**2805ICT System and Software Design**

**3815ICT Software Engineering**

**7805ICT Principles of Software Engineering**

This is an assessed workshop

**WORKSHOP 3**  **UML & first stage assessment**

Student name: Nguyen Quang Huy Student ID: s5257464 Enrolled Course Code: 7805ICT

**1. Cohesion & Coupling (15pts)**

1. What is cohesion, and what is coupling (5pts)?

* Cohesion: The degree to which elements of a component or object interdepend.
* Coupling: The degree to which components or objects depend on one another

1. To improve the quality of a software system, we need to increase or decrease cohesion? Use examples to explain bad and good design regarding to cohesion (5pts).

* To improve the quality of a software system, we need to increase cohesion.
* Example of good design: frequently used methods (funcs) are defined inside classes (in the same file), which improve code reusability and readability
* Example of bad design: methods/funcs are not defined in whatever file(s) it is needed, this decrease code reusability since the method must be defined multiple times.

1. To improve the quality of a software system, we need to increase or decrease coupling? Use examples to explain bad and good design regarding to coupling (5pts).

* To increase the quality of a software system, we need to decrease coupling
* Example of bad design: a system with multiple (a lot) of dependencies like multiple sub-classes, changing something in Module A might cause a ripple effect that affect other modules (childs) in an undesired way(s)
* Example of good design: Modules are clearly separated, there are less dependencies

**2. Requirement Engineering (15pts)**

1. What are the four common software requirement defects? (5pts)

* Incomplete, ambiguous, too weak, too strong

1. What is a formal way (formular) to write high quality requirements? Rewrite the requirement below by using this way (10pts)

The formal way to write: User preconditions, events, post conditions and constraints

“*A user can operate the microwave oven by pressing the start key to make it start to cook for one minute*”.

As an user, I can press the start key to start the “one minute cooking procedure” of the microwave oven

**3. OO design principles explanation (10pts)**

1. Explain the meaning of encapsulation, inheritance, and polymorphism with examples in OO design (5pts).

* Encapsulation: The combining of attributes and behaviour into a single object whereby the actual implementation details are hidden.
* Inheritance: The ability of an object to indicate that it will acquire the attributes and behaviour of some other objects.
* Polymorphism: The ability of different objects to perform the appropriate behaviour in response to the same message

1. Discuss the benefits and disadvantages of them (5pt).

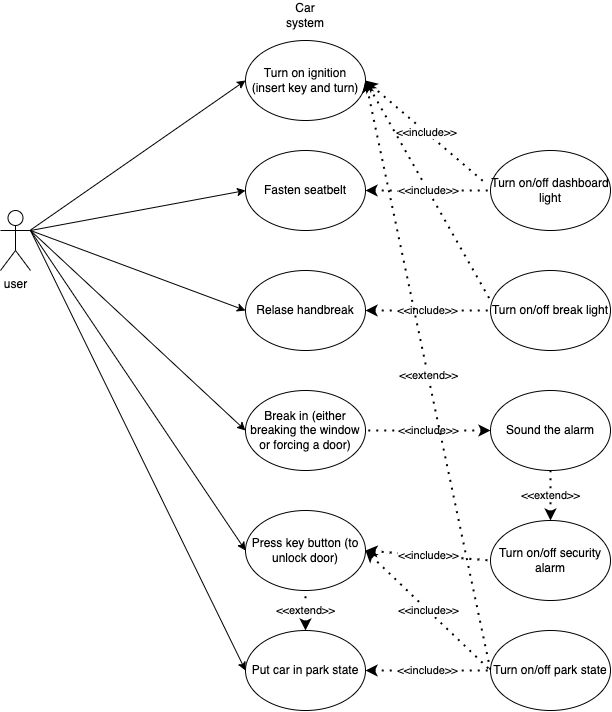
|  |  |  |
| --- | --- | --- |
|  | Advantages | Disadvantages |
| Encapsulation | * **Data Protection**: The program runner will not be able to identify or see which methods are present in the code. Therefore he/she doesn’t get any chance to change any specific variable or data and hinder the running of the program. * **Flexibility**: The code which is encapsulated looks more cleaner and flexible, and can be changed as per the needs. We can change the code read-only or write-only by getter and setter methods. This also helps in debugging the code if needed. * **Reusability**: The methods can be changed and the code is reusable. | * **Code Size:** The length of the code increases drastically in the case of encapsulation as we need to provide all the methods with the specifiers. * **More Instructions**: As the size of the code increases, therefore, you need to provide additional instructions for every method. * **Increased code execution:** Encapsulation results in an increase in the duration of the program execution. It is because more instructions are added to the code therefore they require more time to execute. |
| Inheritance | * Makes real world representation of problems easier * Problems are now modularized, solving takes lesser time. * Improves data security by the use of protected keyword. * Makes code simpler to understand and implement | * two classes (base and inherited class) get tightly coupled. * This means one cannot be used independently of each other. * Also with time, during maintenance adding new features both base as well as derived classes are required to be changed. |
| Polymorphism | * It helps programmers reuse the code and classes once written, tested and implemented. They can be reused in many ways. * Single variable name can be used to store variables of multiple data types(Float, double, Long, Int etc). * Polymorphism helps in reducing the coupling between different functionalities. | * One of the disadvantages of polymorphism is that developers find it difficult to implement polymorphism in codes. * Run time polymorphism can lead to the performance issue as machine needs to decide which method or variable to invoke so it basically degrades the performances as decisions are taken at run time. * Polymorphism reduces the readability of the program. One needs to identify the runtime behavior of the program to identify actual execution time. |

