INTERNATIONAL CENTRE FOR APPLIED SCIENCES
MAHE, MANIPAL

B.Sc. (Applied Sciences) in Engg.

End – Semester Theory Examinations – Nov./ Dec. 2020

III SEMESTER - SOFTWARE DESIGN USING OBJECT ORIENTED PARADIGM (ICS 233)

(Branch: CS)

Time: 3 Hours **Date: 27 November 2020** **Max. Marks: 50**

- ✓ **Answer ALL the questions.**
- ✓ **Missing data, if any, may be suitably assumed**

1. The scenario of home loan/credit card/automobile buyer application process as described below:

   The home buyer buys a house from the home owner. The home buyer needs to apply for mortgage application to buy a home and this requires assessing the home value, the buyer's credit card rating and final approval from a loan officer. The credit card office will check the credit rating with the information of the home value and finally will be approved by the loan officer. Just like the home buyer, the home owner has to be assessed by the assessor for the home value. The same process holds good for an automobile buyer whose credit history has to be checked before a loan approval by loan officer. The credit card applicant's credit history has to be checked by the Credit Card Bureau like the automobile buyer and the credit card approval given by loan officer.

   a) Write a Use case diagram for the description provided. **[7]**

   b) Explain Noun-Phrase Approach and what are the artifacts generated using this approach. **[3]**

2. The below description is a scenario depicting an issue of a book from library.

   The activities to be completed are accepting the ID, and check the same for verification. Invalid library cards are not entertained and returned back to the student. If the card is valid, the book can be issued and ID card is returned back. The book has to be stamped before it is issued, before handing the book to the member.

   a) Draw a Sequential activity diagram for issuing a book from our Library. **[2]**

   b) Change the above to Swimlane to show the activities of student and librarian. **[3]**

   c) Explain the different components of an activity diagram. **[5]**

3. Model a Package Diagram for the classes listed below.
   Person, Car, Driver, Driving License, Wheel (4), Engine, Cylinder (2, 4, 6, 8).
   a) Show the relationship among the classes and also package them appropriately. **[4]**

   b) Draw a sequence diagram for the below mentioned context.

   In the seq, Student registers for exam using exam number with the StudentAdmin. If there are no vacant seats for exam or if it is full then student can reserve a spot for exam. If the Student is successfully able to register for the exam then the lecturer will invigilate the exam. Whether the Student is able to successfully register for exam or reserve for exam for later date, lecturer provides information of exam to the Student. **[6]**

4. a) How do we want to design the classes with respect to Coupling and Cohesion. Take a single example to demonstrate both coupling and cohesion. **[6]**

   b) Observe the below diagram and explain the design pattern. **[4]**

5. Consider the below scenario.

   The e-commerce revolution has made delivery of several types of products easy. The product could be of different types like perishable goods like food, medicine and non-perishable products like clothes, electronics etc. Also, there are different carriers that are selective about the products they ship; some may not ship certain types of products or they may not have the required permission from the regulatory bodies to ship specific types of packages. Hence, there should be no direct impact on the product package and the carriers who transport these packages.

   a) Identify which design pattern will solve the problem and state why the specific design pattern to be used. **[2]**

   b) Design the classes and interfaces for the above scenario and implement the design pattern for the following problem. Also write the test code to demonstrate the same. **[8]**

   Name the interface *IPackageSender* with a method *transportPackage* and implement 2 classes that uses it.
   Name the abstract class *Package* with a method *sendPackage* and implement 2 classes that creates classes from it.

****