Yi Lin

Professor Arias

CMPT220

4 May 2018

Project 2 Writeup

Abstract

My database project utilizes three tables: student data table, Marist data table, and location data table. The Student data table stores the academic, financial, and family information of the students. The Marist data table stores the academic and financial requirements for the students to continue college. The location data table stores the students' home town economy information. After setting source data in ODBC data source, the java program will pull data from those table and use students' data as independent variables and location data and Marist data as dependent variables to calculate the possibility for student to continue college. I will use academic performance and financial ability as the measurement for student continuity.

Introduction

My database project creates a program that utilize three tables: student data table, Marist data table, and location data table, to calculate students' retention at the end of the semester. The Student data table stores the academic, financial, and family information of the students. The Marist data table stores the Marist's requirements for the students to continue college. The location data table stores the students' home town economic information. For the users to use this program, they need to follow the user manual. After the user run the program. The program will pull data from the three tables in a database and preform calculation to determine whether the students are academically and financially sustaining at the end of the semester. After the

calculation, the program will print one table for student information and another for family contribution. Using both table, the user can interpret whether students can continue college.

Detailed System Description

As the first step after proposing the project, I began with constructing a database with three table: student data table, Marist data table, and location data table. For accessibility and ease, I decided to use Microsoft Access to construct the database. I filled the student table with fictional student information. Then, I used data from Marist's website to construct the Marist table and used data from census.gov to construct the location table. After completing those tables in the database, I started to write java code to build the connection to the databases. I searched online and found sample java codes that construct the Microsoft connection using a JDBC driver called "ucannaccess". I decided to write a database class that will use the driver to set a connection to the Microsoft Access database. As the UML show, the database class does not contain any data field. Its purpose is merely setting connection to the database. From the database class, I wrote three subclasses: MaristTable, StudentTable, and LocationTable, that will inherit connection and close method and override the pull data method. As the UML shows, each subclass has a private data field that will store the data that was pulled from the table. The LocationTable, however, overload the pull data method. Upon finishing writing the code for those classes, I followed the sample code and created my first prototype, but it prompted an error when running. I went to my colleagues and professor for help and discovered that one of the reference libraries that the driver required was outdated. After I updated the library, the code ran fine. I moved onto writing the program based on the protype. During the process, syntax error on SQL queries in java codes and methods to pull the data have caused much delay. After reformed the code that pulls the data from the table, my next objective was to write the algorithm to

predict the students' GPA at the end of the semester and the possibility that students' families will pay offer cost of attendance if there is any remaining. To predict the students' GPA, the program would use the class performance of the students and the number of the credit for that class. First, it multiplies the current grade of the class and the credit of the class for every class that student has registered. Second, it sums the products and then divides the sum by the total amount of credit the student registered. Third, after dividing the sum, the program adds the result to the current GPA of the students and divide the result by two. If the student does not have a GPA, GPA equal to zero, then the result of dividing the sum will be the predicted GPA of the student. After predicting the GPA of the student, the if-statement will check the GPA against the minimum GPA require for the student to avoid academic probation. If the predicted GPA is less than the minimum, then the amount of GPA needed will be calculated by subtracting the required GPA by the predicted GPA. To calculate a student's family's financial ability to contribute to the cost of attending, I researched the cost of living in the city where the student lives. Then the program uses the cost of living as filter to determine at what level of income the family will contribute to the cost of attendance. The filtered result will determine the possibility of family contribution for each income level and the total possibility of family contribution in that local area. Next, the program determines the difference between the filtered level of income and the cost of living. The difference will be the maximum amount that a family of that range of income can contribute. After finishing predicting the GPA and calculating the family's contribution, the program will print them in as two tables per student. Initially, I thought of printing the result as sentences rather than tables. For a better view of the data, I decided to print the data as two tables. The first table will have students' information, total possibility of family contribution, and

the amount of money and GPA a student needs to continue school. The second table will print the student's family contribution and the possibility in according to each income level.

Requirements

This program will need to pull data from the database, perform calculations with this data, and print out the outcome as tables for the user to understand whether the student can continue college at the end of the semester. This program will require a database; a java superclass that builds the connection to the database; three java subclasses that will pull data from three different tables in the database; and a main program that will perform calculation utilizing data that the three subclasses pulled and print two table. One table will display the students' information, total possibility of family contribution, and the amount of money and GPA the students need to continue school. Another table will display the students' family contribution and the possibility in according to each income level.

Literature Survey

My determination to work on database project was inspired from the project that my supervisor and colleague are working on in their course. Their project focuses on student retention, but unlike my database project, they consider various factors that allow students to continue their academic careers. They consider not only academic performance and financial ability, but also psychological factors that cause students to dropout, travel distance that causes students to transfer, social issues that students face at Marist College, and opportunities that Marist offers. After listening to their discussion, I wanted to do similar things for my database project.

