Module 1 – Apply – Introduction to Software Development

Benjamin Roberts

Level 4 Software Development

Contents

[Introduction 3](#_Toc166859658)

[Project Description 3](#_Toc166859659)

[How this was decided 3](#_Toc166859660)

[Acceptance Criteria/Project Outcomes 3](#_Toc166859661)

[Stakeholders Involved 4](#_Toc166859662)

[Summary 4](#_Toc166859663)

[QA 4](#_Toc166859664)

[Code Repository 5](#_Toc166859665)

[User Input 5](#_Toc166859666)

[CRUD - Database Connection 7](#_Toc166859667)

[Firewall 7](#_Toc166859668)

[Credit Policy built 8](#_Toc166859669)

[Scorecards 8](#_Toc166859670)

[Characteristics 8](#_Toc166859671)

[Champion Specification 8](#_Toc166859672)

[Challenger Specification 11](#_Toc166859673)

[NULL Handling in characteristics 12](#_Toc166859674)

[Probability of Default formula 12](#_Toc166859675)

[Policy Rules 12](#_Toc166859676)

[Testing the program 13](#_Toc166859677)

[JUnit 13](#_Toc166859678)

[End to End 13](#_Toc166859679)

[References 14](#_Toc166859680)

[Appendix 16](#_Toc166859681)

[Resources: 16](#_Toc166859682)

[Project Brief & Checklist 16](#_Toc166859683)

[Lessons learned 17](#_Toc166859684)

[Learning Criteria: 17](#_Toc166859685)

# Introduction

## Project Description

Create a Java program that implements Credit Decisioning logic at an introductory level. This must include:

1. Some ability to capture relevant applicant information required to make a credit decision and enter this information into the program
2. Execute credit decisioning policy rules that conditionally reject an application if set criteria is met
3. Connect to an external database to retrieve more data to use for decisioning
4. Use additional data gathered from this database to execute Scorecards in a champion/challenger approach (requiring at least 2 scorecards to be created)
5. Return a final decision and an offer to the user depending on the decisioning logic
6. A record of the program run is then saved into an external database (for monitoring and further credit decisioning use)

## How this was decided

The team I work in is primarily responsible for developing and maintaining a Credit Decisioning software platform. The software is provided by a 3rd party and provides a low-code graphic user interface for developing business logic into a callable Web service without the need for specialised IT support.

The Java elements of the Module 1 course demonstrated an opportunity to apply similar business logic into a more technical language, in a role relevant manner.

## Acceptance Criteria/Project Outcomes

1. Final program must have an interface for a user to input application data
2. Application data input must include:
   1. A Match Id of some kind to look up in the external database
   2. Residential Status
   3. Employment Status
3. A database connection is made to allow CRUD operations for logging activity made in the program
4. A database connection is made to allow a lookup to be made based on the data entered on the Application form
5. Multiple credit decisioning policy rules are built and executed in the program flow
   1. Policy rules are groups in a modular way to breakdown testing
   2. A mix of the application input data and data retrieved from the external database is used
6. Multiple credit decisioning scorecards are built and executed
   1. A credit offer is made in a champion/challenger style i.e. only 1 is used as the basis for the offer despite both being executed

# Stakeholders Involved

## Summary

|  |  |
| --- | --- |
| **Stakeholder** | **Input** |
| Credit Risk | * Sign-off for project with Line Manager * Consult for specific business logic that could be applied to the project & would be appropriate to share |
| IT Delivery Managers | * Consulted for advice on how to host the application, specifically for the database link for CRUD operations |
| Asset Management | * Arranged the installation of all software and licenses required for this project, including:   + Eclipse IDE   + JDBC Driver   + Visual Studio 2022 subscription with Microsoft Azure credits |
| Data Delivery | Consulted for:   * advice on setting up an Azure SQL Database user * importing data into a database that could be used for lookup operations |
| QA | * Consulted my DLC for information on Maven Projects to build |
|  |  |
|  |  |

## QA

# Code Repository

All code developed for this project can be found in the below GitHub Repository:

[LVL\_4\_Software\_Dev\_Repo/M1 - Intro to Software Development/Module 1 Apply Project at main · broberts-gtw/LVL\_4\_Software\_Dev\_Repo (github.com)](https://github.com/broberts-gtw/LVL_4_Software_Dev_Repo/tree/main/M1%20-%20Intro%20to%20Software%20Development/Module%201%20Apply%20Project)

# User Input

The Java Scanner class was used to prompt the user for input. Defined below is a class called “UserInput” which has been written to group all methods related to capturing user input together:

A screenshot of a computer program

Description automatically generated

**Figure**: Code used to define the “UserInput” class, importing the necessary Scanner class & defining

Upon running the program, the user will be asked a series of questions:

