1. **Explain the overall design of your program and justify why you chose this particular class structure and set of user-defined methods.**

This project is structured using a custom Model View Controller (MVC) architecture, since there are discrete choices in the system (Card Payment and Cash Payment) it seemed logical to structure each method of payment as a separate model each with their specific implementations. Each model also has a corresponding view that handles all the UI operation for their specific implementation (for example, the Card Payment having a different sub window to enter card details.) The controller is embedded into the view instead of being a separate entity, this design decision stems from the want to keep the project's simplicity.

2. Describe how user input is handled in your program and discuss the role of exception handling (try–except) in preventing incorrect or unexpected behavior.

User Input for this project is gathered through UI elements and handled using the Observer design pattern. Whenever there is a need for numerical or lexical input, the project can gather such inputs via UI elements such as a text box or a text entry, the program can then represent certain actions such as "Pay with Cash" or "Pay with Card" as a UI element (like a button) and the Observer pattern is then handles the implementation. A great example of this design pattern is displayed when a user clicks a button. When a button is pressed, that button triggers an event that is listened to by objects that care about that specific button press and contains specific implementations for such actions. For example a "Pay with Cash" button may be observed by a class that contains implementations for paying with cash which it can then execute when the "Pay with Cash" button is triggered.

3. Identify one limitation of your current implementation and explain how it could be improved using additional OOP concepts or better error handling.

As mentioned in the earlier question, this project is structured following a loose MVC architecture, by embedding Controllers into Views, it makes it so that the code is simpler to write but that also means that data handling is coupled to the representation of data, which depending on the project and its scale, might not be a good idea and can make the codebase very messy and hard to read, though for this simple project, this specific approach does just fine. However, for future recommendation, I believe a decoupling of data handling and data representation might provide a better opportunity for scalability.