SENG226 – Software Design_ 2024

Midterm Exam, March 29th, 2024, at 11:00.

Duration: 60 minutes.

ATTENTION:

- **Answer all questions yourself without access to lecture notes, PowerPoint presentations, a cellphone, a notebook, a textbook, an AI Tool/software, or help from someone else**.
- Carefully read each question, plan your answer, and then clearly express yourself; keep your answer brief but to the point.
- **Do not write on the question sheet!**
- Answer all questions **in sequence and English only**.

Question 1: (15 p) You are required to develop a weight-watcher application for your diet management so you are considering the design aspects. Note that you will not write a program for this question.

a) How would you approach this problem from an object-oriented point of view? Be brief yet specific.

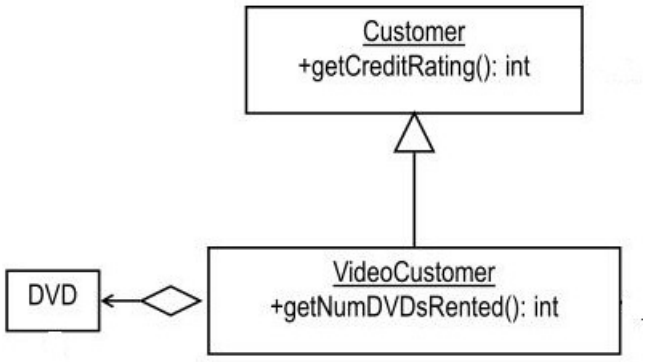
b) If you were to approach this problem design in a classical, non-object-oriented manner, describe how you would begin. Be brief yet specific.

SOLUTION: A la **Exercise 13.2 "Personal Calendar"=> "Weight Watcher"**

a) *Solution outline*: Identify the terms – typically nouns – basic to describing the requirements. Examples are "weight" and "diet."'

b) *Solution outline*: Develop a list of required functional and non-functional features. Example: "Given a particular date, the application shall be capable of displaying a list of diet items consumed and weight for that date.

Question 2: (15 p) Develop the skeleton code for the given class model. Just write the code (Java or otherwise), there is no need to explain.



SOLUTION: **Code from Class Model**

```
class Customer
{
   public int getCreditRating
   {        return 0; }
}
```

```
class DVD
{
}
```

```
class VideoCustomer extends Customer
{
DVD dvd;
public int getNumDVDsRented
{        return 0; }
}
```

Question 3: (15 p) Consider an automated highway entry/exit application (such as a Bosphorus bridge, Adana-Gaziantep-Şanlıurfa Otoyolu) for collecting trip fees at unmanned booths from drivers. The minimum requirement is to store the relevant information when an automobile passes through a booth.

a) For what *purposes* could the design be made flexible? Name a specific, clear purpose not covered elsewhere in this question. Explain your answer.
b) Show the class model of the design of this application relevant to these requirements. The design should be flexible, enabling the handling of cars, trucks, trailers, and other types of vehicles yet to be determined. Justify your design: State in what way it is flexible.

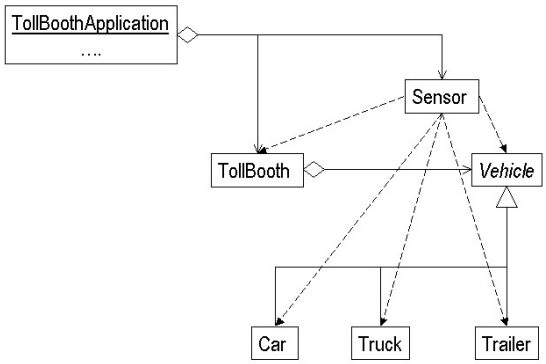
SOLUTION: **Exercise 5.2 "Toll Booth"**

a) A solution:

The application could be required in the future to handle various types of customers differently, compute amounts owed in several different ways, introduce volume discounts, automatically sense the type of vehicle, bill customers in a variety of ways, or display the status on a monitor.

b) A Solution:

TollBooth Solution



Question 4: (20 p) Recall the video store application we examined during the lectures where the main design classes were MainScreen, BarcodeReader, Checkout, Account, and CheckoutOptionDsiplay.

