# Cashman Practical Session

## CashMan 3001

### Candidate Instructions

You should create an application that addresses the assignment problem. You will be given 7 days to complete the solution.

Once complete, please submit the assignment via email.

You will need to supply:

* All Source code
* Automated build
* Automated test suite
* All libraries that are used. Including any third party libraries used
* Instructions on how to build the application
* Instructions on how to execute the application

### Assessment Guide

The candidate assignment is not a "pass-or-fail" exercise. Instead, it is one of the tools used by Suncorp to assess the suitability of a candidate for a position. Through the code you write we will be looking to understand what you think constitutes well-crafted and maintainable Java software.

The minimum code deliverables are:

* Source code
* Automated test suite
* Automated build (including any tools you prefer to use to aid in the creation and maintenance of your software e.g. static analysis utilities)
* We will ask you how long it took to produce your solution, or if you were unable to complete it, how long you expect it would take to complete

It is not required that you complete the entire problem, however it is required that you outline incomplete or incorrect aspects of your solution. The recommended amount of time to spend on this problem is 4 hours.

Feedback on your assessment will be available on request.

### Problem Summary

You are required to design/develop a cash dispensing application for use in an ATM or similar device. There is no need to request authorisation or availability of funds. The application should assume that all requests are legitimate; there will be other components of the system that will do things such as communicating with bank accounts and authorising withdrawals.

#### Mandatory Feature Set

* The device will have a supply of cash (physical bank notes) available.
* It must know how many of each type of bank note it has. It should be able to report back how much of each note it has.
* It should be possible to tell it that it has so many of each type of note during initialisation. After initialisation, it is only possible to add or remove notes.
* It must support $20 and $50 Australian denominations.
* It should be able to dispense legal combinations of notes. For example, a request for $100 can be satisfied by either five $20 notes or 2 $50 notes. It is not required to present a list of options.
* If a request can not be satisfied due to failure to find a suitable combination of notes, it should report an error condition in some fashion. For example, in an ATM with only $20 and $50 notes, it is not possible to dispense $30.
* Dispensing money should reduce the amount of available cash in the machine.
* Failure to dispense money due to an error should not reduce the amount of available cash in the machine.

For an ATM-style of machine (with $20 and $50 notes), the following dispensed amounts are of particular interest:

* $20
* $40
* $50
* $70
* $80
* $100
* $150
* $60
* $110
* $200, when there is only 3x$50 notes and 8x$20 notes available.

#### Optional Feature Set

* Support all other legal Australian denominations and coinage.
* The controller should dispense combinations of cash that leave options open. For example, if it could serve up either 5 $20 notes or 2 $50 notes to satisfy a request for $100, but it only has 5 $20 notes left, it should serve the 2 $50 notes.
* The controller needs to be able to inform other interested objects of activity. Threshold notification in particular is desirable, so that the ATM can be re-supplied before it runs out of cash.
* Persistence of the controller is optional at this time. It can be kept in memory. However, it should go through a distinct initialisation period where it is told the current available amounts prior to being used.