ITCS 209: Object-Oriented Programming

Project 03: Object-Oriented Design in Real-World Application

Due: Friday April 30, 2021 11:55PM

Learning Objectives:

On the course of implementing this project, you will learn some of the basic concepts of the object oriented programming paradigm, and how to apply them to practical, real world programming applications. Specifically, upon accomplishing this project, we expect you to be able to:

1. Be familiar with Concept of Object-Oriented Design.
2. Understand business domain and how to convert requirements to functions.
3. Be a good problem solver, think logically, and design a UML diagram to solve business problems.
4. Enjoy designing with OOP concept.

Introduction:

Digital Technology today is more important now than ever. It plays a virtue role in enabling efficiency and more productivity to any business and organization. However, developing applications to support any business is somewhat complex and require a proper design. The Object-Oriented Concept is one technique that helps in addressing the complexity of the design. In this project, you have to think of any ideas to create your application that would be interesting or useful for others, such as Clubhouse, Twitter, Netflix, Food Panda, Lazada, etc. You have to describe your application, list the function that it could do, design a UML diagram, and implement a wireframe for UI.

Members: 3 Students per group

The thing to do:

1. Select any **interesting domain**.
2. Write a description of the application and state its benefit to others.
3. List all Classes/Interface, their variables, and their Method that the application could do; then draw a UML diagram, using https://www.lucidchart.com/, or other tools with its relation (extends or implements) with the following constraint.

   The Application:
   - Should have at least 5 Class with at least two extensions.
   - Should have at least 2 Interface with at least two implementations.

   Each Class:
   - Should contain at least 3 access modifiers for the variables: public, protected, and private.
   - Should have at least 2 Constructors.
   - Should have at least 6 Methods (exclude Overriding and Overloading methods).
     - **Methods** without parameters at least 2 Methods.
     - **Methods** with parameters at least 2 Methods.
     - **Static Methods** at least 2 Methods.
   - Should have at least 1 Overriding Method and 1 Overloading Method.

   Each Interface
   - Should have at least 2 abstract Methods.
     - **Methods** with return parameters at least 1 Methods.
     - **Methods** without returning parameters at least 1 Methods.

4. Use the UML diagram to produce an API document of the Class/Interface in (3) in the following format:
   - Class/Interface name
   - Class/Interface inheritance
   - Constructor Detail
   - Methods Detail

5. Design a UI Wireframe of the application, using https://app.moqups.com/.
   - Design at least 3 UIs.
   - Design a link between each UI that calling based at least on 1 Method designed in (3).
   - Descript how the application work.

Note that: An example of the API document is provided in an Appendix.

Presentation of the application [In-class/online]**: week 15***

*timeslots will be announced before the presentation date.

Need help with the project?

If you have questions about the project, please first post them on the forum on MyCourses, so that other students with similar questions can benefit from the discussions. The TAs can also answer some of the questions and help to debug trivial errors. If you still have questions or concerns, please make an appointment with one of the instructors. ***We do not debug your code via email***. If you need help with debugging your code, please come see us.

Academic Integrity (Very Very Important)

Do not get bored about these warnings yet. But please, please ***do your own work***. Your survival in the subsequent courses and the ability to get desirable jobs (once you graduate) heavily depend on the skills that you harvest in this course. Though students are allowed and encouraged to discuss ideas with others, the actual solutions must be originated and written by themselves. Collaboration in writing solutions is not allowed, as it would be unfair to other students. Students who know how to obtain the solutions are encouraged to help others by guiding them and teaching them the core material needed to complete the project, rather than giving away the solutions. ***You can't keep helping your friends forever, so you would do them a favour by allowing them to be better problem solvers and life-long learners.*** Your code will be compared with other students (both current and previous course takers) and online sources using state-of-the-art source-code similarity detection algorithms, which have been proven to be quite accurate. If you are caught cheating, serious actions will be taken, and severe penalties will be bestowed on all involved parties, including inappropriate help givers, receivers, and intermediaries.

Appendix

Example of API Document (Detail)

The Document must have all following section:

1. **Cover Page**: Must contain project title and name of your group members.

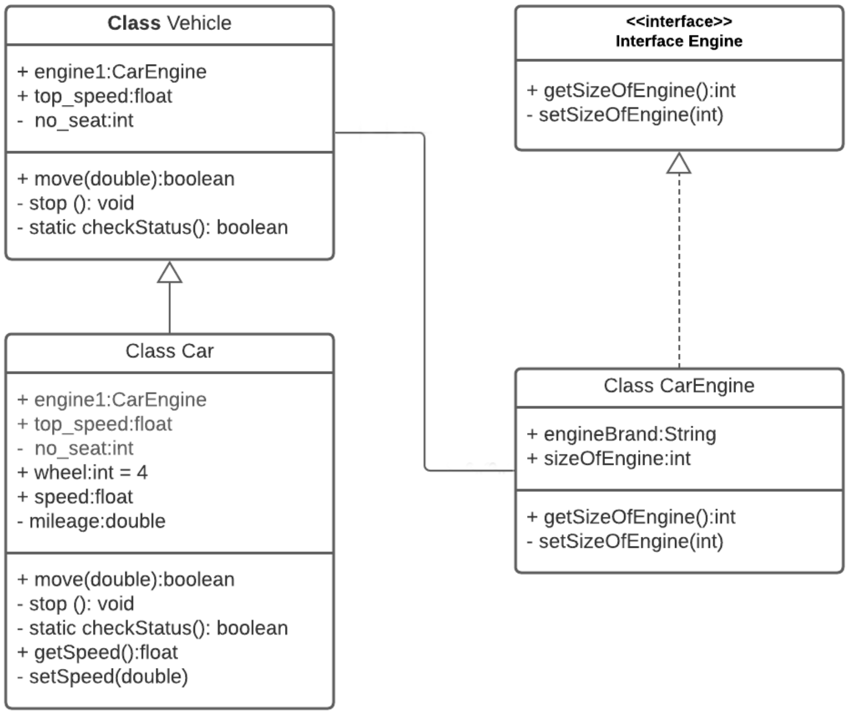
2. **Introduction**: explain about the application as the following list:
    a. Description of the application.
    b. Explain about the Benefit.
    c. Explain about the Application Requirements.