1. Their Forename
2. Surname
3. The ResidentialStatus and their EmploymentStatus, presented as a menu. An example of the code for this is annotated below:

A screen shot of a computer program

Description automatically generated

* “println()” functions print out a menu of options to the console for the user
* A custom defined “getIntInput” method calls the Java Scanner & writes out another prompt to the console with instructions to the user
* A “while” loop traps the user in this loop until they provide a valid option
* The “try-catch” block is included to capture invalid characters e.g. string values when expecting a value of 1 to 6
* The “loopCount” increment logic is included as a failsafe for ending the while loop in case of too many bad inputs and/or preventing an infinite loop

# CRUD - Database Connection

## Firewall

A screenshot of a computer

Description automatically generated

Caption: A screenshot showing a Firewall rule in place on my database to only allow access from my network (with IP Addresses redacted)

# Credit Policy built

The below section details the Credit Decisioning Logic built into the program.

## Scorecards

Credit Scorecards are a tool used by Financial Risk Management teams to model the likelihood of default for a given subject (i.e an applicant/customer) over a period of time (Huang and Scott, 2017). Being a critical part of Credit Decision made them a natural choice to build in this project.

This project implements 2 scorecards in a champion/challenger fashion. This is an approach that deploys multiple competing strategies into a production environment and monitoring which provides the best performance (FICO, 2020). Using this is an effective demonstration of using Object-Oriented programming to build Credit Decisioning logic.

Note that both scorecards are not based on any actual statistical model. They are only for demonstrating how the concept can be implemented in an Object-Oriented program.

### Characteristics

These scorecards consist of a starting/base score plus a series of characteristics, variables with values that represent a statistical insight into the applicant/customer e.g. the E1A09 characteristic in the Champion scorecard count how many delinquent (i.e. in arrears) account they have present on their account (Experian, 2024). Specific values in that variable get assigned Banding Scores (or Points) which represent if they are a positive or negative indication of the applicant’s credit worthiness.

Once calculated, all the Banding scores are added together, along with an initial starting/base score to calculate a final score for the applicant, which represents their credit worthiness. A financial institution can use this as a basis for deciding a credit offer i.e. do they offer them a product and what terms they will offer if they do.

### Champion Specification

Below is a specification of the Champion scorecard built into the program, documented in a style typical in the workplace. The columns mean:

* **Characteristic** = the name of the variable used. Can reference ether data the applicant enters on an application form or data available on their Credit records accessed via Credit Bureaus like Experian
* **Data Type –** documents the type of values the characteristic can hold. Useful for guiding developeers in how to handle the values within
* **Min Value & Max Value –** These relate to the **Points** column & defines the ranges used to set a specific Banding Score/Points. For String values, this is a 1-to-1 mapping for value to points
* **Points –** The actual Banding score a

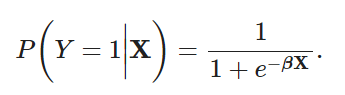
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Characteristic** | **Data Type** | **Min Value** | **Max Value** | **Points** | **Notes** |
| Base Score | Numeric | [default] | | 600 | This is a static value that all instances of this scorecard will start from. |
| E1B07 | String | T | | 0 | Worst Status in the last 6 months of all Active CAIS accounts (SP) |
| N | | 0 |  |
| D | | 0 |  |
| U | | 0 |  |
| 0 | | 27 |  |
| 1 | | 14 |  |
| 2 | | -10 |  |
| 3 | | -23 |  |
| 4 | | -23 |  |
| 5 | | -23 |  |
| 6 | | -23 |  |
| 8 | | -50 |  |
| [default] | | -50 |  |
| E1B09 | Numeric | -1 | 0 | -20 | Number of Active CAIS Accounts (SP) |
| 1 | 2 | 10 |  |
| 3 | 4 | 20 |  |
| 5 | 6 | 30 |  |
| 7 | max | 18 |  |
| [default] | | -20 |  |
| TRD-A-13 | Numeric | -3 | -3 | 40 | Number of Months Since Most Recent Arrears on all Accounts allowed |
| -2 | -2 | -20 |  |
| -1 | 0 | -20 |  |
| 1 | 1 | 10 |  |
| 2 | 2 | 5 |  |
| 3 | max | -30 |  |
| [default] | | -30 |  |
| E1A09 | Numeric | -1 | -1 | 0 | Number of Delinquent CAIS Accounts, excluding Mail Order accounts |
| 0 | 0 | 46 |  |
| 1 | 2 | 24 |  |
| 3 | max | -26 |  |
| [default] | | -26 |  |
| TRD-STL-14 | Numeric | -2 | -2 | 26 | TSMR Active Short Term Loans account opened (Active CAIS) |
| -1 | -1 | 0 |  |
| 0 | 0 | 26 |  |
| 1 | 6 | -24 |  |
| 7 | 18 | -15 |  |
| 19 | 36 | 6 |  |
| 36 | max | 15 |  |
| [default] | | -24 |  |
| ResidentialStatus | String | HomeOwner | | 40 |  |
| PrivateTenantFurnished | | 26 |  |
| PrivateTenantUnfurnished | | 34 |  |
| CouncilTenant | | 10 |  |
| Cohabiting | | 34 |  |
| LivingWithParents | | 10 |  |
| [default] | | 10 |  |

