CISC 327 - Software Quality Assurance

Lecture 6
Course Project
Course Information

• **Mini-Exams**
  - This year, in place of a final examination, there will be four *in-class mini-exams*, worth *12.5%* each
  - Subject to lecture scheduling slips, mini-exams are on:
    - **Friday, September 29**
      - Lectures 1-7: Quality Assurance, Software Process
    - **Friday, October 27**
      - Lectures 8-17: Black Box & White Box Testing
    - **Friday, November 10**
      - Lectures 18-23: Continuous Testing, Software Inspection
    - **Wednesday, November 29**
      - Lectures 24-30: Software Analysis, Metrics, Security
Course Information

• Course Project
  – The course project will consist of six assignments, with handouts for each one
  – Assignment handouts will be early whenever possible, so that you may work ahead at any time
  – Subject to scheduling slips, assignments due on:
    • Thursday, October 5
    • Thursday, October 19
    • Thursday, November 2
    • Tuesday, November 14
    • Wednesday, November 22
    • Friday, December 1 (at latest)
Course Information

• Tutorials and Advising
  – Project advising @ Caslab (WLH 310), **time TBA**
    • Advising times will be *informal*, designed to provide you with *practical* and *technical* advice on your project
    • *Occasionally* a formal tutorial *may* be scheduled if necessary
  – Online links on the course web page for:
    • Linux *command line programming* and *shell scripting*
    • Windows *command line programming* and *batch scripting* if you prefer
Course Information

• Assignment Submission
  – Assignments will be handed in as a PDF document in OnQ, by 10pm on the due date at latest
  – Be sure to indicate clearly your team name and student names and numbers on every submission!
  – This is a course in quality – neatness counts!
  – Think of your submission as a professional paper report, with appropriate titling, sectioning, paging
  – Marked assignments will be returned in OnQ
Course Project

• Project Phases
  – The project will be done in several phases, each of which will be an assignment
  – Phases will cover steps in the process of creating a quality software result in the context of an eXtreme Programming process model
  – Assignments will be on the quality control aspects of requirements, prototyping, testing, integration, and analysis of the product you are building
  – You can always work ahead on the next assignment in advance to manage your time
Course Project

• Project Phases
  – Assignments should always use the simplest possible solution
  – Assignments must be done exactly as the assignment specification says - no exceptions!
  – You can ask the customer (me) for clarifications about the requirements or expectations any time
  – I will respond and post answers quickly
• Queen’s Banking Simulation Console
  – Our very own Bizarro World Interac system
• Consists of a Front End and a Back Office
  – The Front End is a standalone retail banking terminal for pseudo-ATM banking transactions
    • login, logout, deposit, withdraw, transfer, plus account creation and deletion when possible
  – The Back Office is an overnight batch processor to maintain and update a master events file
    • Aggregate the information from the campus-wide set of Front End terminals
QBASIC

- The **Front End**

  - Reads in a file of valid account numbers, processes a stream of transactions one at a time, and writes out a summary file of transactions at the end of the day
QBASIC

• The Back Office
  – Reads in the previous day's master accounts file and applies all of today’s transactions from a set of transaction files to produce new master accounts and valid accounts files
QBASIC Front End Requirements

• Informal Customer Requirements for the Front End
  – The Front End handles a sequence of transactions, each of which begins with a single transaction code (word of text) on a separate line
  – The Front End must handle the following transactions:
    • login start a Front End session (processing day)
    • logout end a Front End session
    • createacct create a new account (privileged)
    • deleteacct delete an existing account (privileged)
    • deposit deposit to an account
    • withdraw withdraw from an account
    • transfer transfer between accounts
QBASIC Front End Requirements

- What does a **sample session** look like?
  - Let's say on a given day, only one customer visits a Front End ATM and **transfers** $100 from one account to another, then **withdraws** $50 of it
QBASIC Front End Requirements

• **login**: start a Front End session
  – should ask for the **type** of session, which can be either
    • **atm**, which means ATM terminal mode
    • **agent**, which means **privileged** (teller) mode
  – after **type** is accepted, reads in the **valid accounts file** (see requirements) and begins accepting other transactions
QBASIC Front End Requirements

