Name of Experiment:

You are going to develop a module of chittagong University ERP (Enterprise Resource Planning) solution for full up form for students final exam. Firstly, students take a form from CSE office and percentage of attendence into the form, if the attendence is less than 70%, the system will create an exception with displaying appropriate message. Then the forms of validated students are submitted to the register office and they will check if the students failed in any previous semester. If do, then generate another exception with appropriate message. Then the form will go to Bank and the student will pay the fees. If student gives less amount than determined fees, then another exception will be ganerated. Then the forms will go to the exam controller office for generating admit card and send the card to see the CSE office. ~~for generating~~ ~~admit~~ Note: Don't worry about the business logic of methods of classes. You can design each method with "System.out.println

Introduction:-

We will have to build a system such that students can take form from CSE office then submit the form to the office. The CSE office, Register, Bank and

Exam Controller should be emulated as class so that the problem can be solved easily.

Objective :-

* to learn how to handle exception.
* to learn how to solve unexpected and undesired state of a program.

Analysis:-

After analysing our problem we have found following components to our solution.

* We have to define classes for all the offical offices to control all the components, like Register office, Accounts, ExamController, Bank, CSE office, Also we have to define the exception classes to handle predefined exception mentioned in the problem statement.

* We will also have to define student and Form named two helper classes.

* a main class that will contain the main method which will simulate the solution of our problem.

From above analysis the conceptual class diagram is given in Figure-1;-

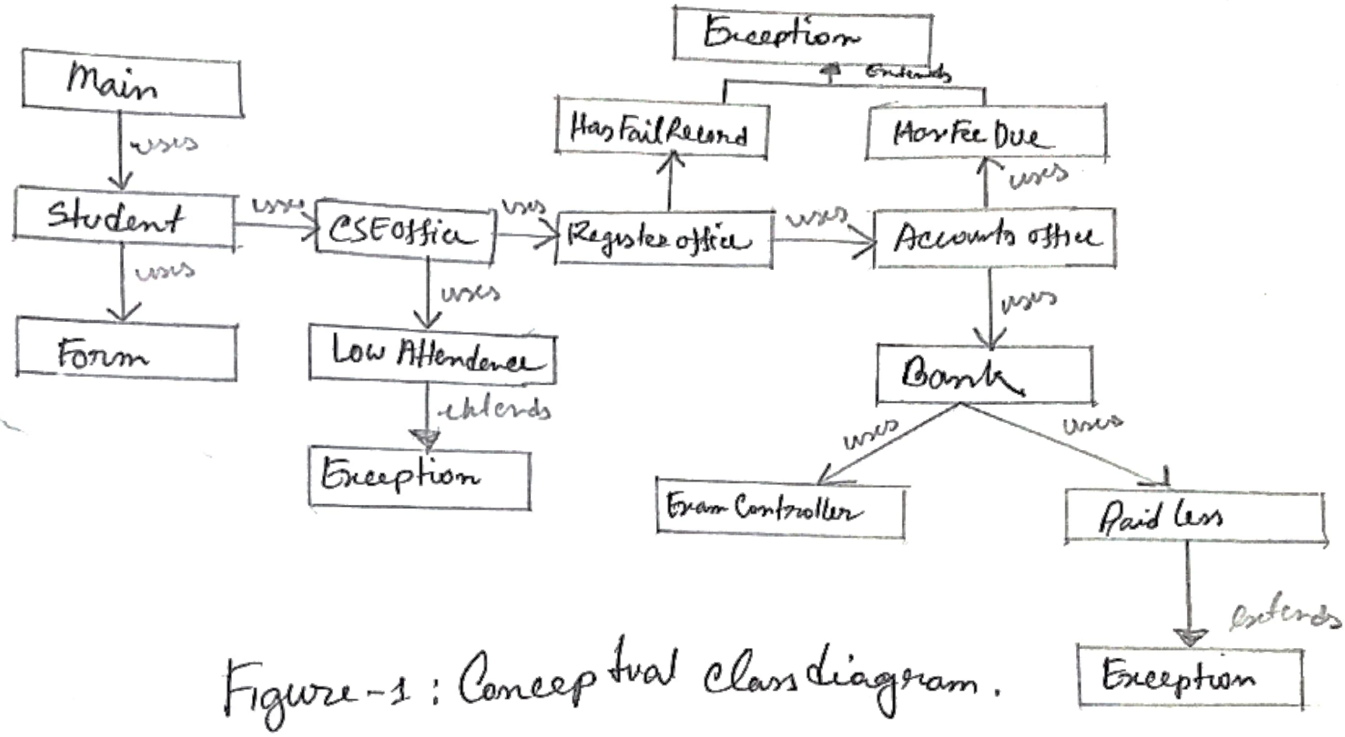

Figure-1 : Conceptual class diagram.

Design:

After analysing our problem the design description of the solution as follows:-

* class student : represent a student.

Data Methods :

* ID : represents student Id
* form : represents the form a student collected
* CSE office : represents the CSE office

Methods :

* collect Form: collects form from the CSE office
* submit form : submit the filled form back to the CSE office

* Class Form :- a helper class that represents form

☐ Data Member:

* ID : represents a student Id.

* Fees : reprents the required fees of a student

☐ Methods:

* setID : sets the Id.

* getID : returns the Id.

* setFees : sets the fees.

* getFees : returns the fees.

* Class CSEOffice :- the class that represents the CSE office.

☐ Data Members:

* regoffice : represents the register office.

☐ Methods :

+ submitForm: receives the forms the students, puts the attendance and sends to the registar office.

* getAttendance :- returns the attendance of students according to their Id.

* class Registar office : the class that represents the registar office.

☐ Data Member :-

* ac office : represents the Accounts office.

