Summer School Day 3 – Design and Classes.

Our class today was looking primarily on the ideas of classes.  
  
1. Discussion Stage (work in pairs and then discuss as a class). We are going to start on the design of a class about the concept of an account. From this user story, let’s see if we can design an object which meets with the needs:

Igor is a customer of The First Bank Kazan. The bank has never had more than 100 customers at any time and allows customers to open accounts, close accounts, and processes internal transactions between their accounts and deposits and debits from customers.

In order to open an account, it is required for Igor to give the bank information such as his first name, last name, and patronym. The bank records his date of birth, sex, and passport number in order to ensure security. First Bank Kazan has three account types, each with their own fees and interest amounts which are applied at the end of the month, these are Savings (5% interest), chequing (1000p fee at end of month), and business (1% interest). Igor has a balance, the bank requires that a Savings account has an initial deposit of 50,000p, a chequing account has a minimum 1,000p initial deposit, and businesses must have 5,000,000p initial deposit. Accounts can have transactions applied to them, which must be checked for validity. At the end of the month all transactions made must be listed in an official statement of the account, an account can have no more than 100 transactions per month.  
  
Igor can make transitions. A transaction is applied to an account based on the account number(s). Transactions can be withdrawals, deposits, transfers, and interest payments. Each transaction is given a transaction number, which is the next in sequence. Transactions should have a means of being displayed at the end of the month in a grand statement of all transactions of all accounts.

2. Create a class which is a file record of a customer based on the design in stage 1. Ensure that errors are protected against (you cannot have an overdrawn account and any transaction should not allow for this). Initialize a few example of this class, and print them to the display in a reasonable manner.  
  
3. Create a class transaction which is an implementation of the transaction from your design. Again, ensure against err transactions (a proper transaction requires an account to be enacted upon, etc.)

4. The bank may wish to have multiple branches in future, how can the current model be extended to account for interactions between branches?