CPTR421 OBJECT ORIENTED DESIGN AND PROGRAMMING

This assignment is due Sunday 11 March and is worth 80 marks and accounts for 40% of your overall grade.

Uploaded your completed project folder to your Google Drive account, provided with your USC email address.

Ensure that you have turned on editing rights to your folder and share this link on the assignment submission area on E-Learn.

Please refer to this video if you are unsure how to perform the above-mentioned steps:

https://youtu.be/SW2PjSUcdwQ

An object-oriented application is required to manage information on bus stops and buses managed by a certain transit system. The application must provide a user interface that allows a user to perform the following operations:

- Add a new bus stop to a bus
- Query for a particular bus stop or bus
- List all the bus stops and buses managed by the Transit System

The application will consist of three domain classes, *BusStop*, *Bus*, and *Transit System*. The user interface of the application will be provided by another class, *TransitSystemApplication*.

UML Diagram for application

Figure 1 shows a simplified UML diagram of the BusStop, Bus and Transit System classes. A Bus object is related to many BusStop objects. Similarly, a TransitSystem object manages many Bus objects.

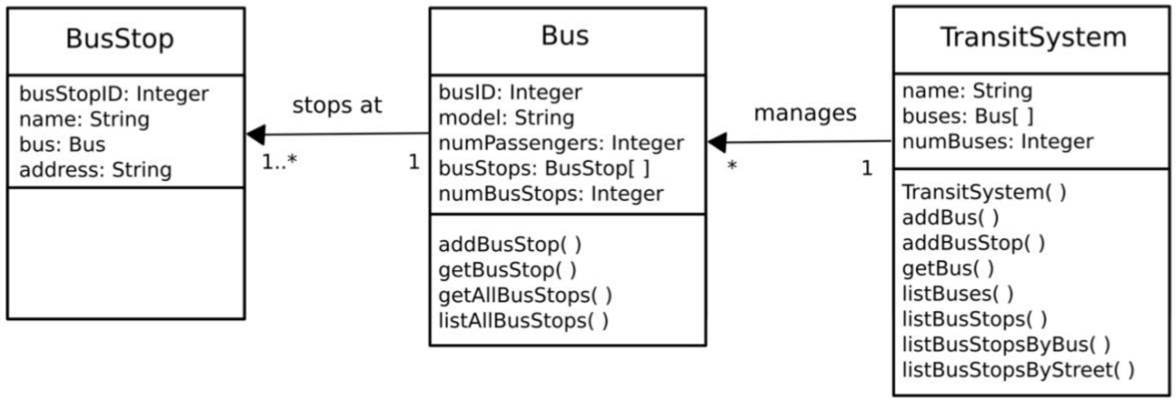
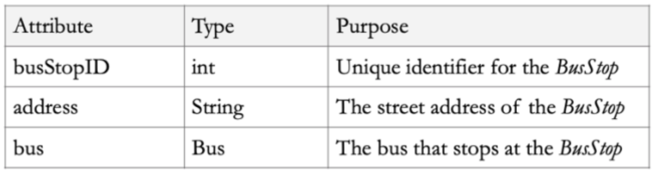


FIGURE 1 UML DIAGRAM FOR CLASSES

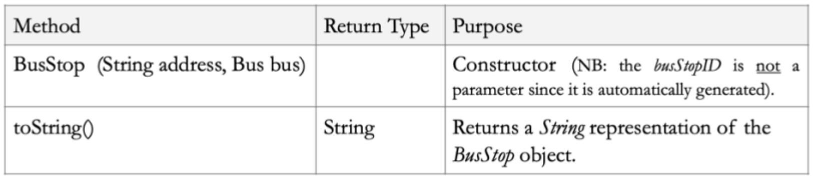
BusStop Class

The BusStop class models the concept of a bus stop for a Bus and is managed by the TransitSystem. Table 1 lists the attributes of the BusStop class.



Attribute	Type	Purpose
busStopID	int	Unique identifier for the *BusStop*
address	String	The street address of the *BusStop*
bus	Bus	The bus that stops at the *BusStop*

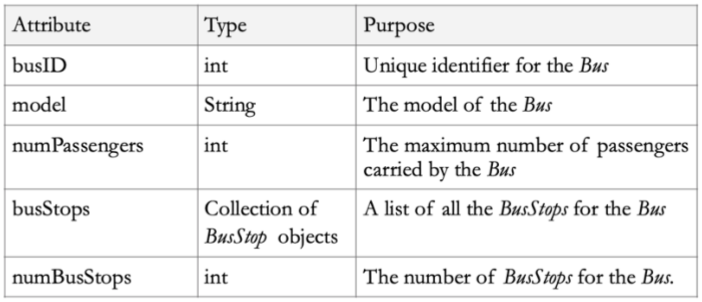
The ID attribute of each *BusStop* should be automatically generated. The first *busStopID* should be 1000 and each new *BusStop* should increment the previous ID by 10. The second *BusStop* will have an ID of 1010, the third will have an *ID* of 1020, and so on. Below shows the methods that should be provided by the *BusStop* class.



Method	Return Type	Purpose
BusStop (String address, Bus bus)		Constructor (NB: the *busStopID* is not a parameter since it is automatically generated).
toString()	String	Returns a *String* representation of the *BusStop* object.

