**IST 412: Complex Software Systems**

**Term: Summer 2020**

# **Topic: Project Module and API Design**

**Pennsylvania State University: World Campus**

**Team 1: Sabahat Aziz, Albana Beqo, Ryan Carey, Dave Blon**



# Class Cohesion Assesment

**Module: Demo**

Class: MFinanceDriver.java

LCOM4 = 1

beginApp() - MenuChoice1/MenuChoice2/MenuChoice3 – scanner.next() - username1/userrname2/username3 – scanner.next() - password – scanner.next() - credentialCheck()- output1 – displayEmployeeOption()

beginApp() - MenuChoice1/MenuChoice2/MenuChoice3 – scanner.next() - username1/userrname2/username3 – scanner.next() - password – scanner.next() - credentialCheck()- output2 – System.out.println()

*Class Cohesion*

The MFinanceDriver class is the controller class that houses the main actions needed to run this program so far. This class represents a combination of an iterative control cohesion and a conditional control relation, where an inner loop gathers input to collect credentials, checks those credentials against the README.txt file, and provides the correct output. Then it repeats the process, and depending on the input, there could be only one of the three possible outputs. This is a more complicated class, however the logic of the code allows for the processing of multiple inputs and providing a single output at any given time.

**Module: Main**

Class: Main.java

LCOM4 = 1

MFinanceDriver. BeginApp()

*Class Cohesion*

The main class creates a new instance of the MFinanceDriver class and calls it app. It then calls the beginApp method in that class. This is inherently a cohesive class as it’s only purpose is to call the controller class (demo here) that houses the method needed to run the program.

**Module: Model**

Class: Address.java

LCOM4= 1

getStreet() - getCity() - getState() - getZip() -toString()

*Class Cohesion*

The address class has a set of attributes that comprise the address of the customer. The attributes have been set as private and getters and setters have been created in order to enable encapsulation and data protection. A toString method has been created to gather the information into one output, which is the full address. The output of this class is then used in other classes such as the customer class below.

Class: Customer.java

LCOM4=1

GetFirstName() - getLastName() - getMiddleName() - getBirthday() - getAddress() - getPhoneNumber() - getId() - toString()

*Class Cohesion*

The customer class is also a basic class with all the necessary getters and setters for data protection and encapsulation. All the attributes and the input from the Address class are used to build the toString method that incorporates all the customer data. The output is a complete customer profile. There are several inputs and one output – the customer.

Class: Employee.java

LCOM4= 1

GetFirstName() - getLastName() - getMiddleName() - getFullName() - getId() - employeeInfo()

*Class Cohesion*

The employee class is also a good class with respect to cohesion. For this class the same principles of encapsulation have been applied and the first three attributes, firtName, lastName and middleName are used as input to the getFullName() method. Then the getFullName method and the id attribute are used as input for the employeeInfo method. Therefore, all attributes and methods are linked and there is one output. This is a cohesive class.

Class: FinancialProfile.java

LCOM4=1

LoanId - loanType – LoanStatus (hardcoded for now) - loanAmount - requestLoan()

*Class Cohesion*

The FinancialProfile class serves to combine several classes to complete a full loan profile. This class creates a new instance of the Customer class, it takes attributes from the LoanRequest class, specifically the loanType attribute and the lonaId attribute, and it combines all the data from these individual classes to form a single output, which is a complete record of the loan produced by the requestLoan method. The new customer created is not fully incorporated in this method yet, but as the code evolves, the intention will be to tie in the loan to the client that it belongs to.

Class: Identifier.java

LCOM4=1

Generate() - setId()- getId() - toString()

*Class Cohesion*

The Identifier class is also a simple class, which has one attribute, the id. The class relies on the java utility method UUID.randomUUID() to generate a random id. The getters and setters encapsulate the id and then the output is passed to the toString method. The randomly generated id is the only output for this class, which means that the class is cohesive.

Class: LoanHistory.java

LCOM4=1

INSTANCE – loanHistory() -add() - getAll()- clear()

*Class Cohesion*

The loanHistory class is also rather cohesive. It serves to collect loanRequest data. The output from the loanRequest class serves as the input for this class. The class builds a LoanHistory array list which is populated with data from the loanRequest, labled INSTANCE in this class. The class has methods to retrieve a specific loanRequest or all the loanRequests that have been added to the array list. Another method is used to clear the request history and remove the loanRequests from the array list. All the methods in this class use the same attribute: INSTANCE which is stored into the loanHistory array list. They use it as an individual input or as part of the larger input which is the array list.

Class: LoanRequest.java

LCOM4=1

GetId() - getLoanType() - getLoanSatus() - getDateTimeCreated() - getAmount()- toString()

*Class Cohesion*

The LoanRequest class is also a very cohesive class. This class uses the output of the identifier class as one of the inputs. It also relies on the java utility class java.time.LocalDateTime. All the inputs listed above are combined to create a loan request toString method that refers to a single loan request output. This class also meets the cohesion criteria and has LCOM4 equal to 1.

**Module: Util**

Class: CredentialCheck.java

LCOM4=1

String username, String password, verifyCredentials method

*Class Cohesion*

The CredentialCheck class is very important, especially for the user applying for a loan. This class will verify the criteria of the user. By using a private final README method, it will have access to information regarding the user. In this class, you can only verify your credentials after inserting your username and password. If the username or password don’t match, a Boolean will let the system know the information you typed in is wrong, therefore it will return false, as in an invalid credential, and you won’t have access to your file. Once you type in the correct username and password data, your credentials will be verified, and you can have access. CredentialCheck has common inputs, common outputs, and common methods so having that data and information makes this class cohesive.

Class: DoubleUtil.java

LCOM4=1

TryParseDouble method, return Double.parseDouble output

*Class Cohesion*

In the DoubleUtil class, there is a tryParseDouble method, that returns a new double representing the value string Text that we have. This class is used for holding large numbers, especially decimal numbers and will come in handy while running reports. Due to the relationship with the numbers and the common methods, this class is cohesive.

Class: StringUtil.java

LCOM4=1

boolean isNullOrEmpty, boolean isNullOrWhitespace, return outputs

*Class Cohesion*

The StringUtil class will provide null-safe methods for handling the Strings in our codes. This is an error-prone approach to making sure we don’t run into issues while handling any of our strings. This class contains a isNullorEmpty method to let us know if our strings are either null or empty, and there is also a isNullorWhiteSpace which checks not only if the strings are null and empty, but also if they consist of whitespace characters. This class also has LCOM requirements that equal to 1, therefore being a cohesive class.