2020 CI401
Introduction to programming

ATM project ideas

1st March 2021

Roger Evans

Module leader

ATM project ideas

- Here are some ideas for your own additions to ATM for your project.
- You don’t have to do any of these, and you certainly don’t have to do all of them – 2-3 of the basic ones will be enough to pass
- Remember though, that not all your marks come from the coding – testing and documenting count too. Don’t forget this, whether you are doing basic changes or really fancy ones.
- Note that the difficulty ratings are indicators – they don’t correspond directly to marks (which depend on many things)

Suggestions for adding features to ATM (assessed)

- Adding effects like colour and CSS styling
- Add the ability to change your password
- Offer to check the balance before withdrawing money
- Restyle the GUI to be more like modern ATMs (eg options displayed on screen, and generic buttons to select them etc)
- Add a mini-statement option
- Add different account types with different functions (eg overdrafts)
- Connect the ATM to a real database (advanced - discuss in labs)
- ‘Encrypting’ the connection to the bank (advanced - discuss in labs)

Adding effects like colour and CSS styling (easy)

- Use simple CSS to change the colours of background, buttons, text etc on the interface (easy)

Add the ability to change your password (medium)

- Provide a change password button on the GUI
- Add new states to the model for password updates – what state do you need to be in to change the password? How many states are there in a password update? Where do you end up? What can go wrong, and where do you go then? (Draw a diagram and include it in your report?)

Offer to check the balance before withdrawing money (medium)

- This is a bit like the password case – while you are offering to check the balance (and then checking it if required), the model needs to be in a special state(s), so that it knows to go to the withdrawal function afterwards
- Draw a diagram explaining what state this starts in, what happens in each new state, what to do when it goes wrong etc.
- At present, you only press withdraw *after* you have typed the amount – you might want to change this design because it is awkward to then ask about the balance (medium/advanced)

Restyle the GUI to be more like modern ATMs (medium/advanced)

- Change the shape and layout of components of the interface (medium)
- Use the JavaFX interface design tools to redesign the whole interface (advanced)
- Are there additional functions you would also include?

Add a mini-statement option (medium/advanced)

- Add a new button to display a mini-statement (maybe the last 5 transactions) on the currently logged in account
- This requires you to store transactions somewhere – probably in the BankAccount object itself (as an array, or ArrayList?)
- A more advanced version would store them in a file, so that the system remembers transactions from the previous time it was run

Add different account types (eg with overdrafts)(medium)

- Use subclassing to add new kinds of bank account, for example an account with an overdraft, or one with mini-statements, or one that limits you to 3 withdrawals per day (imagine the program gets started again every day, so just do 3 withdrawals per run of the program)
- Add methods to manage the new facilities (set/get overdraft etc), and modify other methods (eg withdrawing is allowed if within overdraft)
- Extend the Model and View to give access to these features (Eg display the overdraft). How can you cope with having different kinds of account, which don't all do the same thing?

Connect the ATM to a real database (advanced - discuss in labs)

- You each have an account on https://brighton.domains, which provides you with a database (among other things)
- Create a bank database with a table of bank accounts, and then add a new bank account type which store its information there

‘Encrypting’ the connection to the bank (advanced - discuss in labs)

- The connection to the bank, and especially the passing and storage of the password, is all in readable text
- Change it to use an encrypted form, to make it more secure (So every method call to the bank takes encrypted arguments and returns encrypted results, and the password is stored in the object in encrypted form)