Bus Class

The Bus class keeps track of the *BusStop* objects at which the bus will stop. It has a collection of *BusStop* objects and uses a variable *numBusStops* to keep track of the number of *BusStop* objects in the collection. Below lists the attributes of the *Bus* class. Bus IDs start at 1 and are incremented by 1



Attribute	Type	Purpose
busID	int	Unique identifier for the *Bus*
model	String	The model of the *Bus*
numPassengers	int	The maximum number of passengers carried by the *Bus*
busStops	Collection of *BusStop* objects	A list of all the *BusStops* for the *Bus*
numBusStops	int	The number of *BusStops* for the *Bus.*

The *Bus* class must also provide the methods

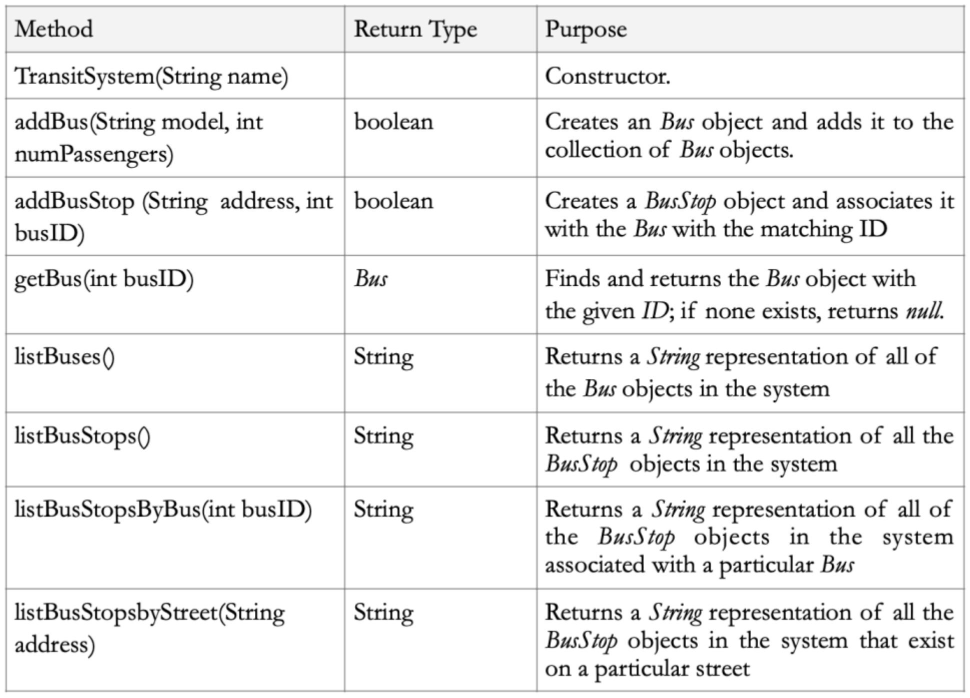
Method	Return Type	Purpose
Bus(String model, int numPassengers)		Constructor.
addBusStop (String address)	boolean	Creates a *BusStop* object and adds it to the collection of *BusStops* only if it does not exist.
getBusStop(int busStopID)	*BusStop*	Finds and returns the *BusStop* object with the given *ID*; if none exists, returns *null*.
getAllBusStops(String streetName)	String	Returns a *String* representation of all of the *BusStop* objects with the supplied street name in their address
listBusStops()	String	Returns a *String* representation of all the *BusStop* objects
toString( )	String	Returns a *String* representation of the *Bus* object.

TransitSystem Class

The *TransitSystem* class keeps track of all of the *Bus* objects. It has a collection of *Bus* objects and uses a variable *numBuses* to keep track of the number of Bus objects in the collection. Below lists the attributes of the *TransitSystem* class.

Attribute	Type	Purpose
name	String	The name of the *TransitSystem*
buses	Collection of *Bus* objects	A list of all of the *Buses* in the system
numBuses	int	The number of *Buses* managed by the *Transit System*

The *TransitSystem* class must also provide the methods



Method	Return Type	Purpose
TransitSystem(String name)		Constructor.
addBus(String model, int numPassengers)	boolean	Creates an *Bus* object and adds it to the collection of *Bus* objects.
addBusStop (String address, int busID)	boolean	Creates a *BusStop* object and associates it with the *Bus* with the matching ID
getBus(int busID)	*Bus*	Finds and returns the *Bus* object with the given *ID*; if none exists, returns ***null***.
listBuses()	String	Returns a *String* representation of all of the *Bus* objects in the system
listBusStops()	String	Returns a *String* representation of all the *BusStop* objects in the system
listBusStopsByBus(int busID)	String	Returns a *String* representation of all of the *BusStop* objects in the system associated with a particular *Bus*
listBusStopsbyStreet(String address)	String	Returns a *String* representation of all the *BusStop* objects in the system that exist on a particular street

Transit SystemApp: User Interface and Main Class

The user interface must enable the user to perform several operations such as:

- Add a new bus to the system
- Add a new bus stop to the system (bus must exist first)
- Display bus stops by street address
- Display bus stops for a given bus
- Display information about all bus stops in the system
- Display information about all buses in the system

The user interface should accept input from the keyboard and generate textual output to the console. The class, *TransitSystemApplication*, should provide the functionality of the user interface. You should note that the user interface must create an instance of the Bus class before doing anything else. After it receives user input, it forwards requests to the domain classes to accomplish the tasks required. The results are received and displayed on the console.

Implementation Requirements

Accessors and mutators should be provided for attributes of the domain classes as necessary. Use any java collection of your choice. Information hiding must be enforced as much as possible.

Submission Instructions

The code for each class in the application should be written in separate source files as follows:

BusStop class: BusStop.java

Bus class: Bus.java

TransitSystem class: TransitSystem.java

User Interface class: TransitSystemApplication.java