MacBank ABM Demonstration

By: Jasper Chan
Kevin Bezjak
Thomas Griffatong

Outline

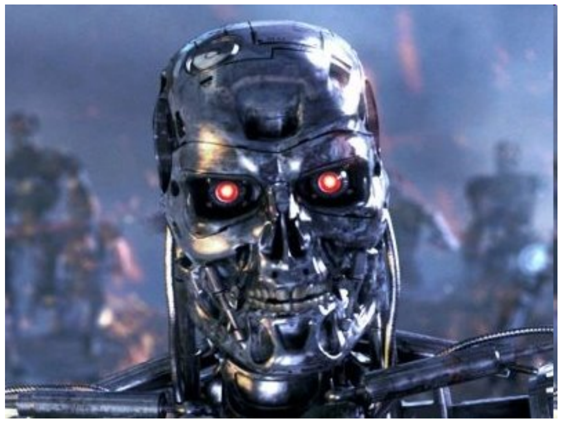
- Course information
- What is software engineering? Why is it so important?
- Software Life Cycle
- What is a specification? How do you verify it?
- Request for Proposal – How do we transition from RFP to SRS?
- Our SRS document
- How do we go from High-level design to Low-level design?
- What is a MacBank ABM?
- ABM Implementation
- Conclusion
- What can we take home from all this?

Course Information

- Course: 3K04 - Software Development
- Professor: **Dr. Kamran Sartipi**
- Focus:
  1. Software Design Process
  2. Documentation
  3. Using Specification
  4. Module specification, interfaces, internal and documentation
  5. Software inspection and testing
  6. Professional responsibility

People involved

- Jasper Chan
- 3rd year Mechatronics

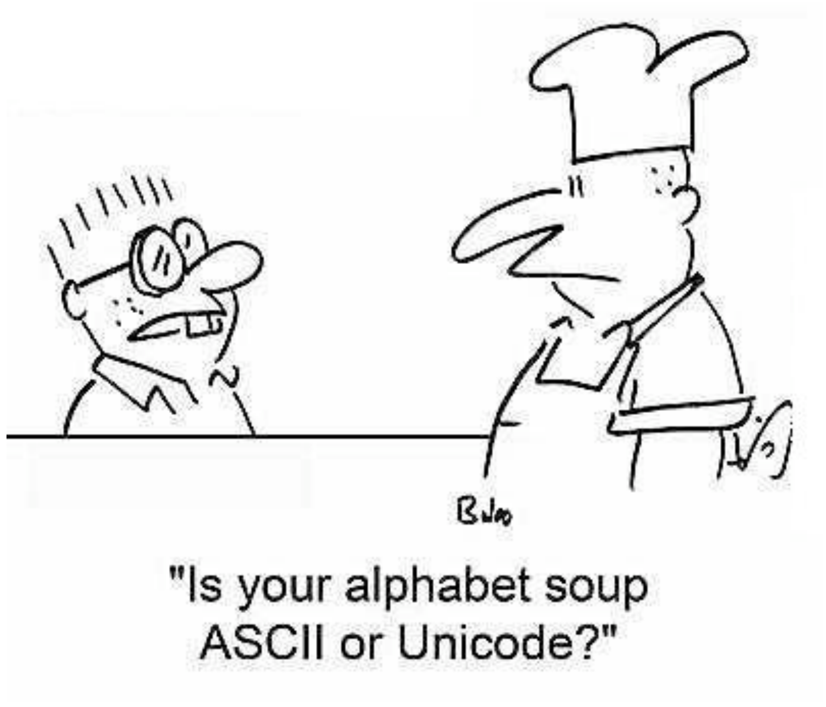


- Kevin Bezjak
- 3rd year Computer Engineering

- Thomas Griffatong
- 3rd year Computer Engineering

What is Software Engineering?

- Software engineering is an engineering discipline which is concerned with all aspects of software production
- The application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software -- IEEE definition



Why is Software Eng. Important?