**4. Use case diagram (15pts)**

Based on the requirements given below to draw a use case diagram. Marking criteria:

1. All the use case diagram elements are included correctly.
2. All functional requirements are covered
3. All use cases are named properly (as verb + noun )
4. Correctly use include and extend relationship





**5. Class diagram (20pts)**

Based on the requirements given in question 2, draw a class diagram. Marking criteria:

1. All functional requirements are covered, all classes are identified.
2. Suitable methods should be given.
3. Correctly use UML notations.
4. You need to create a class Light as parent class for BrakeLight and DashboardLight
5. Should include correct multiplicity
6. Should include all three different types of relationship: aggregation (composition), generalization, and association
7. Use CapitalCamelCase for class names and lowerCamelCase for method names.

Diagram

Description automatically generated

**6. Sequence diagram (15pts)**

Based on the requirements given in question 4, draw a sequence diagram to describe the scenario: the driver press key button to switch the alarm off and unlock the door, and then the driver open door to enter the car and put the car to park. Marking criteria:

1. Correctly use sequence diagram notation.
2. The class names and method names must match what in the class diagram.

Diagram

Description automatically generated

**7. Interview (10pts)**

The workshop teacher (assessor) will arrange a one-to-one interview at week 10 or 11 workshop time. During the interview, the assessor may ask you to explain your answers and discuss the marking. The interview will also cover your work in the assignments. The purposes of the interview are:

1. Help you to clarify marking issues.
2. Check your understanding.
3. Check if the submitted work was done by you and other possible academic integrity issues.

If the assessor suspects that some of the submitted work was not completed by you, it might raise a concern and might deduct your marks accordingly.

**8. Additional exercises for 3815ICT and 7805ICT**

This question is not assessed, however, if you don’t complete this question, your other questions will not be marked (the question is only for students who enrol in 3815ICT and 7805ICT).

Write 15 lines of a reflective report on the previous activities. Analyse and evaluate their relevancy to your future work.

* Cohesion and coupling: this question tested my general understanding about the pro and cons of cohesion and coupling. This further emphasize the importance of good coding practice in my assignment
* Using a formular way to write requirement helped me in communicating between members of my group
* Use case diagram help me identify all the functions needed for the assignment application
* Sequence diagram gave me a general idea of the application class structure
* I will be drawing a lot more sequence diagram to figure out the limitation of my assignment application

**9. Additional exercises for 7805ICT**

This question is not assessed, however, if you don’t complete this question, your other questions will not be marked (the question is only for students who enrol in 7805ICT).

Design an open-ended question (that means there may be several correct answers) that could be suitable for

1. A final exam

Give a quick example of the importance of UML in software engineering

1. A job interview for software engineer.

\*Give the interviewee a class diagram

“Within 15 minutes give me a scenario where this system would fail, what is your suggestion to fix said loophole?”