– Constraints:
  • no transaction other than login should be accepted before a login
  • no subsequent login should be accepted after a login, until after a logout
  • after an ATM login, only unprivileged transactions are accepted
  • after an agent (privileged) login, all transactions are accepted
QBASIC Front End Requirements

• **logout**: end a Front End session
  – should write out the *transaction summary file* (see requirements for the file) and stop accepting any transactions except **login**
  – **Constraints**:
    • should only be accepted when logged in
    • no transaction other than **login** should be accepted after a logout
QBASIC Front End Requirements

• **createacct**: create a new account
  – should ask for the new **account number and name** (as text lines)
  – should save this information for the **transaction summary file**, but no transactions on the new account should be accepted in this session
QBASIC Front End Requirements

— Constraints:

• privileged transaction, only accepted when logged in to agent mode
• new account number is exactly seven decimal digits not beginning with 0
• new account number must be different from all other current account numbers
• new account name is between 3 and 30 alphanumeric characters, possibly including spaces but not beginning or ending with a space
QBASIC Front End Requirements

• **deleteacct**: delete an existing account
  – should ask for the **account number** and **account name** (as text lines)
  – should check that the account number is valid, and save the account number and name in the **transaction summary file**

  – **Constraints**:
    • privileged transaction, only accepted when logged in to **agent** mode
    • no further transactions should be accepted on a deleted account
QBASIC Front End Requirements

- **deposit**: deposit to an account
  - should ask for the **account number** and the **amount to deposit** in cents (as text lines)
  - should check that the account number and amount are valid
  - should save info for the **transaction summary file**
  - **Constraints**:
    - deposits above $1,000.00 should be rejected in ATM mode (max $999,999.99 in agent mode)
QBASIC Front End Requirements

• **withdraw**: withdraw from an account
  – should ask for the account number and the amount to withdraw in cents (as text lines)
  – should check that the account number and amount are valid
  – should save info for the transaction summary file

– Constraints:
  • withdrawals above $1,000.00 should be rejected in ATM mode (max $999,999.99 in agent mode)
  • a total of at most $1,000.00 can be withdrawn from a single account in a single ATM session (no limit in agent mode)
QBASIC Front End Requirements

• **transfer**: transfer from one account to another
  – should ask for the **from account number**, the **to account number**, and the **amount to transfer** in cents (as text lines)
  – should check that the account numbers and amount are valid
  – should save info for the **transaction summary file**

– **Constraints**:  
  • Transfer amounts above $1,000.00 should be rejected in ATM mode (max $999,999.99 in agent mode)
QBASIC Front End Requirements

• Transaction Summary File
  – At the end of each session (processing day), when the logout transaction is processed, a transaction summary file for the day is written, listing every transaction made in the session
  – Contains transaction messages (text lines) of the form:

    CCC AAAA MMMM BBBB NNNN
QBASIC Front End Requirements

• Transaction Summary File

  
  CCC AAAAA MMMMM BBBB NNNN

  – CCC is a three-character transaction code, where
    DEP = deposit, WDR = withdrawal, XFR = transfer,
    NEW = create account, DEL = delete account,
    EOS = end of session
  – AAAAA is the first (to) account number
  – MMMMM is the amount, in cents (e.g., 123 = $1.23)
  – BBBB is the second (from) account number
  – NNNN is the account name
QBASIC Front End Requirements

• Constraints:
  – each line is at most 61 characters (plus newline)
  – the transaction code is always the first three characters of the line
  – items are separated by exactly one space
  – account numbers are always exactly seven decimal digits, not beginning with 0 (e.g., 1000327, 9379210)
  – monetary amounts are between 3 and 8 decimal digits, 000 to 99999999, representing $0.00 to $999,999.99
  – account names are between 3 and 30 alphanumeric characters (A-Z, a-z, 0-9), possibly including spaces, but not beginning or ending with a space (e.g., XYZ, ThisAcct, My 3rd account, …)
  – unused numeric fields are filled with zeros (e.g., 0000000 for account numbers, 000 for monetary amounts)
  – unused account name fields are filled with three asterisks: ***
  – the file ends with an end of session (EOS) transaction code
QBASIC Front End Requirements