- Standardization as to how to create a piece of software to minimize errors
- Errors can be fatal!
  - Therac-25
    - Software failed to detect and prevent patient from receiving overdose of radiation
    - Inadequate software design and development process resulted in the deaths of three people due to radiation poisoning

Software Life Cycle

1. <u>Requirements and analysis and specification</u>
   - Purpose: Identify and document the exact requirements for the system Example: ABM interview, SRS
2. <u>System design and specification</u>
   1. High-level/Architectural Design – overall organization of system in terms of high level components and interactions among them
   2. Low-level/Detailed Design – Defining interfaces within in each component and relationships between other components
   - Example: SDS
3. <u>Coding and module testing</u>
   - Engineer produces actual code that will be delivered to customer
     - Example: When we were physically coding our modules etc
4. <u>Integration and system testing</u>
   - All modules that have been developed and tested individually are put together to be tested as a whole system
     - Example: Putting it together to test, GUI etc
5. <u>Delivery and maintenance</u>
   - After system passes all tests, it is delivered to the customer and enters modification phase
     - Example: Lab 10 when we handed in our ABM software

Specification

- A statement of agreement between producer and consumer
  - Emphasizes on what to do vs how to do it
- Good specifications are precise, clear, unambiguous, consistent with internal/ external completeness
- Example of good specifications:
  - Keep track of the amount of money the ABM machine contains in its stock and alert the bank staff when the stock is equal, or less than $10,000.

Specification (continued)

- Uses of Specification:
  - Statement of user requirements
    - Can lead to failure if not clear
  - Statement of the interface between the machine and controlled environment
    - Miscommunication can cause unexpected inputs and the system can fail
  - Statement of requirements for implementation
    - Write out SRS, SDS etc
  - Reference point during maintenance
    - Something you can refer to when you make an upgrade to the system

Specification (continued)

- How do we verify specification?
  1. Observe dynamic behaviour of the system to check whether it does what we think it should do (aka simulator)
  2. Analyzing the properties of system to deduce specifications (called property analysis)

Request for Proposal (RFP) – How do we transition from RFP to SRS?

- Document that provides us general information with regards to the client, Brief hardware specification and general software requirements
  - It was our job to clarify ambiguous requirements and inquire additional information for exact details as to how each method was to be handled
- This was the given Request for Proposal

Our SRS document

- An SRS is the customer's assurance that the development organization understands the issues or problems to be solved and the software behaviour necessary to address those problems
  - Serves as an input to the design specification; parent document to subsequent documents
  - It also serves as a product validation check

1. Careful review and analysis of RFP
2. Generate questions to ask and clarify requirements

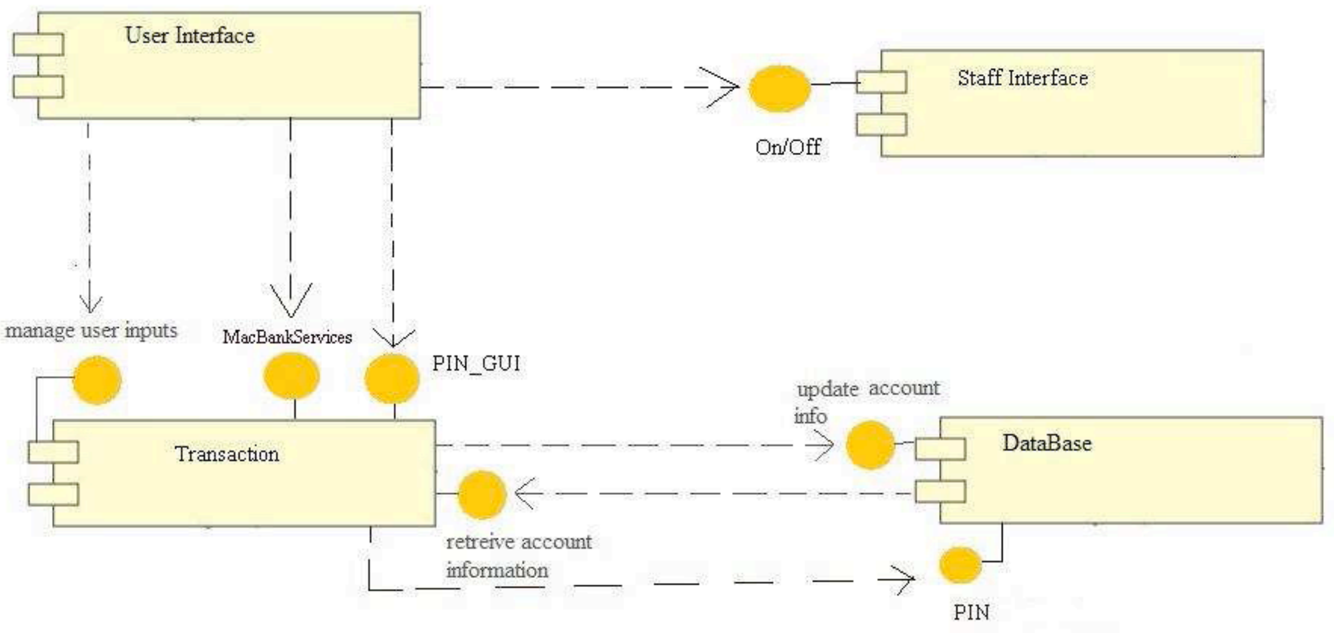
- Here is our SRS

How did we create our High-level design?