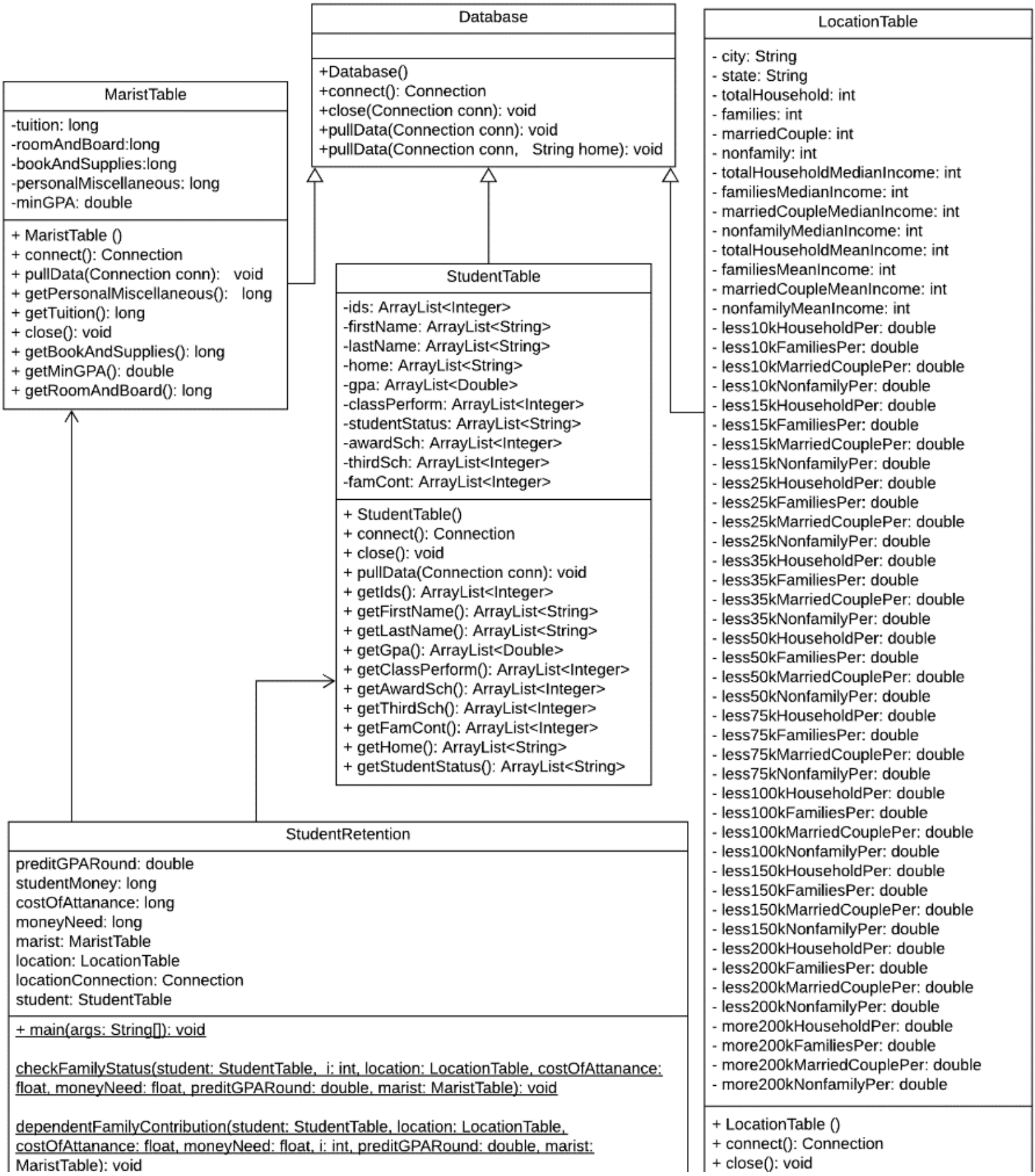
User Manual

To have the program displace both tables for each student, the user must enter the data into the database with appropriate value. For the student data table, the user should enter student information. For the Marist data table, the user should enter the latest Marist data. For the location data table, the user should enter the newest data of the location where students live. After updating the database, the user should enter the database as source data in system DNS in the ODBC Data Sources. This protocol allows the database to be connected to ODBC data source. It is a simple process. First, the user needs to select "Add" in "System Data Source" tab in ODBC Data Sources. Second, the user needs to choose "Microsoft Access Driver" and select finish. Third, the user needs to give a name, and select the database to connect to. After finished the steps, the user may select "Ok." Those are the steps for setting the connection between the Microsoft Access Driver and the database, so the java program can build the connection to the database. After finishing the steps and having the database and the program ready, the user may run the program. The console should print two tables for each student.