3. **Classes and Interfaces**: show list of all classes and interfaces in Table format as the following format.

Class	Description
Vehicle	The Vehicle class contains variables and methods of general vehicle. It uses as a blueprint for other class that want to design their own vehicle.
Car	The Car class is use to create a instance object Car with its properties and methods.
CarEngine	The CarEngine class implements from Engine to create an instance object Car's engine as an object in the class Car.

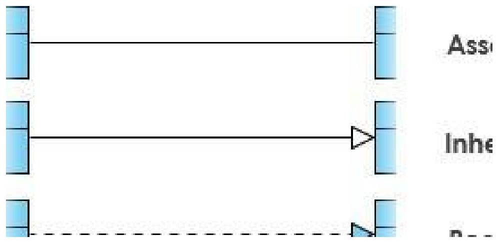
Interface	Description
Engine	Engine is an Interface contains abstract Methods for building Engine.

4. **UML Diagram**: show UML diagram of all classes and interfaces presented in (3). Example of UML Diagram is as follows:



Hint: The detail of how to write a UML diagram can be found in the following links:

- The following three relations are the must to include in the UML: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/uml-class-diagram-tutorial/



- https://www.lucidchart.com/pages/uml-class-diagram#section_2

5. **API document**: list of all Classes/Interface with their Constructors and Methods as the following template:

Example of Class Car:

Class Car

public class **Car**
extends Vehicle

Class Description: The class Car contains variables and methods for representing an instance object Car extends from Vehicle. The object car provides methods to get a current speed and set a new speed. The engine status can be checked through the method checkStatus().

Constructor and Description	
Constructors	**Description**
Car()	Construct a new car object with default variable; wheels=4, speed=0, and mileage=0.
Car(int speed)	Construct a new car object with fix speed variable.

Methods	
Modifier and Type	**Description**
public boolean	**Move(double speed)** • Throw a IllegalArgumentException if the speed parameter input is negative. • This method is invoking for set the status of Car when speed parameter is greater than '0'. **Parameters** • speed – the speed of the instance object Car. **Throws:** • IllegalArgumentException – if the speed is empty or negative.
private void	**Stop()** • This method is for setting the status of Car to false by calling the method setSpeed(0). **Parameters** • None **Throws:** • None
private Boolean	**static checkStatus()** • This method is for checking the status of Car. There are two states "true" and "false", which indicate the status of engine "on" and "off" respectively.

	Parameters • None **Throws:** • None
public float	getSpeed .
private void	setSpeed(double speed) .

Example of actual JAVA doc can be found in: https://docs.oracle.com/javase/7/docs/api/

6. **User Interface (Wireframe)**: show a mockup of your application and describing its flow as the following format (can use https://app.moqups.com/ any other tool is fine).

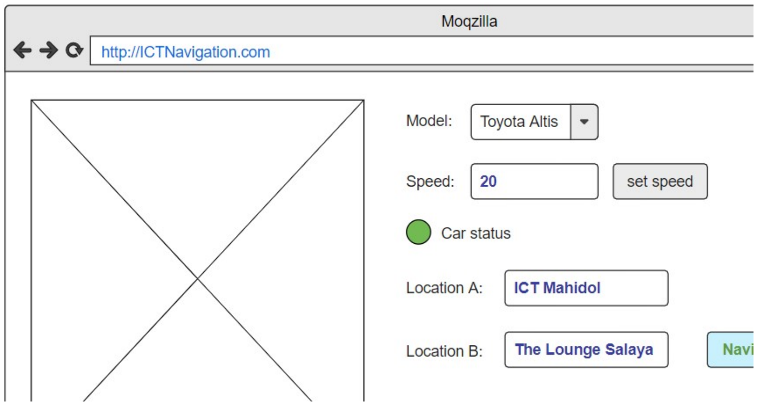


Figure1: The web application for showing navigation from location A to location B based on the Car Model.

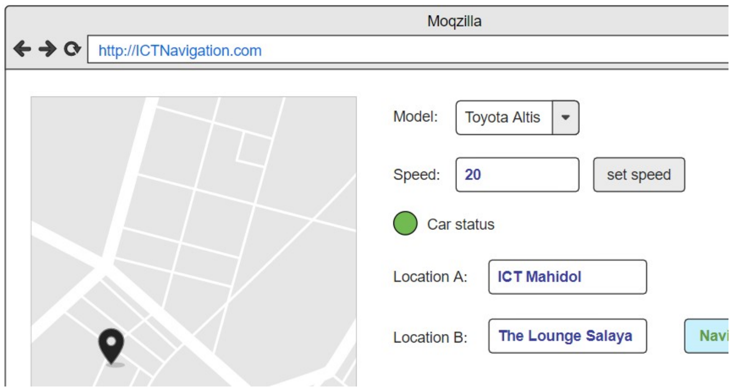


Figure2: The web application result after click navigate from Figure1.

The Application start from filling the car model and its speed. After click the set speed button, if the speed greater than 0, the car status will be changed to "green". The user, thus, can fill in location A and location B as presented in Figure1. The result of shortest path and time for the particular car model will be shown as presented in Figure2.