- Information used from the SRS:
  - User Interface
  - Functional Requirements
  - Use Case Diagram

Architecture

- Component diagram designed off the Use Case



Architecture: User Interface

- User interface is where customer interacts with machine
- Connected to transaction
- Customer uses it to call transactions
- Staff also use it for their needs

Architecture: Transaction

- User interface calls transactions
- Depending on input certain transaction done
- Also communicates with the database for accounts and their information to be used
- Updates database when customer finished

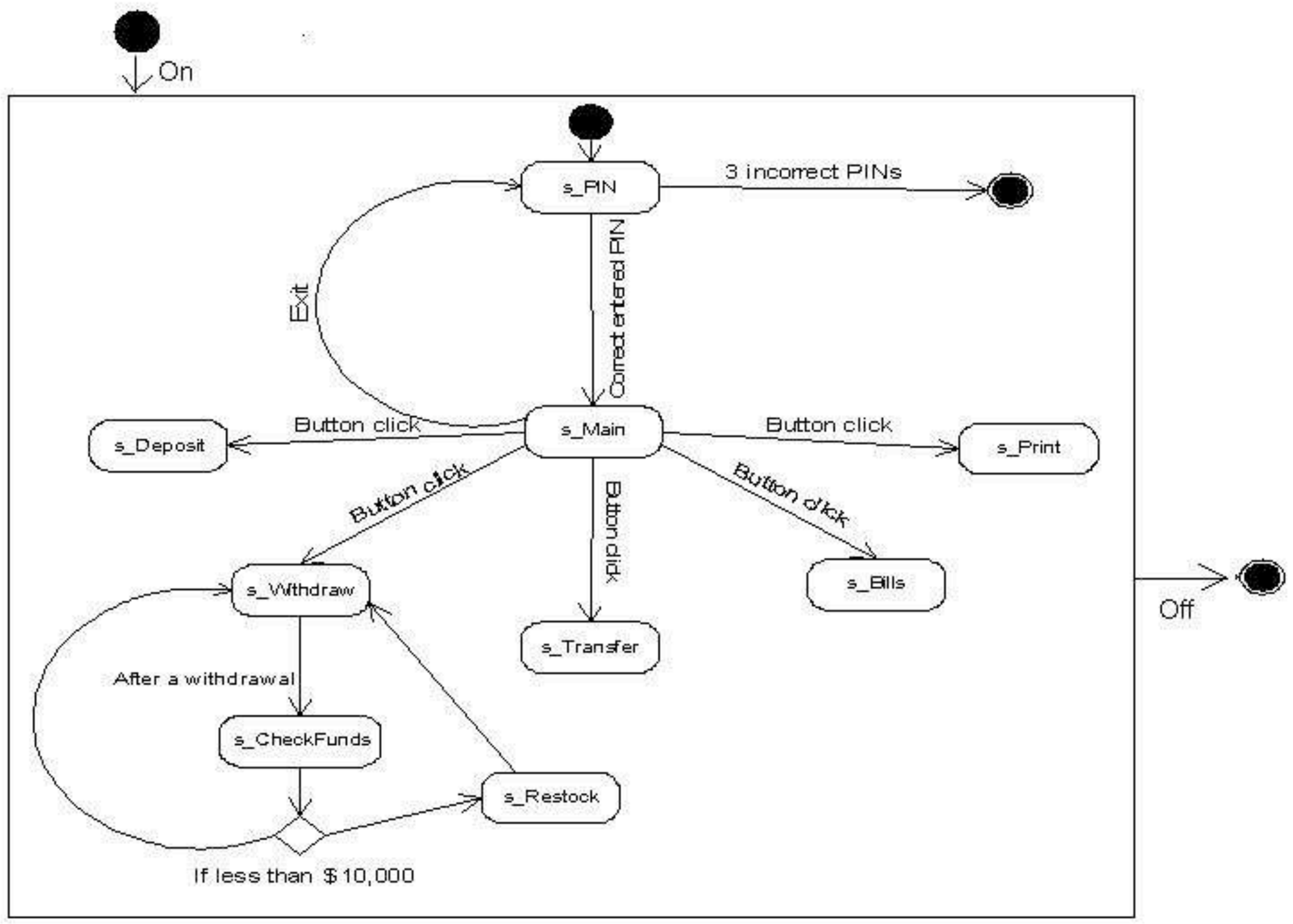
Architecture: Database

- Holds all info for ABM
  - Amount of money left inside
  - Customer accounts and information