Conclusion

Utilizing both tables, the user can better understand of the possibility for a student to continue school at the end of the semester from the perspective of any point in the semester. The best time to use this program is in the mid-semester when midterm grades are being posted for students. Mid-semester has enough information that can be used to predict a student's performance at the end of the semester. The mid-semester is also a good time for students to start making changes to help them continue college. This program will allow the user to understand the students' need to continue college from academic and financial perspective. I hope that by accomplishing the project I will gain more experience and skills in writing code regarding java and database.

UML


Database
+Database()
+connect(): Connection
+close(Connection conn): void
+pullData(Connection conn): void
+pullData(Connection conn, String home): void
MaristTable
-tuition: long
-roomAndBoard:long
-bookAndSupplies:long
-personalMiscellaneous: long
-minGPA: double
+ MaristTable ()
+ connect(): Connection
+ pullData(Connection conn): void
+ getPersonalMiscellaneous(): long
+ getTuition(): long
+ close(): void
+ getBookAndSupplies(): long
+ getMinGPA(): double
+ getRoomAndBoard(): long
StudentTable
-ids: ArrayList<Integer>
-firstName: ArrayList<String>
-lastName: ArrayList<String>
-home: ArrayList<String>
-gpa: ArrayList<Double>
-classPerform: ArrayList<Integer>
-studentStatus: ArrayList<String>
-awardSch: ArrayList<Integer>
-thirdSch: ArrayList<Integer>
-famCont: ArrayList<Integer>
+ StudentTable()
+ connect(): Connection
+ close(): void
+ pullData(Connection conn): void
+ getIds(): ArrayList<Integer>
+ getFirstName(): ArrayList<String>
+ getLastName(): ArrayList<String>
+ getGpa(): ArrayList<Double>
+ getClassPerform(): ArrayList<Integer>
+ getAwardSch(): ArrayList<Integer>
+ getThirdSch(): ArrayList<Integer>
+ getFamCont(): ArrayList<Integer>
+ getHome(): ArrayList<String>
+ getStudentStatus(): ArrayList<String>
LocationTable
- city: String
- state: String
- totalHousehold: int
- families: int
- marriedCouple: int
- nonfamily: int
- totalHouseholdMedianIncome: int
- familiesMedianIncome: int
- marriedCoupleMedianIncome: int
- nonfamilyMedianIncome: int
- totalHouseholdMeanIncome: int
- familiesMeanIncome: int
- marriedCoupleMeanIncome: int
- nonfamilyMeanIncome: int
- less10kHouseholdPer: double
- less10kFamiliesPer: double
- less10kMarriedCouplePer: double
- less10kNonfamilyPer: double
- less15kHouseholdPer: double
- less15kFamiliesPer: double
- less15kMarriedCouplePer: double
- less15kNonfamilyPer: double
- less25kHouseholdPer: double
- less25kFamiliesPer: double
- less25kMarriedCouplePer: double
- less25kNonfamilyPer: double
- less35kHouseholdPer: double
- less35kFamiliesPer: double
- less35kMarriedCouplePer: double
- less35kNonfamilyPer: double
- less50kHouseholdPer: double
- less50kFamiliesPer: double
- less50kMarriedCouplePer: double
- less50kNonfamilyPer: double
- less75kHouseholdPer: double
- less75kFamiliesPer: double
- less75kMarriedCouplePer: double
- less75kNonfamilyPer: double
- less100kHouseholdPer: double
- less100kFamiliesPer: double
- less100kMarriedCouplePer: double
- less100kNonfamilyPer: double
- less150kHouseholdPer: double
- less150kFamiliesPer: double
- less150kMarriedCouplePer: double
- less150kNonfamilyPer: double
- less200kHouseholdPer: double
- less200kFamiliesPer: double
- less200kMarriedCouplePer: double
- less200kNonfamilyPer: double
- more200kHouseholdPer: double
- more200kFamiliesPer: double
- more200kMarriedCouplePer: double
- more200kNonfamilyPer: double
+ LocationTable ()
+ connect(): Connection
+ close(): void
StudentRetention
preditGPARound: double
studentMoney: long
costOfAttanance: long
moneyNeed: long
marist: MaristTable
location: LocationTable
locationConnection: Connection
student: StudentTable
+ main(args: String[]): void
checkFamilyStatus(student: StudentTable, i: int, location: LocationTable, costOfAttanance: float, moneyNeed: float, preditGPARound: double, marist: MaristTable): void
dependentFamilyContribution(student: StudentTable, location: LocationTable, costOfAttanance: float, moneyNeed: float, i: int, preditGPARound: double, marist: MaristTable): void