• Valid Accounts List File
  – Consists of text lines each containing only an account number
  – **Constraints:**
    • each line is exactly 7 characters (plus newline)
    • account numbers are always exactly seven decimal digits, not beginning with 0 (e.g., 1000327)
    • the file ends with the special (invalid) account number 0000000
    • Comes from the Back End, so you can assume it is well-formed
QBASIC Front End Requirements

• General Requirements for the Front End
  – The Front End should never crash, and should never stop except as directed by transactions
  – The Front End cannot depend on valid (terminal) input – it must gracefully and politely handle bad input of all kinds
    • But: you can assume that input is at least lines of text!
QBASIC Back Office Requirements

• Informal Customer Requirements for the Back Office
  – The Back Office reads the Master Accounts File and the Merged Transaction Summary File (see below)
  – It applies all transactions to the master accounts to produce the New Master Accounts File and the New Valid Accounts List File
QBASIC Back Office Requirements

• The Back Office enforces the following business constraints, and produces a failed constraint log on the terminal as it processes transactions

  – Constraints:
    • No account should ever have a negative balance
    • A deleted account must have a zero balance
    • A newly created account must have a new unused account number
    • The account name given in a delete transaction must be the same as the name associated with the deleted account number
QBASIC Back Office Requirements

• The Master Accounts File
  – The Master Accounts File consists of text lines of the form:

    AAA MMM NNN

where:
  • AAA is the account number
  • MMM is the account balance, in cents
  • NNN is the account name

– Constraints:
  • each line is at most 47 characters (plus newline)
  • items are separated by exactly one space
  • account numbers, balances and names are as described for the Transaction Summary File
  • the Master Accounts File must always be kept in ascending order by account number
QBASIC Back Office Requirements

• **The Merged Transaction Summary File**
  – The concatenation of any number of Transaction Summary Files output from **Front Ends**, ended with an empty one (one containing no real transactions, just a transaction with an EOS transaction code and unused other fields)

• **The New Valid Accounts List File**
  – A file containing every active account number in the **New Master Accounts File**, in the format described for the **Front End**
QBASIC Back Office Requirements

• General Requirements for the Back Office
  – The Back Office uses only internal files, so it can assume correct input format on all files
  – However, the values of all fields should be checked for validity, and the Back End should stop immediately and log a fatal error on the terminal if any value is invalid
Assignment 1: Front End Requirements Tests

- Due Thursday, October 5th
  - Create and organize a complete set of requirements tests for the Front End of QBASIC to test for every required behaviour
  - Bonus for discovering missing or erroneous requirements (if customer agrees)

- Hand in as a PDF file by OnQ submission before 10pm on the due date, but you are encouraged to work ahead and hand in assignments early!  (...once we post them)
CISC 327 Course Project

• You should hand in:

1. An organized list of all your test cases and what they are intended to test (a table of test names and intentions, in English)

2. For each test case, the actual test input file and expected output file (as text file printouts)

3. A test plan document, outlining how your tests are organized (in directories or whatever), how they will be run (as shell scripts, Windows batch files, or whatever), and how the output will be stored and organized for reporting and comparison with later runs (make text file printouts of any directory structures and script files created)
CISC 327 Course Project

• What does a test case look like?

**Test** T1: login command, ATM case
**Purpose:** check that login is accepted
**Input** t1in.txt:
  - login
  - atm
  - logout
**Input files:** valid accounts file with no accounts
**Output files:** transaction summary file with no transactions
**Terminal output** t1out.txt:
  - empty, or possibly information messages in response to commands
CISC 327 Course Project

• But first: Assignment #0!
  – Choose teammates to pair program with
  – Think about the programming language and environment you want to work in
  – Sign the team agreement, due Tuesday in lecture or on OnQ
  – If you haven’t yet found teammates, email me