Assignment 1

AMOS MBEKI SCT211-0466/2017

ELIJAH KYULE SCT211-0014/2017

ANNETTE WAITHIRA IRUNGU SCT211-0002/2017

KENNEDY MUIA MWANGANGI SCT211-0202/2017

Jomo Kenyatta University of Agriculture and Technology

Assignment 1

Question 1

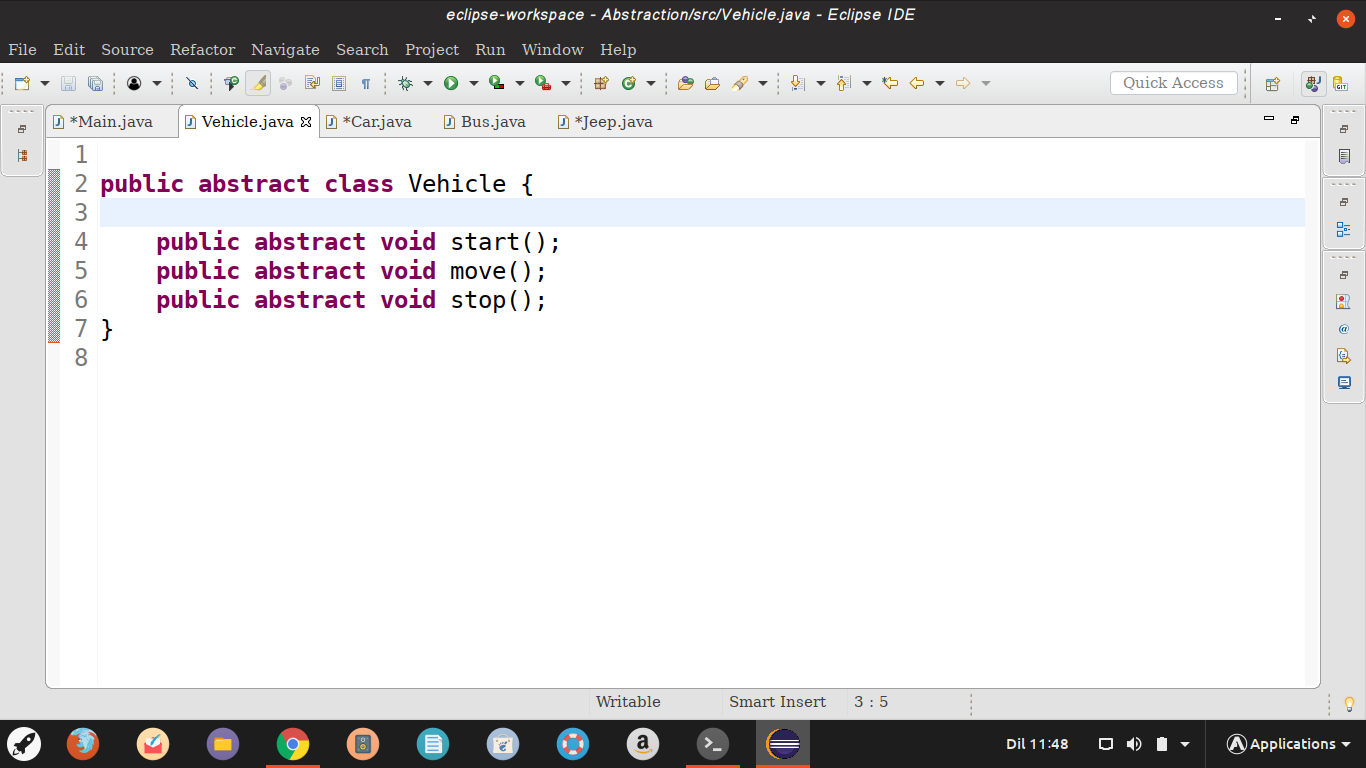
The four major principles of the Object Oriented methodology are data abstraction, polymorphism, encapsulation, and inheritance. Abstract is a concept or an idea which is not associated with any particular instance. Using abstract classes or interfaces, one expresses the intent of the class rather than the actual implementation.

Secondly, polymorphism means one name many forms, and it can be divided into two types - static and dynamic. Static polymorphism is achieved using method overloading and dynamic polymorphism is implemented using method overriding. It is closely related to inheritance. A programmer can write code that works on the superclass and any subclass type as well.

