2805ICT System and Software Design
3815ICT Software Engineering
7805ICT Principles of Software Engineering
This is an assessed workshop

WORKSHOP 3 **UML**

Student name: **Yasin Cakar** Student ID: **s2921450** Enrolled Course Code: **3815ICT**

Student name: **Mohammad Mari** Student ID: **s5185052** Enrolled Course Code: **7805ICT**

Student name: **David Todorovic** Student ID: **s5167246** Enrolled Course Code: **3815ICT**

1. OO design principles explanation

- Explain the meaning of encapsulation, inheritance, and polymorphism with examples in OO design (10pts).

Answer:

Encapsulation: The combining of attributes and behaviour into a single object whereby the actual implementation details are hidden, this is also referred to as information hiding, an example of that would be a "class" all the internal implementation of an object is encapsulated in a class.

Inheritance: The ability of an object to indicate that it will acquire the attributes and behaviour of some other objects, "Inheritance" is an important concept of OO design because it allows for code reuse for example if class "A" has the "foo" method and class "B" is inheriting class "A" then class "B" will have access to "foo" methods as well

Polymorphism: The ability of different objects to perform the appropriate behaviour in response to the same message, which means having the same function name being used for different object types, an example of that: assuming we have class "A" that has a method inside it called "type" and this method returns a string "This is class A" and we have class "B" which has the same "type" method as class "A" and it returns the string "This is class B" so when creating instances of both classes were still using the same method name to find out the type IE:
a = A()
b = B()
a.type() → "This is class A"
b.type() → "This is class B"

b) Discuss the benefits and disadvantages of them (10pt).

Encapsulation:

Benefits:

- Modifying the code becomes easier as well as the testing aspects since all the logic is encapsulated inside an object, hence, changes will happen in one area.

Disadvantages:

- Developers have to do extra work (more code)

Inheritance:

Benefits:

- Allows code reuse – classes can inherit methods and attributes from other classes; hence it eliminates redundant code.
- Since inheritance promotes for code reuse, this leads to less development time and updates become easier.

Disadvantages:

- Most of the time not all of the methods from the parent class will be used by the child class, hence, the child class may inherit many methods from the parent class and only use a few of them (memory waste).
- Increase coupling between parent and child class (IE. Any changes in the parent class will impact the child class as well).
- Performance issues – inherited methods are slower that none-inherited ones (this may vary from language to language and between different compilers).

Polymorphism:

Benefits:

- Enhances code reusability, maintainability, and readability.

Disadvantages:

- Performance issues – it may be slower when using polymorphism functions.

2. Use case diagram

Based on the requirements given below to draw a use case diagram (30pts). 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

Answer:

PROBLEM STATEMENT - Car System

It is required to do an analysis and high-level design for part of the functional requirements for a computerised Car System with an automatic gear-shift. The following details apply:

FR1: The car can only be started if it is in the PARK state when the driver inserts the key in the ignition and turns it on.

FR2: A dashboard light remains on if the driver's seatbelt is not fastened when the driver is seated and the ignition is on.

FR3: If the handbrake is on when the ignition is on, the brake-light turns on.

FR4: The security alarm is on when the car is locked, and if anyone tries to break in by breaking a window or forcing a door the alarm will sound.

FR5: When the driver, on approaching the car, presses the key-button it unlocks the door and turns the security alarm off.

FR6: When the car is unlocked the driver may get in and put the car into the park state.

Model the whole process from the driver pressing the key-button, to the driver seated, to having started the engine.