The Checkout use case is given as follows:

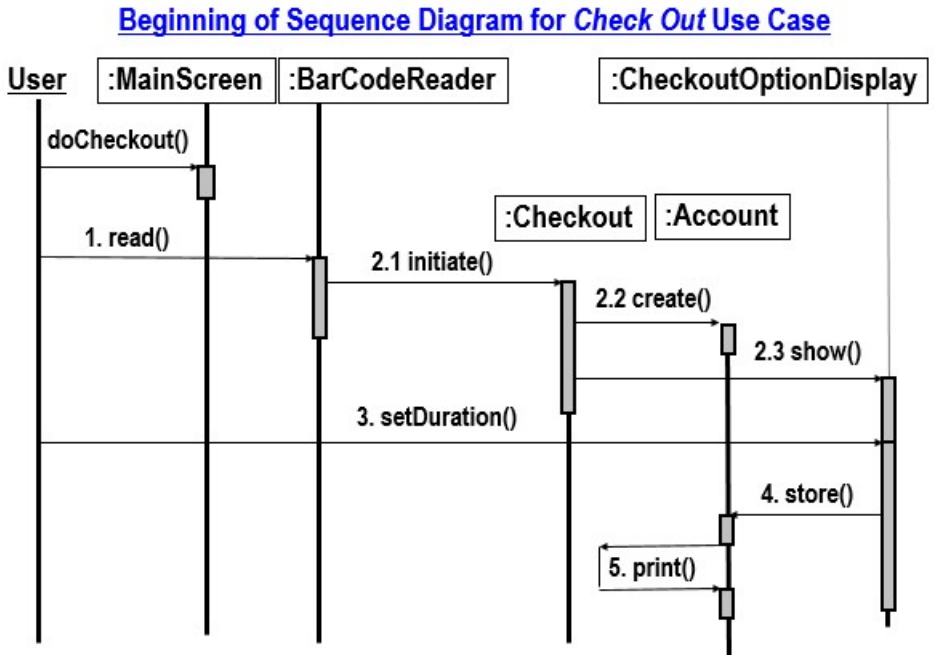
1. User swipes bar code
2. Application prompts for rental duration
3. User enters duration
4. Application stores record
5. Application prints the customer's account status.

Give a sequence diagram for this use case; you may explain your reasoning very briefly if needed.

SOLUTION: **Video Store Checkout Use Case example.**

A solution

The figure shows a sequence diagram for the "Checkout" use case. Note that humans can directly interact with GUI classes only.



Question 5: (15 p) Answer the following questions (5 p each).

a) One of the formal approaches to application correctness is maintaining robustness. Mention the three actions necessary to promote robustness.

b) Name the important aspects of OO useful for application development by mentioning the keywords without defining them.

c) What aspects of the class design you would leverage for the re-use of class combinations?

SOLUTION:

a) The three actions necessary to promote robustness are (i) **Verifying input** (ensuring environmental robustness), (ii) **Initializing** to improve robustness, and (iii) **Parameter passing technique**.

b) The important aspects of OO useful for application development are **Class Introduction**, **Instantiation**, **Inheritance**, and **Polymorphism**.

c) One would try to reduce dependency among classes by **leveraging Inheritance, Aggregation, and Dependency** among classes.

Question 6: (20 p) The following Draw class implements a few interfaces all of which are combined under the single class definition. For a proper reuse environment, one could split these interfaces and thus reorganize the Draw class. Given the Draw class below, design a feasible reorganization by identifying suitable interfaces: name each interface and spell out the functions that would be suitable for accessing through it.

```
class Draw
{   ...
    int setColor( String ) { ... }
    Pen getStandardPen() { ... }
    int getLogoStyle() { ... }
    void setColor( int ) { ... }
    void drawLogo( int, int ) { ... }
    void speedUpPen( int ) { ... }
...
}
```

SOLUTION: Textbook example Fig. 2.19:

Interface Example: a *Draw* Class

- ***Functions dealing with the pen used***
  - Pen getStandardPen()
  - void speedUpPen(int)
  - . . .

  } ***Pen*** **interface**
- ***Functions dealing with the colors available***
  - void setColor(int)
  - int setColor(String)
  - . . .

  } ***Color*** **interface**
- ***Functions covering the drawing of the company's logo***
  - void drawLogo(int, int)
  - int getLogoStyle()
  - . . .

  } ***Logo*** **interface**
- *. . .. . .*