According to the UML Class Diagram in **Fig. 2** is it possible for the **canWithdraw** operation to be inherited and used directly in the **ChildAccount** class? Describe what problems may arise **when implementing this design** and what can be done to solve these.

1 a) Yes, the canWidthdraw function can be used on the ChildAccount class because ChildAccount inherits this function and the function is public. When implementing this design, there are problems. For example, in the base class canWidthdraw function, it compares the amount to withdraw is less than the balance. In the ChildAccount class, this will be implemented differently since this will need to take in account the balance AND the minimum paid in and max paid in.

The C++ **ATM** class includes a **BankAccount** pointer as one of its data members. Explain why, in this case study, using a **BankAccount** instance instead of a pointer to implement this relationship, would not be appropriate. Give specific examples **from your final solution** to illustrate your answer.

1 b) make sense to use pointer instead of instance because if there is a BankAccount instance, copying another subclass to a base class means object slicing occurs. Some of the information from the subclasses of BankAccount is lost in the bank account pointer. However, using a pointer means the BankAccount pointer can point to a subclass and this means down casting can occur so information from the subclass is recovered and can be accessed.

In the C++ implementation given, what is the nature of relationship between the **Card** and **List<string>** classes, how should it be represented in UML and what C++ mechanisms are involved in its C++ implementation?

1 c)