THE UNIVERSITY OF SYDNEY

SOFT2201/COMP9201 Week 3 Tutorial

UML and Implementation

UML

Unified Modelling Language is a visual language that allows software designers to visualise data and functionality of their system and how they interact with each other. There are several different UML diagram types, in this tutorial we will be focusing on the class diagram.

Question 1: Heirarchy and Properties

You have been provided an example UML class model, this is a simple class hierarchy, outlining shared properties and functionality between the base class and sub-types.

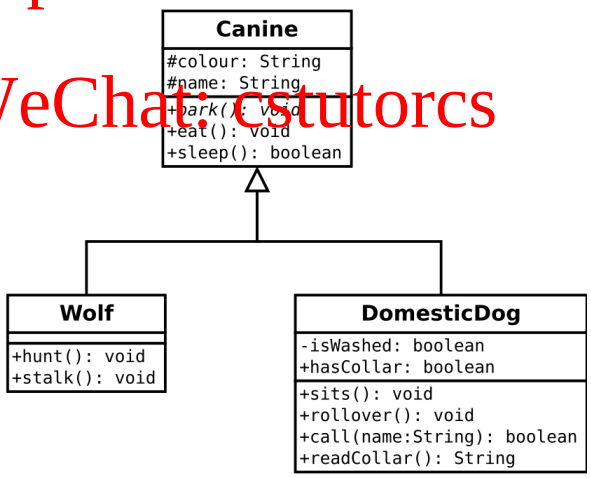
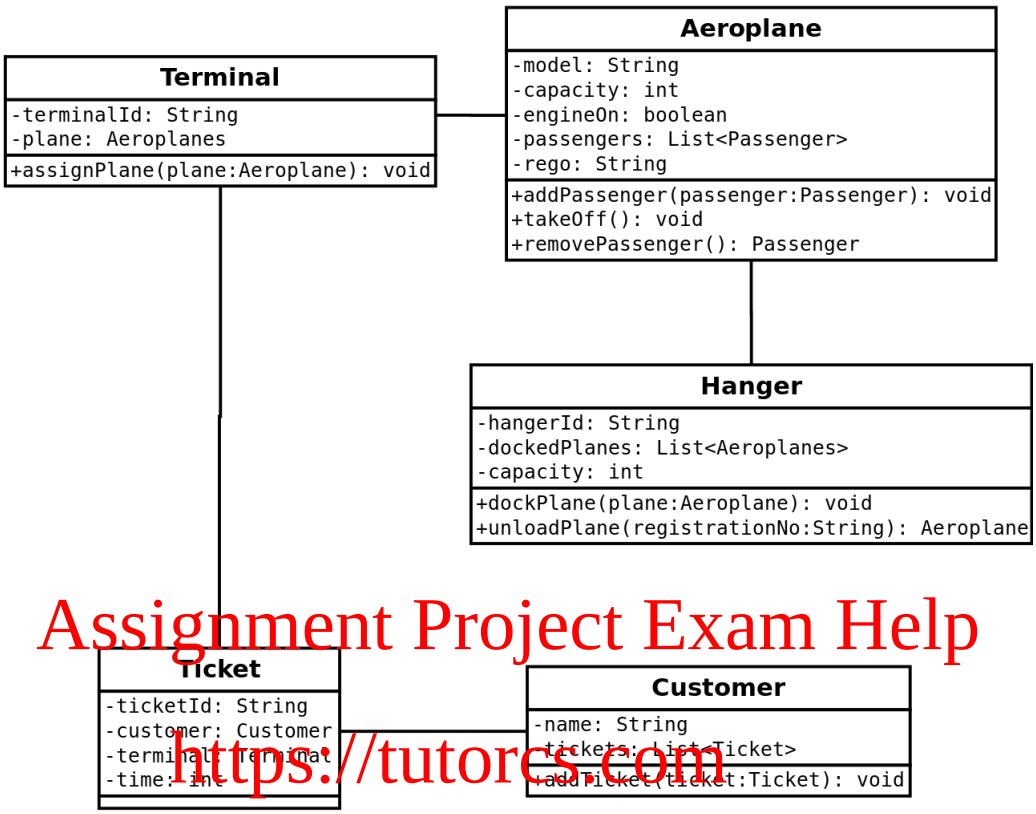


Figure 1: UML Class diagram

Discuss and answer the followings:

- What is the parent class of `Wolf`?
- What properties are shared between `DomesticDog` and `Wolf`?
- What methods are shared between `DomesticDog` and `Wolf`?
- Outline the access level associated with each field
- Does either sub-type of `Canine` override one of its methods?

Question 2: Associations



Analyse the UML diagram and identify and answer the following:

- Outline all the associations within the UML Diagram
- How many `Aeroplanes` can a `Hanger` contain? Annotate the multiplicity of the association
- How many `Tickets` can a `Customer` hold? Annotate the multiplicity of the association
- Identify any association relationships where an object cannot exist without at least one other type (Composition)
- Identify any aggregate relationship within the diagram

GRASP Guidelines

GRASP is a mental toolset, a guideline to follow when designing software. Identifying pattern violations can provide an understanding of flaws within the software architecture.

As a quick summary of the GRASP guidelines.