☐ Methods :

* submitForm : takes the form of the students and sends to the accounts office.

* checkIfFailed : checks if a student has failed previously or not.

* class Account office : the class that represents the Accounts office.

☐ Data Members :

* bank : represents the bank.

☐ Methods :

* calculateFees : calculates the fees and sends the form with specified fee to the bank

* hasDue : returns true if the student has any due and false otherwise.

* fees : returns the specified fee for a student.

* class Bank : the class that represents the bank

☐ Data Members :-

* econtroller : represents the exam controller

☐ Methods :

* takesFees : takes the fees from the students and sends the form to the exam controller.

* class ExamController : the class that represents the Exam Controller.

☐ Methods :

* generateAdmitCard : genarates admit card for student with given form.

* class LowAttendance : the exeption class that is thrown when a student has attendance lower the 70%

☐ Methods
* to string : returns the low attendence message.

* class HasFailRecord : the exception class that is thrown when a student has previously failed in an exam.

☐ Methods:
* to string : returns the fail record message.

* class HasFeesDue : the exception class that is thrown when a student has only fees due.

☐ Methods:
* to string : returns the has fees due message.

* class PaidLess : the exception class that is thrown when a student pays less than specified fees.

☐ Methods:
* to string : returns the paid less message.

*class Main : the main class of the program that contains the main method.

4 Methods :

*main : the main method of the program.

From above design description the architectural class diagram is given in figure-2:-

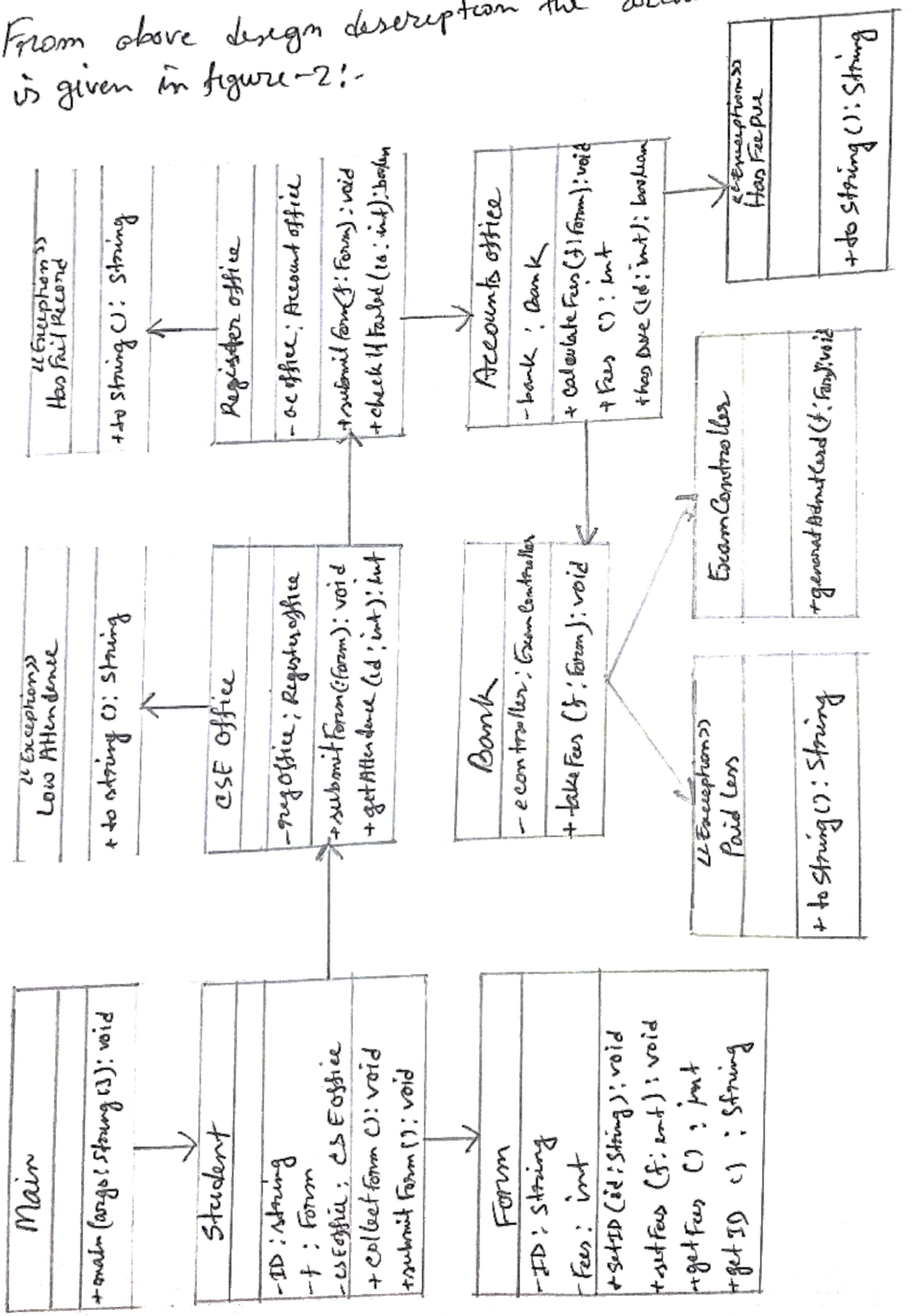

Figure-2: Architectural class diagram.

HasFailRecord :: toString() :
returns the student has failed previously message.

HasFeesDue :: toString() :
returns the student has fees due message.

PaidLess :: toString() :
returns the student has paid less message

Implementation:-

- the implementation is attached with the report.

Conclusion:-

We created all the classes that represents the official offices. We also defined own exception classes and finally defined a main class with main method that simulates the solution.