marriedFamilyContribution(student: StudentTable, location: LocationTable, costOfAttanance: float, moneyNeed: float, i: int, preditGPARound: double, marist: MaristTable): void
singleFamilyContriubution(student: StudentTable, location: LocationTable, costOfAttanance: float, moneyNeed: float, i: int, preditGPARound: double, marist: MaristTable): void
predictGPA(i: int, student: StudentTable): double
checkGPA(preditGPARound: double, marist: MaristTable): void
getGPA4PointScale(grade: int): double
+ pullData(Connection conn): void
+ getCity(): String
+ getState(): String
+ getTotalHousehold(): int
+ getFamilies(): int
+ getMarriedCouple(): int
+ getNonfamily(): int
+ getTotalHouseholdMedianIncome(): int
+ getFamiliesMedianIncome(): int
+ getMarriedCoupleMedianIncome(): int
+ getNonfamilyMedianIncome(): int
+ getTotalHouseholdMeanIncome(): int
+ getFamiliesMeanIncome(): int
+ getMarriedCoupleMeanIncome(): int
+ getNonfamilyMeanIncome(): int
+ getLess10kFamiliesPer(): double
+ getLess10kHouseholdPer(): double
+ getLess10kMarriedCouplePer(): double
+ getLess10kNonfamilyPer(): double
+ getLess15kHouseholdPer(): double
+ getLess15kFamiliesPer(): double
+ getLess15kMarriedCouplePer(): double
+ getLess15kNonfamilyPer(): double
+ getLess25kHouseholdPer(): double
+ getLess25kFamiliesPer(): double
+ getLess25kMarriedCouplePer(): double
+ getLess25kNonfamilyPer(): double
+ getLess35kHouseholdPer(): double
+ getLess35kFamiliesPer(): double
+ getLess35kMarriedCouplePer(): double
+ getLess35kNonfamilyPer(): double
+ getLess50kHouseholdPer(): double
+ getLess50kFamiliesPer(): double
+ getLess50kMarriedCouplePer(): double
+ getLess50kNonfamilyPer(): double
+ getLess75kHouseholdPer(): double
+ getLess75kFamiliesPer(): double
+ getLess75kMarriedCouplePer(): double
+ getLess75kNonfamilyPer(): double
+ getLess100kHouseholdPer(): double
+ getLess100kFamiliesPer(): double
+ getLess100kMarriedCouplePer(): double
+ getLess100kNonfamilyPer(): double
+ getLess150kHouseholdPer(): double
+ getLess150kFamiliesPer(): double
+ getLess150kMarriedCouplePer(): double
+ getLess150kNonfamilyPer(): double
+ getLess200kHouseholdPer(): double
+ getLess200kFamiliesPer(): double
+ getLess200kMarriedCouplePer(): double
+ getLess200kNonfamilyPer(): double
+ getMore200kHouseholdPer(): double
+ getMore200kFamiliesPer(): double
+ getMore200kMarriedCouplePer(): double
+ getMore200kNonfamilyPer(): double

Reference

Data Access and Dissemination Systems (DADS), Census Bureau's Population Estimates Program. “Income in The Past 12 Months (In 2016 Inflation-Adjusted Dollars).” *American FactFinder*, U.S. Census Bureau, 2012-2016, https://factfinder.census.gov/faces/tableservices/jsf/pages/productview.xhtml?src=CF.

DePersio, Greg. “How Much Money Do You Need to Live in Miami?” Investopedia, Investopedia, 8 Sept. 2015, www.investopedia.com/articles/personal-finance/090815/how-much-money-do-you-need-live-miami.asp. Accessed 28 March 2018

DePersio, Greg. “How Much Money Do You Need to Live in NYC?” Investopedia, Investopedia, 9 Mar. 2018, www.investopedia.com/articles/personal-finance/092415/how-much-money-do-you-need-live-nyc.asp. Accessed 28 March 2018

Imran, Mohd. “Java Database Connectivity Using MS Access - Part-1(Insertion or Register).” YouTube, YouTube, 31 July 2013, www.youtube.com/watch?v=t1SfzP3LlcE&t=46s. Accessed 2 April 2018

Saleem, Aamir, director. Java Program to Access the MS Access Database with Ucanaccess Library with Java 8 or Above. Youtube, 2 Dec. 2016, www.youtube.com/watch?v=Zb1AWUroicM. Accessed 2 April 2018

SJ. “Java JDBC: An Example to Connect MS Access Database.” BenchResources.Net, 24 Apr. 2017, www.benchresources.net/jdbc-msaccess-database-connection-steps/. Accessed 2 April 2018