--------------------------	--------------------------

NOTE: AT LEAST ONE of each of the following

- | | | | | | | |
|-----------------------|-----------------------|-----------------------|-----------------------|-----------------------|-----------------------|--|
| <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | UML dynamic diagram |
| <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | use case |
| <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | storyboard |
| <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | wireframe |
| <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | pseudocode |
| <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | 'Planning A Complex Algorithm' worksheet |
| <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | spike solution |

TEST PLANS

<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
--------------------------	--------------------------	--------------------------	--------------------------	--------------------------	--------------------------	--------------------------	--------------------------	--------------------------	--------------------------	--------------------------

WORKING PROGRAM WHICH PASSES STANDARDJS VALIDATION

<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

TEST RESULTS

<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
--------------------------	--------------------------	--------------------------	--------------------------	--------------------------	--------------------------	--------------------------	--------------------------	--------------------------	--------------------------	--------------------------

ERROR LOGS EACH RECORDING AT LEAST 10 ERRORS

<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
--------------------------	--------------------------	--------------------------	--------------------------	--------------------------	--------------------------	--------------------------	--------------------------	--------------------------	--------------------------	--------------------------

PIP

<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
--------------------------	--------------------------	--------------------------	--------------------------	--------------------------	--------------------------	--------------------------	--------------------------	--------------------------	--------------------------	--------------------------