Architecture: Staff Interface

- Uses the UI to access ABM for restocking
- In our software, we used a button to represent it but in real life, it would be replaced with a key hole

High Level Design – State Chart



High Level Design

- Started with the skeleton:
  - Main screen, PIN screen and transactions
  - Built off the base to incorporate more advanced designs such as PIN errors and turning off the ABM.
  - Added step by step until all requirements are met

How did we go from High-level to Low-level design – Step 1

- Carefully review the High-level design state chart and write out our module overview
  - This is a crucial step since it is the basis for our low-level design
  - Our approach: sit down as a group and start writing the module of the initial state of the machine to the end
    - Bring out ideas as to how to implement to create a better idea as to what variables are needed, what other modules can access it etc

Step 2

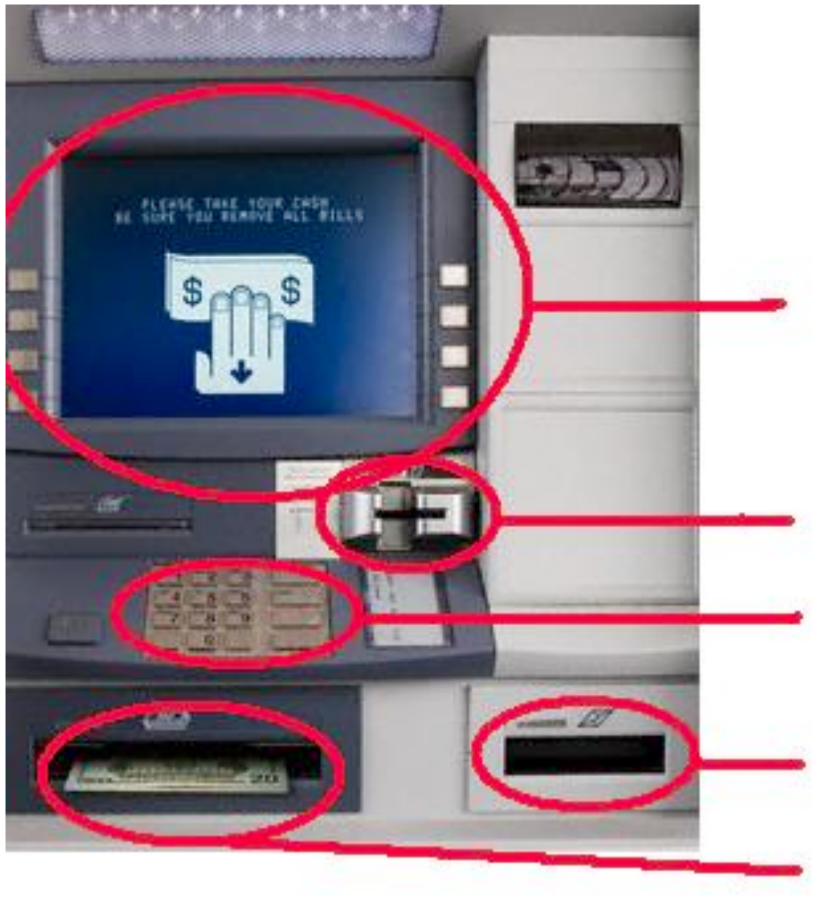
- We needed to then establish the relationship between each module with a Class Diagram
  - This step helps us visualize how each module is going to be used and ultimately, implemented

Step 3

- Write out our Module Specification with state charts for each individual module for clarity and visual representation
  - This describes in detail about each module
    - Information such as the amount of variables, which modules can call on etc is shown

What is the MacBank ABM?

- Automated Bank Machine (ABM) used to decrease waiting time for clients who want to use basic banking activities



ABM Implementation

- Java was chosen because it has a GUI, more libraries and is a lot more functional
- GUI:
  - Created using the wizard
  - Followed what was created in our documentation

ABM Implementation (Part 2)

- 30 sec Timer: ABM signs out if inactive
  - Easily implemented with swing
    - C would require a timer
- Reset accounts on midnight
  - Accesses the system time of computer
    - Not able to do this in C

ABM Implementation (Part 3)

- Classes:
  - Using classes makes passing information for transactions and database much easier
  - Keeps certain data protected from access

ABM Demonstration

Conclusion

- What did we learn about teamwork?
  - Full dedication to the project and team regardless of personal schedules
  - Rising up to the occasion when your teammate requires help
- It is greatly beneficial to work with students from different programs; widens the area of expertise available at disposal
- Valuable Lesson:
  - How to transition from interview to finished product

What can we take home from all this?

