Software Engineering with Objects and Components
Group Project: Deliverable 2
Marking Scheme

Part 1 - Team Information

Tutorial Group (T01-T06)	T06	**Tutor's Name**	Shahzad Asif
Tutorial Group's Name	iSolutions	**Team: O, P or D**	Plant
Team Members			
Ankit Sawa		Scott Hofman	
Andrei Manolache		Simon Zhong	
Callum Jamieson			

Part 2 - Individual Marking Scheme

Individual Marking Scheme					
	Estimated Contribution [tick one box only]				
Team Member	**U**	**S**	**G**	**E**	**Marks**
Ankit Sawa	☐	☐	☐	☒	97
Andrei Manolache	☐	☐	☐	☒	97
Callum Jamieson	☐	☐	☐	☒	97
Scott Hofman	☐	☐	☐	☒	97
Simon Zhong	☐	☐	☐	☒	97
	☐	☐	☐	☐	

U - Unsatisfactory: Some attempt but, contribution very weak due to poor skills, unsatisfactory attendance or other factors. **Note that an unsatisfactory evaluation will result in reduced marks.**

S - Satisfactory: Met all team requirements and contributed ordinary.

G - Good: Above average effort and contribution.

E - Excellent: A sustained and outstanding contribution to team effort.

Part 3 - Deliverable Marking Scheme

Deliverable Marking Scheme		
Deliverable Part	**Questions**	**Marks**
Prioritized Use Cases	**Q1.** Do your use cases capture functional system requirements? Do your use cases make good use of the features (e.g., use case relationships) of UML intended to support requirements gathering?	
	Our use case accurately reflects the behaviour of our plant system by using the required relationships responsibly. Functional requirements captured by our diagram demonstrate the complete nature of our system. We have implemented all of our prioritized use cases - we were able to capture all of our given scope.	[/ 5] 5
Refined Class Diagram	**Q2.** Does your class diagram make good use of the features (e.g., attributes, methods and constraints that are necessary) of UML intended to support detailed design?	
	The class diagram employs an exact description of system's structure through accurate usage of attributes, operators and relationships. This design has been improved upon since our previous version, and the changes help amplify this description.	[/ 5] 5
Change History	**Q3.** Does your change history record change rationale? Does your change history enable (requirements) traceability practice?	
	Our change history records change rationale. We provided documentation that describes the improvements made throughout our project's history and the reasoning behind it. On comparing Deliverable 1 and Deliverable 2, one can easily trace what changes have been made. Thus, requirements traceability can be implemented because we can trace a forward and backward path of our project's history.	[/ 5] 4
Sequence or Communication Diagrams	**Q4.** Are your sequence (communication) diagrams well-formed? Provide a brief assessment of how well your diagrams match the UML specification. Identify any instance of your diagrams deviating from the standard approach.	
	The sequence diagram is designed to show the sequential interactions between our objects. It exactly follows the defining parameters of the UML specifications by using boxes for objects, ordering the messages to display interaction and activation boxes to represent the performing processes.	[/ 5] 5
		continued on the next page

continued from previous page

Deliverable Part	Questions	Marks
	Q5. Do your sequence (communication) diagrams capture/realise your chosen Use Cases? Provide a brief assessment of the strengths and weaknesses of your sequence (communication) diagrams as a means of formalising the activity in your Use Cases.	
	Our sequence diagrams show the interaction and sequential order between the created entities. It uses the information from the use case to create a cohesive interaction. This models the flow of logic within our system. The sequence diagram fails to capture the repetition of placing multiple orders. This results from the sequence representing a single scenario - a weakness within our sequence diagram.	[/ **10**] 10
	Q6. Assess the extent to which your sequence (communication) diagrams are consistent with the class diagram.	
	The consistency is exact because it follows the class diagram without the use of modifications. Every class implemented within our system has been modelled as objects with respective lifelines capturing their focus. Again, our class diagram can reflect multiple orders - the sequence diagram cannot for the limitations indicated above.	[/ **10**] 10
Activity or Statechart Diagrams	**Q7.** Are your activity (statechart) diagrams well-formed? Provide a brief assessment of how well your diagrams match the UML specification. Identify any instance of your diagrams deviating from the standard approach.	
	The Activity diagram is constructed from a limited number of objects connected with arrows that represents the order in which activities happen. The defined notation for this type of diagram is used properly to capture the logic of the system in a algorithmic way. There are no deviations/extra features from the standard activity diagram approach.	[/ **5**] 5
	Q8. Do your activity (statechart) diagrams capture dynamic behaviours of your chosen/implemented classes? Provide a brief assessment of the strengths and weaknesses of your activity (statechart) diagrams as a means of formalising the dynamic aspects of your Use Case/Class Diagrams.	
	The diagram helps captures the logic of the use case with precision and it also helps to visualise the complex processes by showing the overall flow of control with appropriate notation. At no point in our activity diagram can we represent multiple plants - we assume that this diagram represents a single order. If we cannot assume this, this is a weakness.	[/ **10**] 9