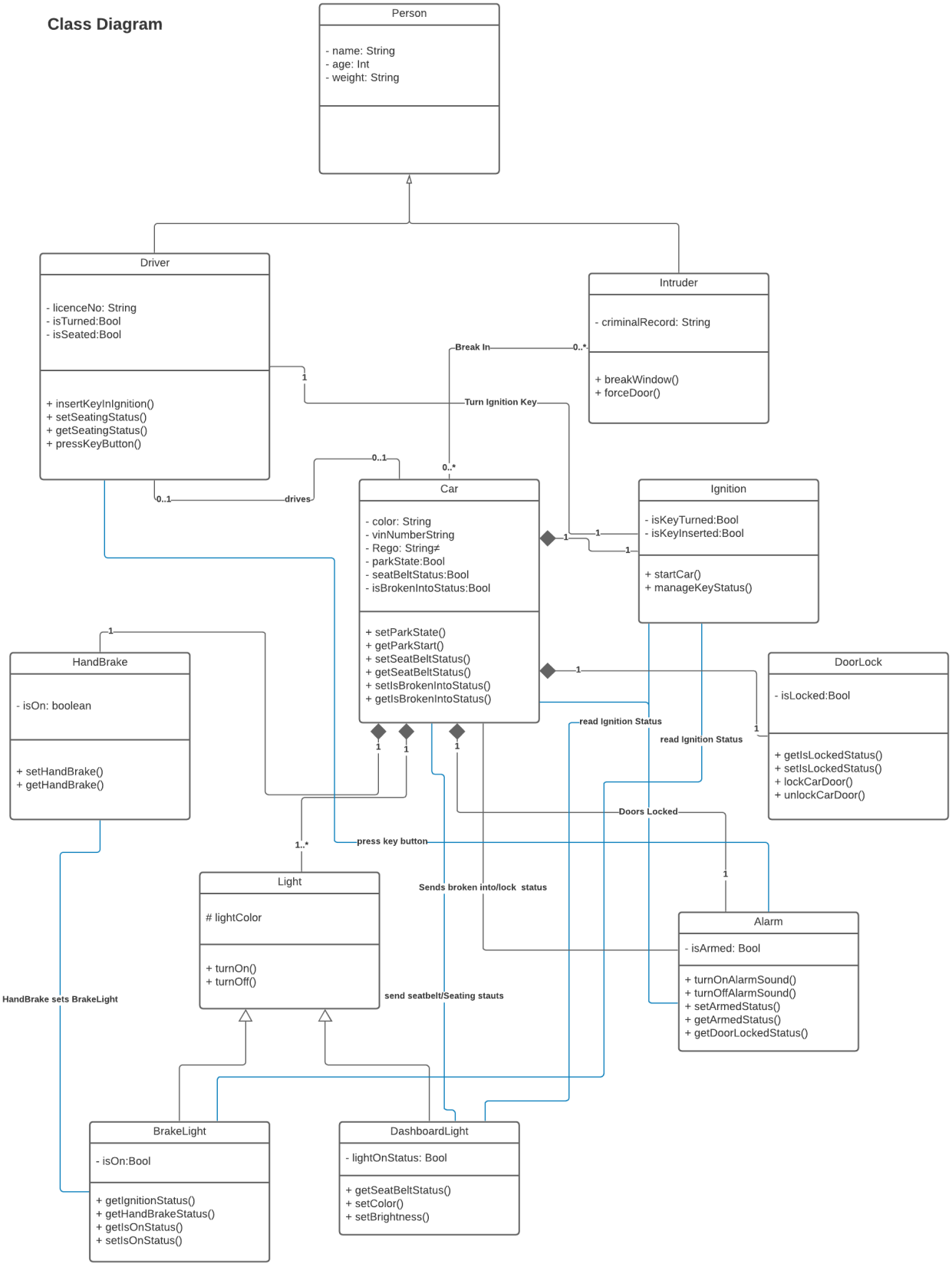
3. Class diagram

Based on the requirements given in question 2, draw a class diagram (30pts). 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
7. Use CapitalCamelCase for class names and lowerCamelCase for method names.

Answer:

Class Diagram



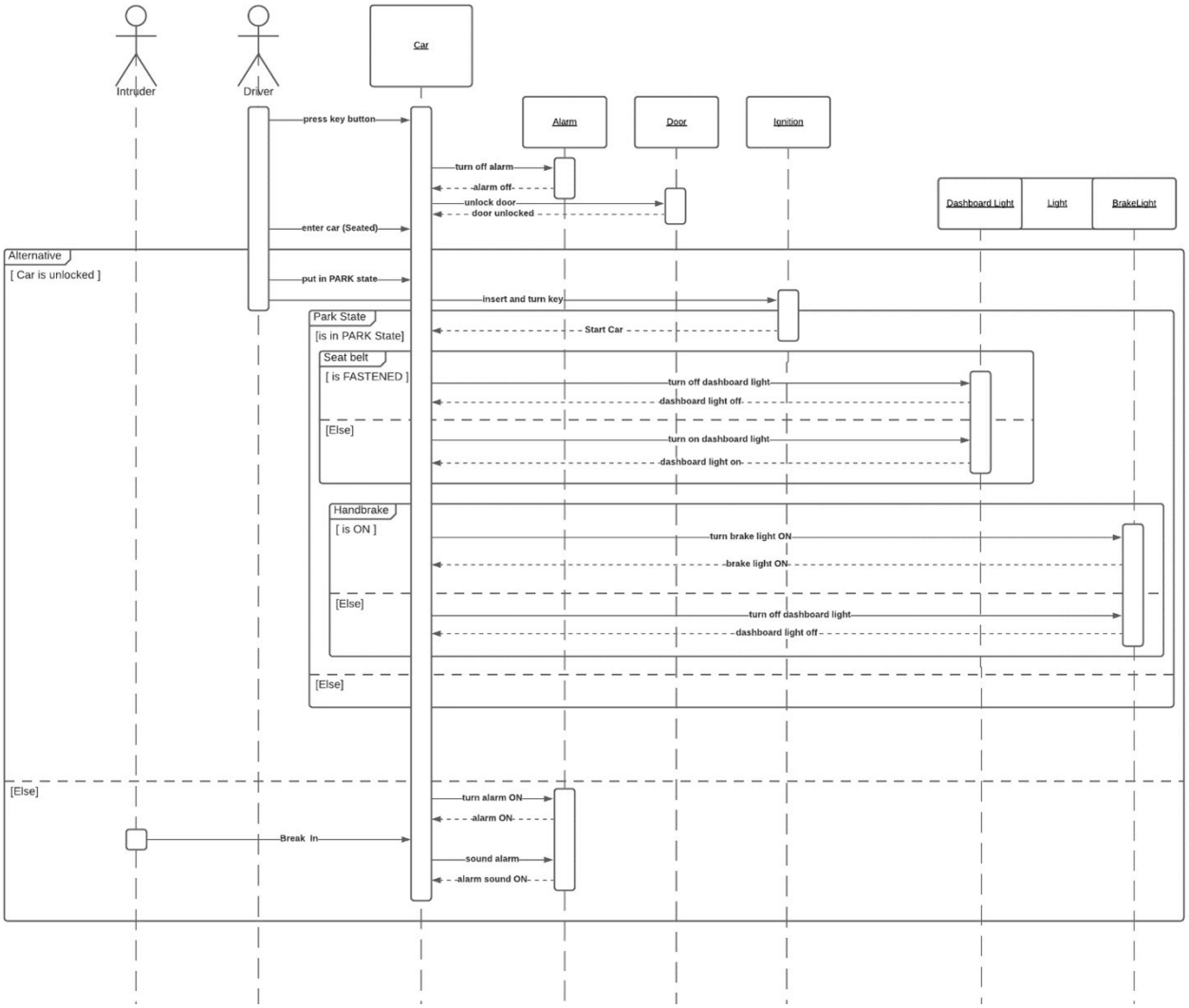
4. Sequence diagram

Based on the requirements given in question 2, draw a sequence diagram to describe the senior: the driver press key button to switch the alarm and unlock the door, and then the driver open door to enter the car and put the car to park (20pts). Marking criteria:

1. Correctly use sequence diagram notation.

2. The class names and method names must match what in the class diagram.

Answer:



5. 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.

Answer:

All the modelling methods, use case, sequence diagrams, and UML class diagrams provide different views of the interested system to help identify the system domain, stakeholders, functionality, and the flow of events within the intended system.

These three methods are very resourceful in identifying, requirements deficiencies such as ambiguity, redundancy, inconsistency, and incompleteness as well as identifying missing requirements.

Use case diagrams to show all the system domains, processes, and actors involved at a high level. They provide a functional view of the systems by showing the structure of the systems and their interaction with the external environment and the flow of data between all of its elements.

Class diagrams represent individual entities as objects rather than functions, encapsulating entities into objects this assists in providing a clean design with separation of concerns for complex problems.

Sequence diagrams complement use-case and class diagrams by involving actors/stakeholders with their system domains where all entities and actors are in an ordered flow of events within the context of the system. This will highlight any deficiencies in the representation tools mentioned above.

Each process assists in understanding elements of the other process. For instance, a use case diagram provides the big picture, while the sequence diagram can assist in identifying methods in the class diagram using the sequential interactions between actors and use cases in the system.

6. 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
2. A job interview for software engineer.

Answer:

In some system domains, certain activities depend on two or more previous activities to be fulfilled as a precondition. Sometimes we describe this using the logical OR and logical AND conditional statements. What approach would you use to represent such requirements using the UML modelling tools we have covered and why?