- We can safely say that we are confident in our own abilities to be able to contribute our work for any future SRS and SDS documents that may come in our future career
  - Don’t get cocky though! There is still a lot we can learn about software development & documentation!

Heh.. **Cock**-y…

Thank you for your attention!

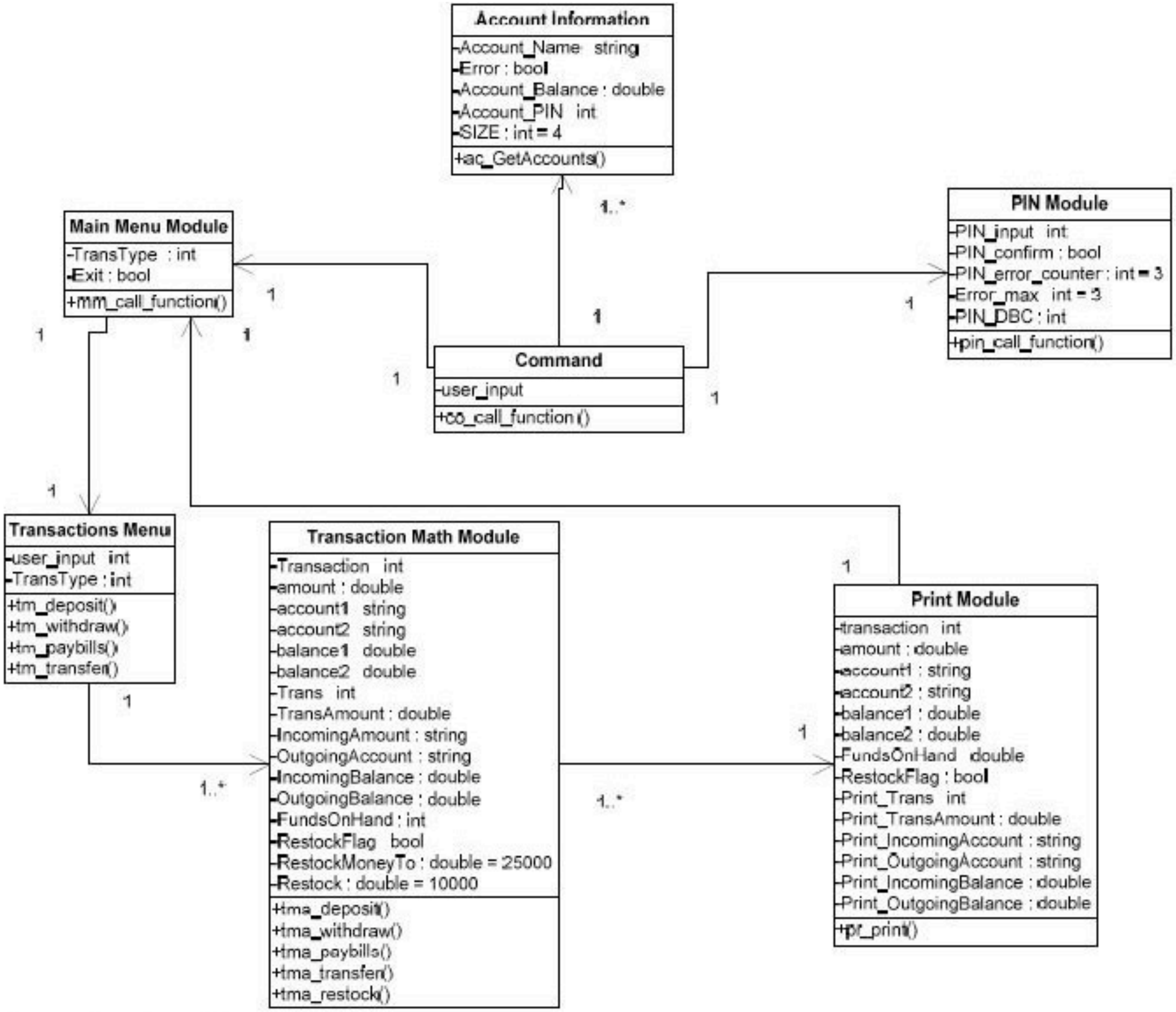


Figure 4: *Class Diagram of ABM*

Return to page