continued on the next page

continued from previous page

Deliverable Part	Questions	Marks
	Q9. Assess the extent to which your activity (statechart) diagrams are consistent with the class model.	
	Yes, it represents an accurate step-by-step representation of the class diagram. It provides a flowchart view of how the classes interact with each other, and provides a direct path from order processing to order completion. Unlike the sequence diagram, this models the behavior for multiple orders being place at once.	[/ 10] 10
Java Prototype	**Q10.** Do your Java classes make good use of Object-Oriented features (e.g., inheritance, visibility, etc.) intended to support Object Oriented design, team work and reuse?	
	The classes reflect the nature and complexity of the object by capturing there state and related behaviours with the help of parameters and methods. Variables are set to the required visibilities when needed and methods have the appropriate return values. Also, some methods, while not being implemented, were left in the code for potential reuse, demonstrating a foresight . Comments describe all functions.	[/ 5] 5
	Q11. Assess the extent to which your Java classes are consistent with the class diagram.	
	The class diagram and Java classes are remarkably similar, since one models the system behavior and the other implements it. Since the implementation is a complete representation of the system, it is almost identical. The only part not represented in our Java classes is the deliveries, because that was outside the scope of our project.	[/ 5] 5
Testing	**Q12.** Does the code built to implement your tests compile and test the system in the manner you intended in your test plan. Provide a justification for all the tests you want to apply to your chosen classes.	
	Our design allows us to implement jUnit testing on the majority of our classes (the ones crucial to system functionality). Everything runs as it should with no compilation errors and it delivers the expected software. All tests ensure that our system performs the correct actions given a method, or returns the correct value. This ensures consistency and correctness throughout the project.	[/ 5] 5

continued on the next page

continued from previous page

Deliverable Part	Questions	Marks
	Q13. Do your classes pass all the tests? Do you believe your test code tests the system adequately? If Java classes fail to pass some testes, you should explain the problem and whether or not you have fixed it.	
	The system passes all tests provided. It either checks the expected values against the method calls, or checks the expected state after a void method call. It behaves adequately and there are no errors in the implementation.	[/ 5] 4
	Q14. Assess how effective your tests are in detecting coding errors in your chosen classes. Provide a justification in term of the coverage you are achieving in the tests.	
	To ensure there is an objectivity in our tests (to ensure we check for all possible errors), we perform multiple iterations of test instances in every test method. This ensures that a faulty result, if it existed, would be caught. Because every major class has test cases, the major functionality should perform as expected. A potential flaw would result from our tests being tailored to our expectations, but since the code is designed around these expectations, it should function properly.	[/ 5] 5
	Q15. How extensible is your test code? If you decided to increase the number of tests how much extra effort would it take to incorporate those tests. Award higher marks for greater extensibility. Justify your award of mark by making a case for the extensibility of the test set.	
	For each individual method within our main classes, we provide a test method that checks the desired outcome. The only code that we have not tested is the Runner class (which currently functions as a user interface) and the various print statements alerting the user as to the success or failure (since these are evident).	[/ 5] 5
	Q16. Have you discussed and addressed issues of integration (e.g., integration between different classes and different system parts)?	
	We have discussed integration and implemented a mock runner class to emulate the delivery team's functionality. Upon integration, we would remove the user interface, and allow our system to function autonomously with delivery's inputs. Also, the database we have employed would need to be integrated with the entire system.	[/ 5] 5
Deliverable Mark		[/100] 97

Deliverable 2 Marking Scheme - Instructions

This form is to be used by your team to assess your deliverable 2 and to distribute the mark among your team's individual members. *Please complete and return the form together with your deliverable 2.* This form consists of three parts.

Part 1 - Team Information. This part records your team's details. Please fill in the table in the Part 1 with the relevant information.

Part 2 - Individual Marking Scheme. This part records your team's distribution of the mark among individual team members. The final mark for an individual is the **Deliverable Mark** (see Part 3). **The marks have to be justified and accompanied by supporting cases.**

Part 3 - Deliverable Marking Scheme. This part provides you a marking scheme for your deliverable 2. Your deliverable 2 should consist of different parts: *Prioritised Use Cases*, *Refined Class Diagram*, *Change History*, *Sequence (Communication) Diagrams*, Activity (Statechart) Diagrams, *Java Implementation* and *Testing*. The Deliverable Marking Scheme in Part 3 shows the maximum mark allocated to each part. You have to assess each part of your deliverable 2. The table provides relevant questions that structure your assessment process. You should assign (part of) the allocated marks for each question. The marks depend on the quality of your work. **Note that you have to justify any allocated mark by answering each corresponding question and providing an assessment supporting your mark allocation.** *You have to justify your mark assignment by answering the questions and proving an assessment in a separate sheet.* The **Deliverable Mark** is the sum of the marks of each part.