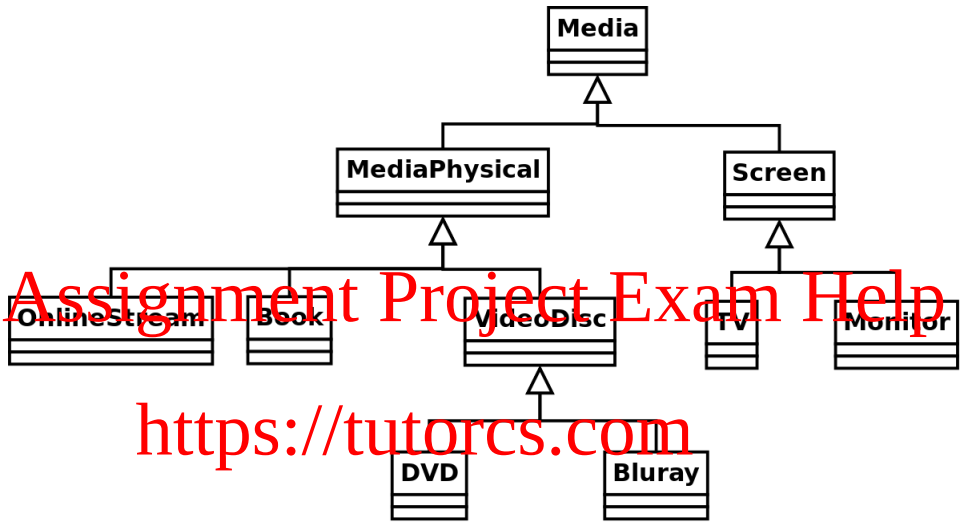
- Information Expert, Assign a responsibility to the information expert - the class that has the information necessary to fulfill the responsibility.
- Creator, Responsibility of object creation.
- Controller, can represent two components of a system. 1. Represents the overall system, a root object which represents a major subsystem. 2. Represents a use case scenario within which the system operation occurs (a use-case or session controller).
- Low Coupling, how an object type is connected to another type. Consider the assignment of responsibilities and any unnecessary binding to other types.
- High Cohesion, Where responsibility of the element share common characteristics (properties, methods, associations...).
- Polymorphism, When related alternatives or behaviour vary by type, assign responsibility for the behaviour using polymorphic operations to the type for which the behaviour varies.
- Pure Fabrication, Assign a highly cohesive set of responsibilities to an artificial or convenience behaviour class (or type) that does not represent a problem concept-something made up, in order to support high cohesion, low coupling, and reuse.
- Indirection, Assign the responsibility to an intermediate object to mediate between other components or services, so that they are not directly coupled
- Protected Variations, Identify points of predicted variation or instability; assign responsibilities to create a stable interface around them.

Question 3: SOLID vs GRASP

Discuss with your class and tutor about **SOLID** and **GRASP**. Identify overlapping concepts between the principles and discuss how each principle applies to software design.

Question 4: Critiquing a model

Given the following class diagram, discuss with your class about issues with the current application model.



Common issues that arise from improper modelling:

- Weak base classes
- Improper parent or interface type associated
- Repetitive logic and data
- Class dependent on other classes
- Classes performing more functionality than it should be
- Misassigned responsibility to an object

Consider what **GRASP** or **SOLID** principles have been violated by the heirarchy.

Question 5: Extract Pokemon

You have been given a file that stores pokemon data called `pokedex.json`. Use the `simple-json` library as a way to read the pokedex.json file and output the contents to the `stdout`.

You can add the dependency to your project by adding the following to the dependecies section in your gradle.build file.

```
implementation 'com.googlecode.json-simple:json-simple:1.1.1'
```

The element within pokedex.json file is in the following layout:

```
{
    "kanto_no" : index,
    "type_1" : first_type,
    "type_2" : second_type,
    "pokemon" : name
}
```


Import the following types into your project and attempt to parse the `pokedex.json file`.

```
import org.json.simple.JSONArray;
import org.json.simple.JSONObject;
import org.json.simple.parser.JSONParser;
```


JSON-Simple provides some examples in its wiki, you can access this here.

Read the contents of the file into a `String` object and utilise the `.parse` method to interpret the data into a `JSONArray`. You may need to apply a cast on the return result.

Question 6: Modelling

As a group, you will need to construct a UML class diagram, assigning responsibilty to elements identified. After you have finished mapping the UML class diagram, discuss and rationalise your decisions with your tutor.

You are designing an point of sale system for a bike shop, the bike shop sells variety of bicycles such as mountain, commuter, road, single speed and electronic bikes. A customer is able to purchase a bike, services (such as repairs and maintenance) or other products such as a tubes, locks, grips, helmets and lights.

To ensure the system can be audited and for orders to be managed, purchases are grouped by an order, one or more items that are purchases are incorporated in an order. Each order belongs to a customer. Each order has a total amount to be paid, payment method (BankTransfer, Cash, Credit) order number and payment date.

Question 7: From design to code

Once you have modelled the application, implement your model and try and make a workable point of sale system using the existing interfaces within the app.

You will need to use the `registerProduct` method within the `App` class to make a `Product` selectable from UI.

Once you have implemented the interfaces correctly for the types within your model, ensure that the following features work:

- Items are accessible from the drop down menu
- When purchasing a type from the drop down, it will add an entry within the table
- It will allow the user to specify quantity of a product
- It will output an csv containing the items purchased