Backend Developer Test – ATM Simulation

Instructions

You will create an application that addresses the problem described below.

Your application must be written in Java, Python, NodeJS or PHP.

Your application may have a Web UI, or it can run on the Command Line. This test is focused on the Backend, so any Frontend "sugar" won't be considered as beneficial in this test.

All source code must be delivered as part of the solution, as well as clear instructions on how to build and run your application.

Your application and instructions must be targeted at either the Linux or Mac platform (not Windows).

If you are not able to provide a completely working solution, that's okay, we will be looking at your code, and the approach that you have taken as the critical part of the assessment.

Evidence of a true Test Driven Development approach will be highly regarded.

Correct use of appropriate design patterns will also be highly regarded.

Incorrect use of design patterns, or tests that add no value will be considered a negative.

If you are using libraries and/or frameworks, please provide a small explanation of the library/framework, and why they added value to your solution.

Good luck.

Requirements

We need an application that simulates the Backend logic of a cash dispensing Automatic Teller Machine (ATM).

Of course the application is not required to distribute money, but it should be able to simulate and report the outcome of people requesting money.

This simulation will not require any authentication or PIN to access the ATM.

Rather it is to be focused on keeping track of the current cash of the ATM, and dispensing only the notes available.

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 remove notes.

It must know how many of each type of bank note it has and it should be able to report back how much of each note it has.

It must support $20 and $50 notes.

It should be able to dispense only legal combinations of notes. For example, a request for $100 can be satisfied by either five $20 notes or two $50 notes. It is not required to present a list of options.

If a request cannot 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.