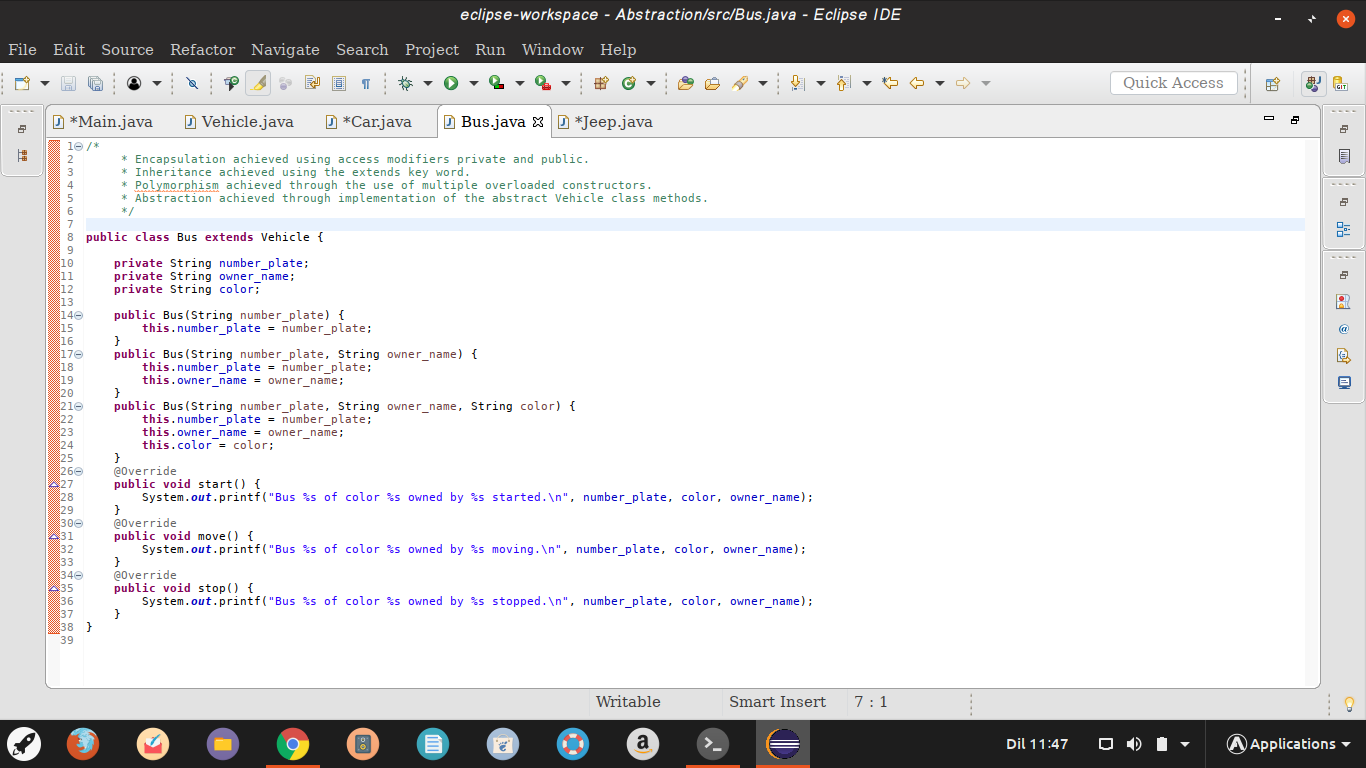
Furthermore, encapsulation is a mechanism of hiding data implementation by restricting access to public methods. Instance variables are kept private and accessor methods are made public to achieve encapsulation.

Lastly, inheritance expresses either an ‘is-a’ or a ‘has-a’ relationship between two objects. Using inheritance in derived classes the programmer can reuse the code of existing super classes. Below are Java code snippets explaining the 4 principles:

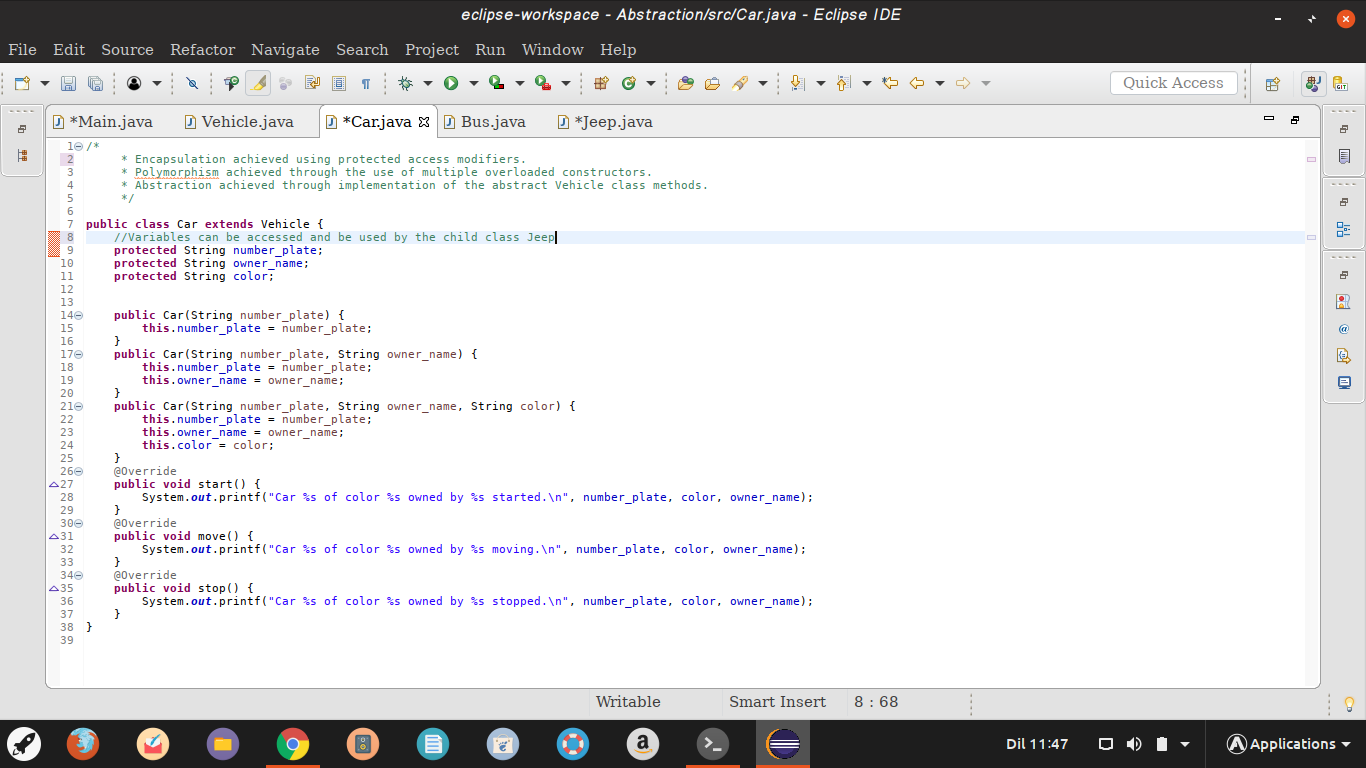
Abstract Vehicle class:



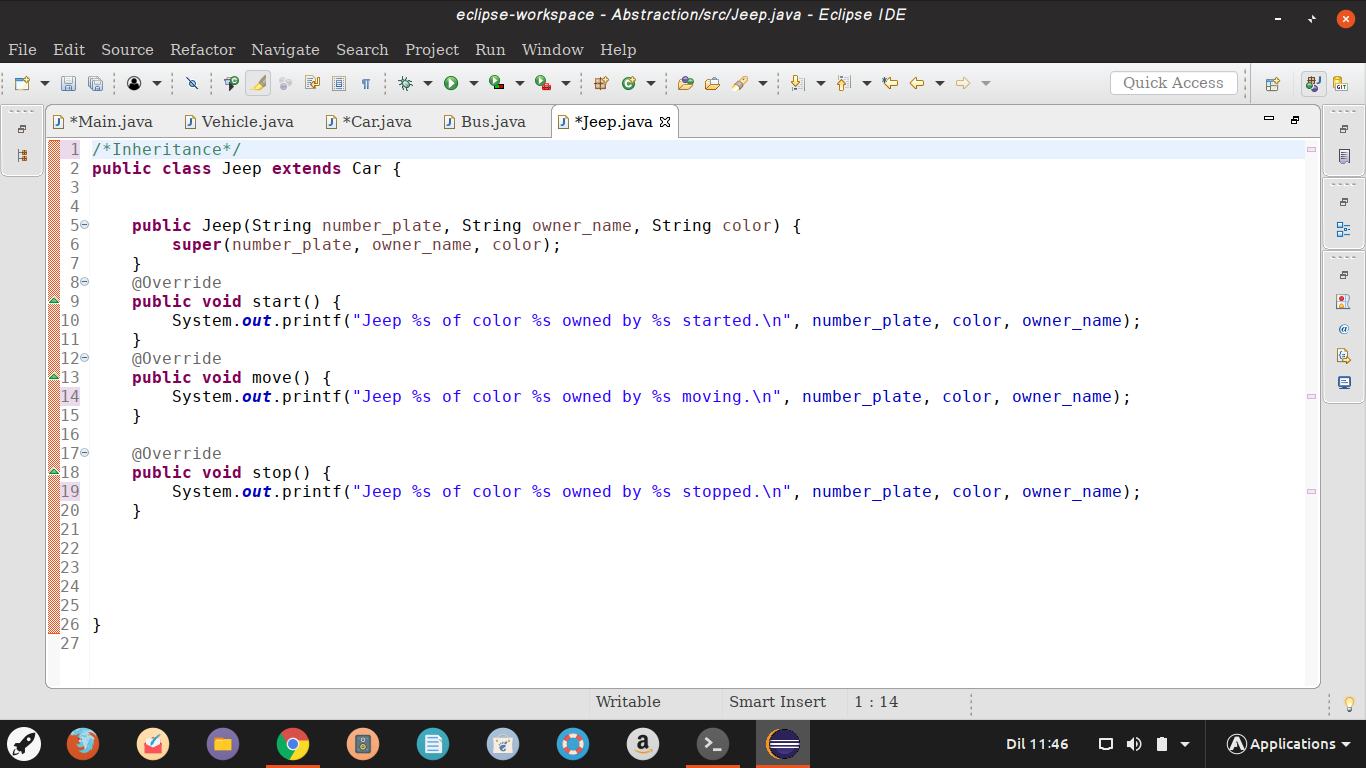
Bus class that extends the abstract vehicle class methods:



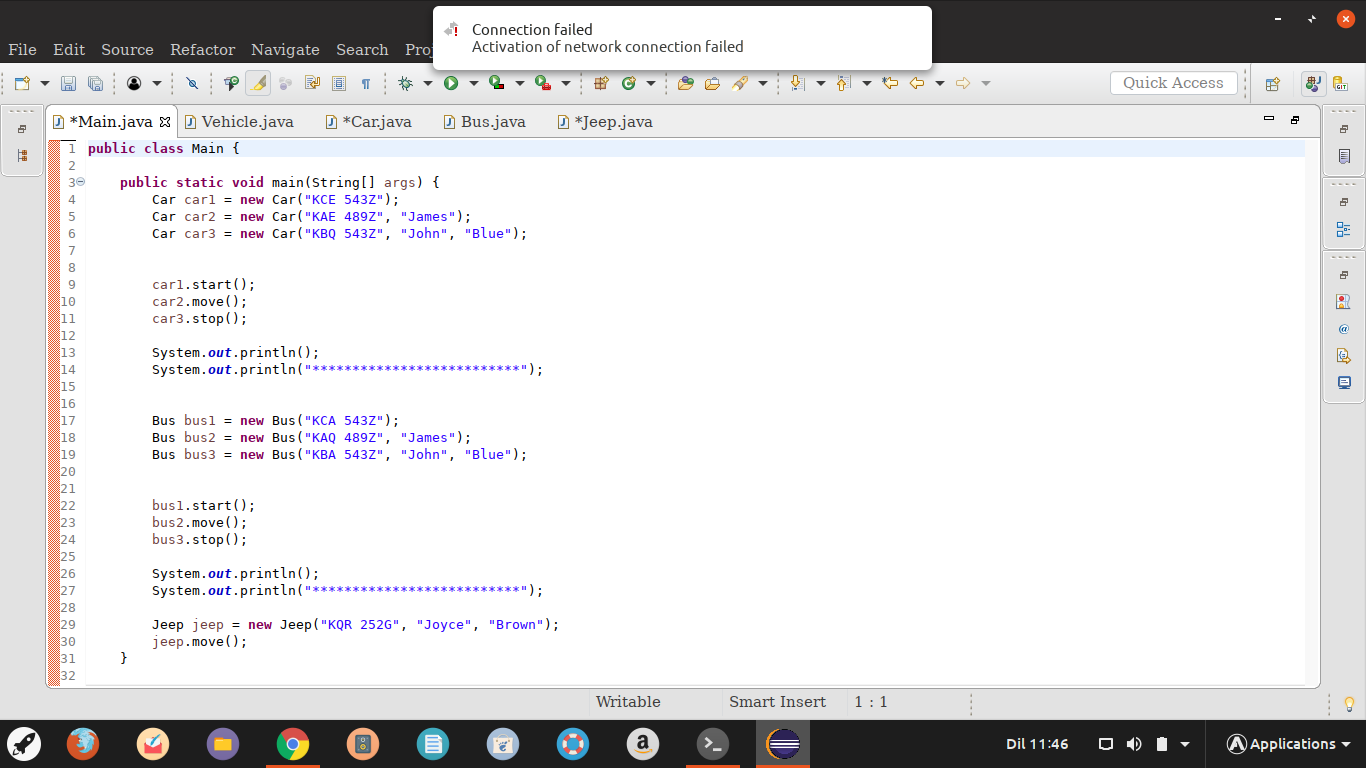
Car class that extends the abstract vehicle class methods and serves as a super class to class Jeep:



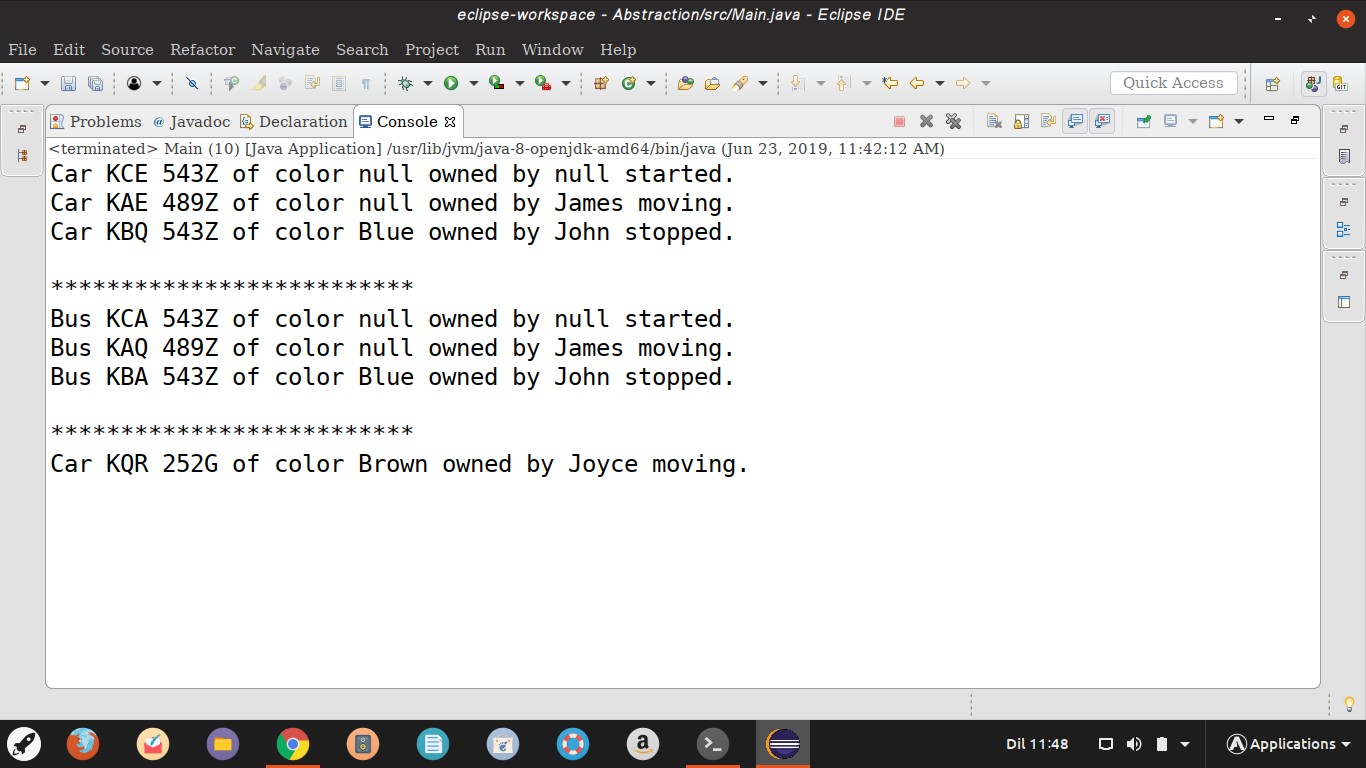
Jeep Class - a sub class to car class:



Main class that creates objects of class Car, Bus and Jeep, and then calls their methods:



The console output:



Question 2

All data flow diagrams include four main elements namely: external entities, processes, data stores, and data flows.

1. External entity.

Also known as an actor, source or link and terminator as it shows how data flows between the entity and the system being represented. It represents the sources of data to the system or destination of data from the system, and is often beyond the area of influence of the developer.

Symbol:

External Entity

1. Process

This is work or actions performed on incoming data flows to produce data flows meaning they show data transformation or change. Data coming into a process must be worked on in various ways, thus, all processes must have inputs and outputs. A major function of a process is computation and decision making.

Symbol:

Process

1. Data flow

It represents the inputs or outputs of data to or from a process by displaying the movement of data between external entities, processes, and data stores. The data flow is represented using an arrow symbol indicating the direction of the flow. The data could be electronic, written or verbal, and input and output data flows are labeled based on the type of data or its associated process or data flow.

Symbol:

Data Flow

1. Data store

A data store is a holding place for information within the system

It is represented by an open ended narrow rectangle

Data stores may be long term files such as sales ledger or may be short term accumulators

Each data flow is given a reference followed by an arbitrary number

Symbol:

Data Store

Question 3

Juja Classifieds level 0 diagram:

Juja Residents

Post

Online Advertising System

Search

Juja Classifieds level 2 diagram:

Sellers

Post Ad

1.0

Central Ads database

Ad Info

2.0

Ads storage & updates

Sellers’ data

3.0

4.0

5.0

Final Ad update

Search results management

Buyers

Buyers’ data

Ads processing

Ads sorting

Processed Ad

Ad Info

Seller Info

Search results

Buyer Info

Sorted data

Question 4

In software and program design, there are numerous methods for analyzing business requirements and developing specifications, and converting practices into computer programs and hardware configuration including structured analysis and structured design. Hierarchical architecture views the whole system as a hierarchy in which the software system is decomposed into logical modules or subsystems at different levels in the hierarchy. This approach is typically used in designing system software such as network protocols and operating systems. A structured chartshows the breakdown of a system to its lowest manageable levels. They are used in structured programming to arrange program modules into a tree. Each module is represented by a box, which contains the module’s name. The tree structure visualizes the relationships between modules.

There are lines that connect the boxes and represent the connection or ownership between activities and sub-activities as they are used in organizational charts. As a design tool, they help the programmer to recursively break down a problem into parts small enough to be understood by the user. The process is called top-down-design,or functional decomposition.During the design stage, the chart is drawn and used as a way for the client and various software designers to communicate. During implementation, the chart is continually referred to as “the master plan.” A structured chart depicts: the size and complexity of the system, the number of readily identifiable functions and modules within each function, and whether each identifiable function is a manageable entity or should be broken down into smaller components.

A structured chart can also be developed by first creating a single structure, which resembles the root of an upside-down tree, which forms the structured chart. The next step is to conceptualize the main sub-tasks that must be performed by the program to solve the problem. Next, the programmer focuses on each sub-task individually, and represents how each can be broken down into even smaller tasks. Eventually, the program is broken down until the leaves of the tree represent simple methods that can be coded with a few program statements. A structured chart is also used to diagram associated elements that comprise a run stream or thread. It is often developed as a hierarchical diagram, but other representations are allowed.

The representation must describe the breakdown of the configuration system into subsystems and the lowest manageable level. An accurate and complete structured chart is the key to the determination of the configuration items, and it acts a visual representation of the configuration system and the internal interfaces among its configuration items. During the configuration control process, the structured chart is used to identify configuration items and their associated artifacts that a proposed change may impact. Lastly, if a structured chart has already been developed, the expert needs to review it to ensure it represents the current structure and if not, update the chart where needed.