### Challenger Specification

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Characteristic** | **Data Type** | **MIn Value** | **Max Value** | **Points** | **Notes** |
| Base | Numeric | [default] | | 580 | This is a static value that all instances of this scorecard will start from. |
| NDSPCII | Numeric | -3 | -1 | 0 | Consumer Indebtedness Index (SP) |
| 1 | 10 | 45 |  |
| 11 | 20 | 23 |  |
| 21 | 25 | 14 |  |
| 26 | 40 | 5 |  |
| 41 | 50 | -20 |  |
| 51 | max | -40 |  |
| default | | -40 |  |
| TRD-A-06 | Numeric | -2 | -2 | 0 | Number of Consistently up-to-date accounts L12m |
| -1 | -1 | -10 |  |
| 0 | 0 | -20 |  |
| 1 | 1 | 6 |  |
| 2 | 2 | 14 |  |
| 3 | max | 21 |  |
| default | | -20 |  |
| E1B13 | Numeric | -1 | -1 | -10 | Number of CAIS Status 3 or worse, within the last 6 Months for all Active CAIS accounts (SP) |
| 0 | 0 | 26 |  |
| 1 | 1 | -12 |  |
| 2 | max | -56 |  |
| default | | -56 |  |
| E1B01 | Numeric | -1 | -1 | 0 | Number of Active CAIS Accounts Opened in the Last 3 Months (SP) |
| 0 | 0 | 34 |  |
| 1 | 1 | 12 |  |
| 2 | 2 | -11 |  |
| 3 | max | -33 |  |
| default | | -33 |  |
| TRD-STL-19 | Numeric | -2 | -2 | 32 | Total Number of Default Short Term Loans accounts (Default CAIS) |
| -1 | -1 | 0 |  |
| 0 | 0 | 28 |  |
| 1 | 1 | -14 |  |
| 2 | max | -27 |  |
| default | | -27 |  |
| EmploymentStatus | String | FullTime | | 45 |  |
| PartTime | | 24 |  |
| Student | | 6 |  |
| SelfEmployed | | 33 |  |
| Unemployed | | -10 |  |
| Retired | | 22 |  |
| default | | -10 |  |

### NULL Handling in characteristics

### Probability of Default formula



Probability of Default formula (Silva et al, 2020).

Applied into the project as:

A screen shot of a computer program

Description automatically generated

Where:

* Math.exp() function provides the Exponent value
* 500
* 60 represents a constant, where each jump in the score in this increment changes the Probability of Default

## Policy Rules

* Decline = no offer of credit given to the applicant
* Accept

|  |  |  |  |
| --- | --- | --- | --- |
| **Rule Name** | **Decision** | **Logic** | **Notes** |
| Worst Arrears status is 2+ payments behind | Decline | E1B08 regex match 2|3|4|5|6|8 |  |
| Experian Score below threshold | Decline |  |  |
| Scorecard | Decline |  |  |
| Existing Customer | Refer |  |  |
| Accept | Accept | If none of the above Policy rules are hit |  |

# Testing the program

## JUnit

## End to End

# References

Bionic Turtle. (2011). FRM: Logistic distribution maps credit score to probability of default (PD). [Online]. Youtube. Last Updated: 2011. Available at: <https://www.youtube.com/watch?v=HnQ6QjeGMm0> [Accessed 9 May 2024].

Experian. (2024). DelphiSelect API Documentation - for Credit Decisioning data. [Online]. DelphiSelect API Documentation. Last Updated: 2024. Available at: <https://developer.experian.com/products/uk/delphi-select/delphi-select-api-docs> [Accessed 17 May 2024].

FICO. (2020). A Guide to Champion/Challenger Testing Approaches: Overview and Benefits. [Online]. FICO Community. Last Updated: 2020. Available at: <https://community.fico.com/s/blog-post/a5Q2E000000YNqoUAG/fico2227> [Accessed 17 May 2024]

Huang and Scott. (2017). Credit Risk Scorecard Design, Validation and User Acceptance. [Online]. Credit Risk Scorecard Design, Validation and User Acceptance. Last Updated: 2017. Available at: <https://cer.business-school.ed.ac.uk/wp-content/uploads/sites/55/2017/03/Credit-Risk-Scorecard-Design-Huang-and-Scott.pdf> [Accessed 17 May 2024].

