# Coding exercise

Solution Developer

## Introduction

Thank you for being interested to join our team here at SEB. This exercise will help us better understand your strengths as a developer and will provide a starting point.

Whilst we use a range of the libraries and frameworks to help us build great applications, in these exercises we are going to try and keep things a bit more basic. Partly, this is to save your time, partly it's so that we can focus on some core skills like mastery of the language, code structure, readability, testability, and ease of reasoning.

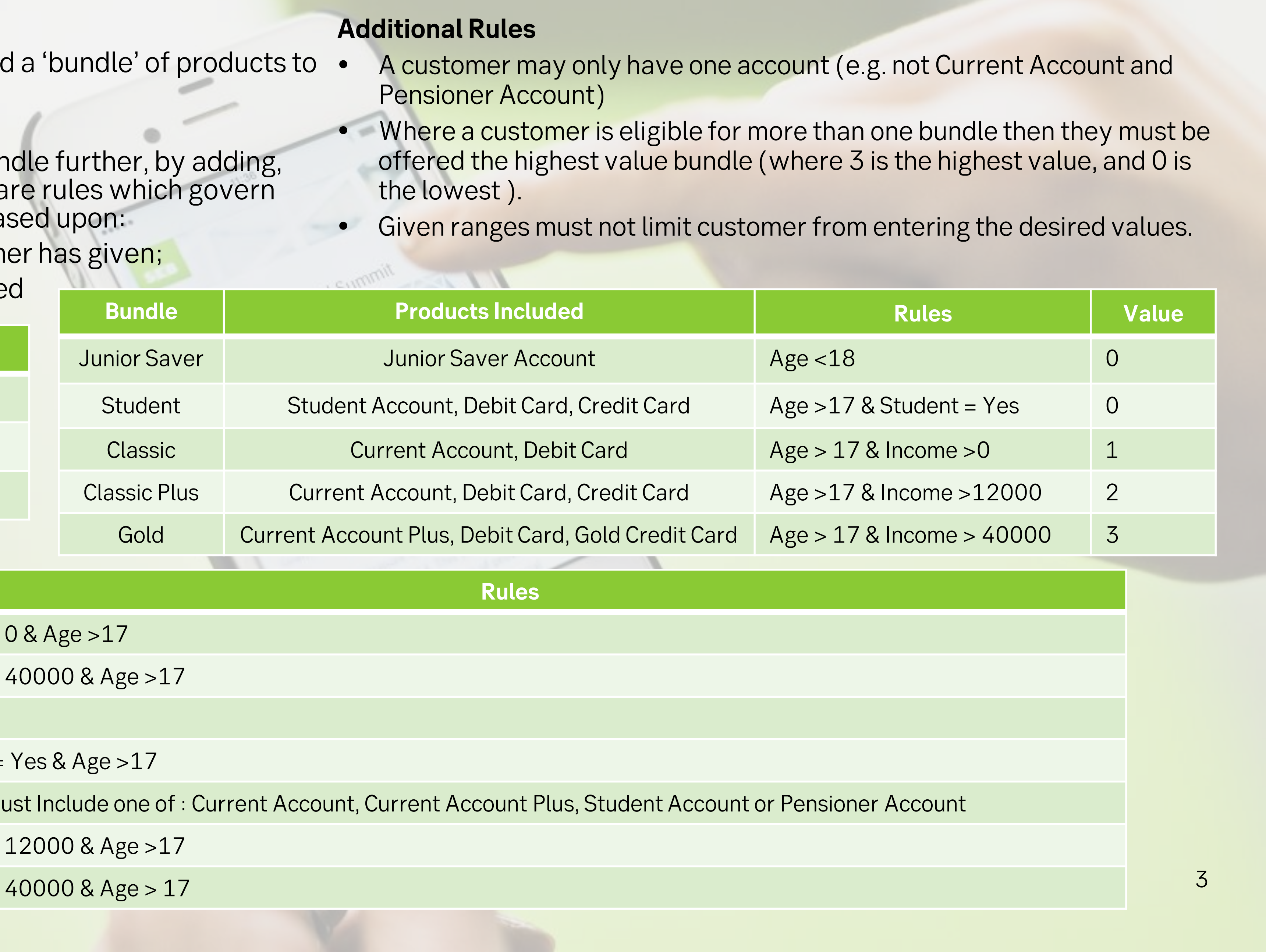
# Description

- The bank is producing a tool to recommend a 'bundle' of products to prospective customers.

- The customer may then customize this bundle further, by adding, removing, our upgrading products. There are rules which govern what products a customer may choose, based upon:
  - Answers to questions that the customer has given;
  - Other products that they have selected

**Additional Rules**

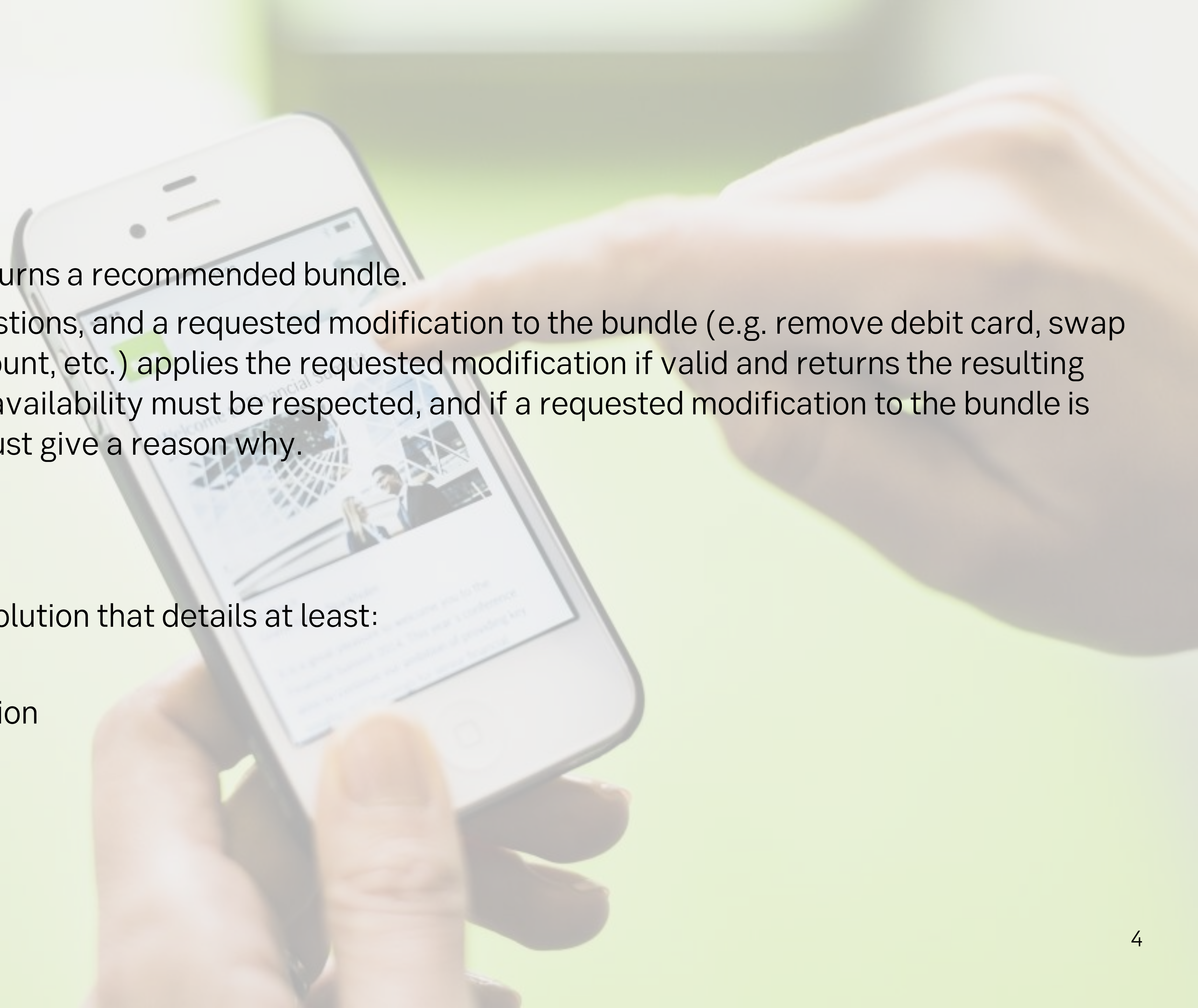
- A customer may only have one account (e.g. not Current Account and Pensioner Account)
- Where a customer is eligible for more than one bundle then they must be offered the highest value bundle (where 3 is the highest value, and 0 is the lowest ).
- Given ranges must not limit customer from entering the desired values.



| Question | Company state |
|---|---|
| Age | 0-17, 18-64, 65+ |
| Student | Yes, No |
| Income | 0, 1-12000, 12001-40000, 40001+ |

| Bundle | Products Included | Rules | Value |
|---|---|---|---|
| Junior Saver | Junior Saver Account | Age <18 | 0 |
| Student | Student Account, Debit Card, Credit Card | Age >17 & Student = Yes | 0 |
| Classic | Current Account, Debit Card | Age > 17 & Income >0 | 1 |
| Classic Plus | Current Account, Debit Card, Credit Card | Age >17 & Income >12000 | 2 |
| Gold | Current Account Plus, Debit Card, Gold Credit Card | Age > 17 & Income > 40000 | 3 |

| Product | Rules |
|---|---|
| Current Account | Income > 0 & Age >17 |
| Current Account Plus | Income > 40000 & Age >17 |
| Junior Saver Account | Age < 18 |
| Student Account | Student = Yes & Age >17 |
| Debit Card | Bundle must Include one of : Current Account, Current Account Plus, Student Account or Pensioner Account |
| Credit Card | Income > 12000 & Age >17 |
| Gold Credit Card | Income > 40000 & Age > 17 |



Write a unit tested module that:

- Given answers to the questions, returns a recommended bundle.
- Given a bundle, answers to the questions, and a requested modification to the bundle (e.g. remove debit card, swap classic account for classic plus account, etc.) applies the requested modification if valid and returns the resulting bundle. All of the rules for product availability must be respected, and if a requested modification to the bundle is not permitted then the response must give a reason why.

A readme.txt should accompany the solution that details at least:

- How to run the application
- How to run the tests of the application

# Guidance

All work must be your own. You will be expected to explain, discuss, and extend your code in the interview.

The purpose of this exercise is to help us understand how you design solutions, and your core Java skills. Often there isn't a single best answer so write your code how you see best and be prepared to discuss your choices in the interview.

You may use the latest version of Java, HTML, JavaScript. You may use any testing library. Your code should have object oriented design and should be easy maintainable, extendable and configurable.