QA Apprenticeships. (2024). Module 1 - Java JDBC Slides. [Online]. QA Apprenticeships. Last Updated: 2024. Available at: <https://github.com/broberts-gtw/LVL_4_Software_Dev_Repo/blob/main/Module%201%20-%20Intro%20to%20Software%20Development/Live%20Course%201%20Materials/__Course%20Stretch%20files/Java/presentation/JDBC.pptx> [Accessed 1 May 2024].

Microsoft Azure. (2010). Adding Users to Your SQL Azure Database. [Online]. azure.microsoft.com. Last Updated: June 21, 2010. Available at: <https://azure.microsoft.com/en-us/blog/adding-users-to-your-sql-azure-database/> [Accessed 3 May 2024].

Microsoft. (2023). Northwind Database - TSQL. [Online]. Northwind Database - TSQL. Last Updated: 2023. Available at: <https://github.com/microsoft/sql-server-samples/blob/master/samples/databases/northwind-pubs/instnwn> [Accessed 1 May 2024].

Silva et al. (2020). A logistic regression model for consumer default risk. [Online]. National Library of Medicine. Last Updated: 2020. Available at: <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC9041570/> [Accessed 9 May 2024].

Stack Overflow. (2008). Can an abstract class have a constructor?. [Online]. Stack Overflow. Last Updated: Updated Dec 30, 2016. Available at: <https://stackoverflow.com/questions/260666/can-an-abstract-class-have-a-constructor> [Accessed 3 May 2024].

Stack Overflow - ArrayList. (2021). Adding objects to an array list -. [Online]. Stack Overflow - ArrayList. Last Updated: 2021. Available at: <https://stackoverflow.com/questions/70406298/adding-objects-to-an-array-list-cannot-invoke-xxx-add-b> [Accessed 14 May 2024].

W3Schools. (Unknown). Java Abstraction. [Online]. W3Schools. Available at: <https://www.w3schools.com/java/java_abstract.asp> [Accessed 3 May 2024].

# Appendix

## Resources:

[Java Interface (w3schools.com)](https://www.w3schools.com/java/java_interface.asp)

[Java Abstraction (w3schools.com)](https://www.w3schools.com/java/java_abstract.asp)

[Java Exceptions (Try...Catch) (w3schools.com)](https://www.w3schools.com/java/java_try_catch.asp)

## Project Brief & Checklist

The following list is a reminder and checklist on what you should include in your final submission.

* **Introduction**
  + Description of the project.
  + The approach.
  + Project outcomes.
  + How the KSBs are evidenced through your workplace activity.
  + How the project was assigned to you.
  + The stakeholders you liaised with for this.
  + The key requirements.
  + Other considerations e.g., budget.
  + The way you interacted with various stakeholders.
* **Logic and maths**
  + write good quality code (logic) with sound syntax in at least one language.
  + apply the maths required to be a software developer (e.g. algorithms, logic and data structures).
  + Additionally, make sure you show evidence that you have applied the following:
    - Elements of programming: variables, assignment statements, data types, conditionals, loops, arrays, and input / output
    - Functions: modular programming dividing a program into components that can be independently debugged, maintained, and reused writing at least two reusable functions
    - Algorithms and data structures: classical algorithms for sorting and searching and fundamental data structures
* **Linking data**
  + effectively link code to the database and data sets.
  + link to a range of database types and embed data queries within your code.
  + make a connection to the database.
  + execute CRUD statements on the database.
  + use one-off queries and stored procedures.
  + transform returned data into a format required by the application.

## Lessons learned

* Git Repository needs to be separate from the full Apprenticeship repository

## Learning Criteria:

S7 - Apply structured techniques to problem solving, can debug code and can understand the structure of programmes to identify and resolve Issues

S4 - Test code and analyse results to correct errors found using unit testing

S1 - Create logical and maintainable codes

B10 - Committed to continued professional development

B4 - Works collaboratively with a wide range of people in different roles, internally and externally, with a positive attitude to inclusion and diversity

B7 - Communicates effectively in a variety of situations to both a technical and non- technical audience

B5 - Acts with integrity with respect to ethical, legal and regulatory ensuring the protection of personal data, safety and security

B1 - Works independently and takes responsibility. For example, has a disciplined and responsible approach to risk, and stays motivated and committed when facing challenges

B6 - Shows initiative for solving problems within their own remit, being resourceful when faced with a problem to solve

B8 - Shows curiosity to the business context in which the solution will be used, displaying an inquisitive approach to solving the problem. This includes the curiosity to explore new opportunities, and techniques; the tenacity to improve methods and maximise performance of the solution; and creativity in their approach to solutions

B9 - Demonstrates creativity and tenacity in their approach to solutions and the methods used to come to a solution, for example sees the task through to the end by devising new solutions and despite obstacles and problems along the way