Importance of Modeling

Object-Oriented Analysis and Design

Introduction to Modeling

- Modeling is a fundamental engineering practice used to understand, design, and communicate complex systems before building them in reality.

Dog House Example

- A simple dog house can be built with minimal planning.
- • Few requirements
- • Low risk of failure
- • Easy to rebuild if mistakes occur

Family House Example

- Building a family house requires planning and blueprints.
- • More users and requirements
- • Compliance with standards
- • Managing cost and changes

High-Rise Building Example

- Complex projects need extensive modeling.
- • Multiple stakeholders
- • High cost of failure
- • Detailed architectural models required

Software Development Analogy

- Many software projects are complex like high-rise buildings,
- yet are sometimes approached like simple dog houses.
- This often leads to failure.

Why Software Projects Fail

- • Lack of planning and modeling
- • Over-reliance on coding alone
- • Unmanaged complexity and change

What Is a Model?

- A model is a simplification of reality.
- • Provides blueprints
- • Focuses on important aspects
- • Can be structural or behavioral

Uses of Modeling in Other Fields

- • Architecture and construction
- • Aircraft and automobile design
- • Electronics and communication systems
- • Movies, business, economics, and sociology

Why Do We Model?

- Modeling helps us to:
- 1. Visualize systems
- 2. Specify structure and behavior
- 3. Guide construction
- 4. Document decisions

Modeling and Complexity

- We model complex systems because we cannot understand them fully at once.
- Modeling supports divide-and-conquer and higher abstraction.

Benefits of Modeling in Software

- • Helps build the right system
- • Reduces risk and rework
- • Improves communication
- • Essential as systems grow more complex

Principles of Modeling

Object-Oriented Analysis and Design (OOAD)

Introduction to Modeling Principles

- Modeling has a long history in engineering disciplines.
- It helps in understanding, designing, and managing complex systems.

Principle 1: Choice of Models Matters

- The models we choose strongly influence how a problem is understood and solved.
- • Right models reveal key insights
- • Wrong models mislead and focus on irrelevant issues

Example from Science & Engineering

- • Quantum physics problems become easier with the right mathematical model
- • Wind tunnel models help predict building behavior
- • Mathematical simulations allow testing more scenarios

Impact on Software Development

- Different viewpoints create different systems:
- • Database view → Entity-Relationship models
- • Structured analysis → Algorithm-centric models
- • Object-Oriented view → Classes and interactions

Principle 2: Levels of Precision

- Every model can be expressed at different levels of detail.
- • High-level view (30,000-foot view)
- • Low-level detailed view (implementation details)

Levels of Precision in Software

- • Simple UI prototypes for users
- • Detailed models for interfaces and performance issues
- • Analysts focus on 'what'
- • Developers focus on 'how'

Principle 3: Models Must Reflect Reality

- Models must have a clear connection to the real world.
- Simplifications should not hide critical details.

Reality and Software Models

- • Poor connection causes divergence between design and implementation
- • Object-oriented modeling helps unify system views
- • Reduces gap between analysis and design

Principle 4: No Single Model Is Enough

- Complex systems require multiple models.
- Each model captures a different aspect of the system.

Building Analogy

- Buildings require multiple plans:
- • Floor plans
- • Electrical plans
- • Plumbing plans
- • Heating and structural plans

Nearly Independent Models

- Models can be studied separately but are interrelated.
- Together, they provide a complete understanding of the system.

Object-Oriented Modeling & UML

OOAD – Section 1.3 and Conceptual Model of UML

Object-Oriented Modeling

- Engineers use different models (structural and dynamic) depending on concerns.
- In software, modeling can be algorithmic or object-oriented.

Algorithmic vs Object-Oriented View

- Algorithmic perspective:
- • Focus on procedures and control flow
- • Hard to maintain as systems grow

- Object-Oriented perspective:
- • Focus on objects and classes
- • Better for change and complexity

Objects and Classes

- Object:
- • Identity
- • State
- • Behavior

- Class:
- • Blueprint describing common objects

Why Object-Oriented Modeling?

- • Works for systems of all sizes
- • Supported by modern languages and tools
- • Enables component-based development
- • Foundation of UML

Introduction to UML

- Unified Modeling Language (UML)
- • Created by Booch, Rumbaugh, and Jacobson
- • Became a de facto modeling standard
- • Used to design systems and programs

What is UML?

- • A modeling language, not just diagrams
- • Captures syntax and semantics
- • Used for software and database modeling

Overview of UML

- UML is used for:
- • Visualizing
- • Specifying
- • Constructing
- • Documenting systems

Visualizing

- • Uses standard graphical notation
- • Semi-formal language
- • Helps understand complex systems visually

Specifying

- • Models are precise and unambiguous
- • UML defines syntax and semantics
- • Captures analysis, design, and implementation decisions

Constructing

- • UML models map to OO languages
- • Forward engineering: Model → Code
- • Reverse engineering: Code → Model

Conceptual Model of UML

- Three major elements:
- 1. Basic Building Blocks
- 2. Rules
- 3. Common Mechanisms

1. Basic Building Blocks

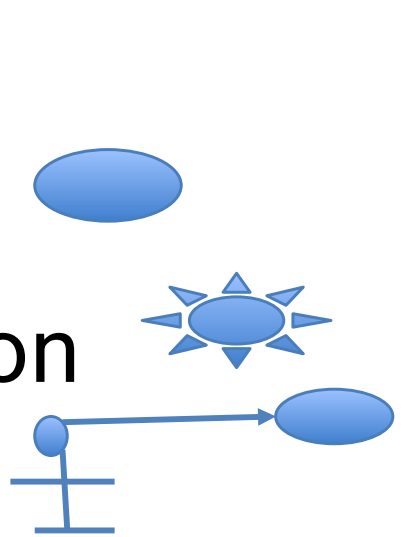
- UML vocabulary includes:
- • Things
- • Relationships
- • Diagrams

Things in UML

- Four kinds:
- • Structural
- • Behavioral
- • Grouping
- • Annotational

Structural Things (Noun)

- Static parts of a model:



- • Class
- • Interface
- • Collaboration
- • Use Case
- • Active Class
- • Component
- • Node

Behavioral Things (verb)

- Dynamic behavior:

- • Interaction (message exchange)
- • State Machine (states and transitions)

Grouping Things

- • Package

Organizes model elements

Exists only at development time

Annotational Things (explanatory part)

- • Notes

Used for comments and constraints

Relationships in UML

- Four types:
- • Dependency - - - - - >
- • Association (Aggregation) _____
- • Generalization _____|>
- • Realization ----------|>

UML Diagrams

- UML includes 9 diagrams:
- Class, (S)
- Object, (S)
- Use Case, (D)
- Sequence, (D & I)
- Collaboration, (D & I)
- Statechart, (D)
- Activity, (D)
- Component, (S)
- Deployment (S)

Static vs Dynamic Modeling

- Static:
- • Class
- • Object
- • Component
- • Deployment

- Dynamic:
- • Use Case
- • Sequence
- • Collaboration
- • Statechart
- • Activity

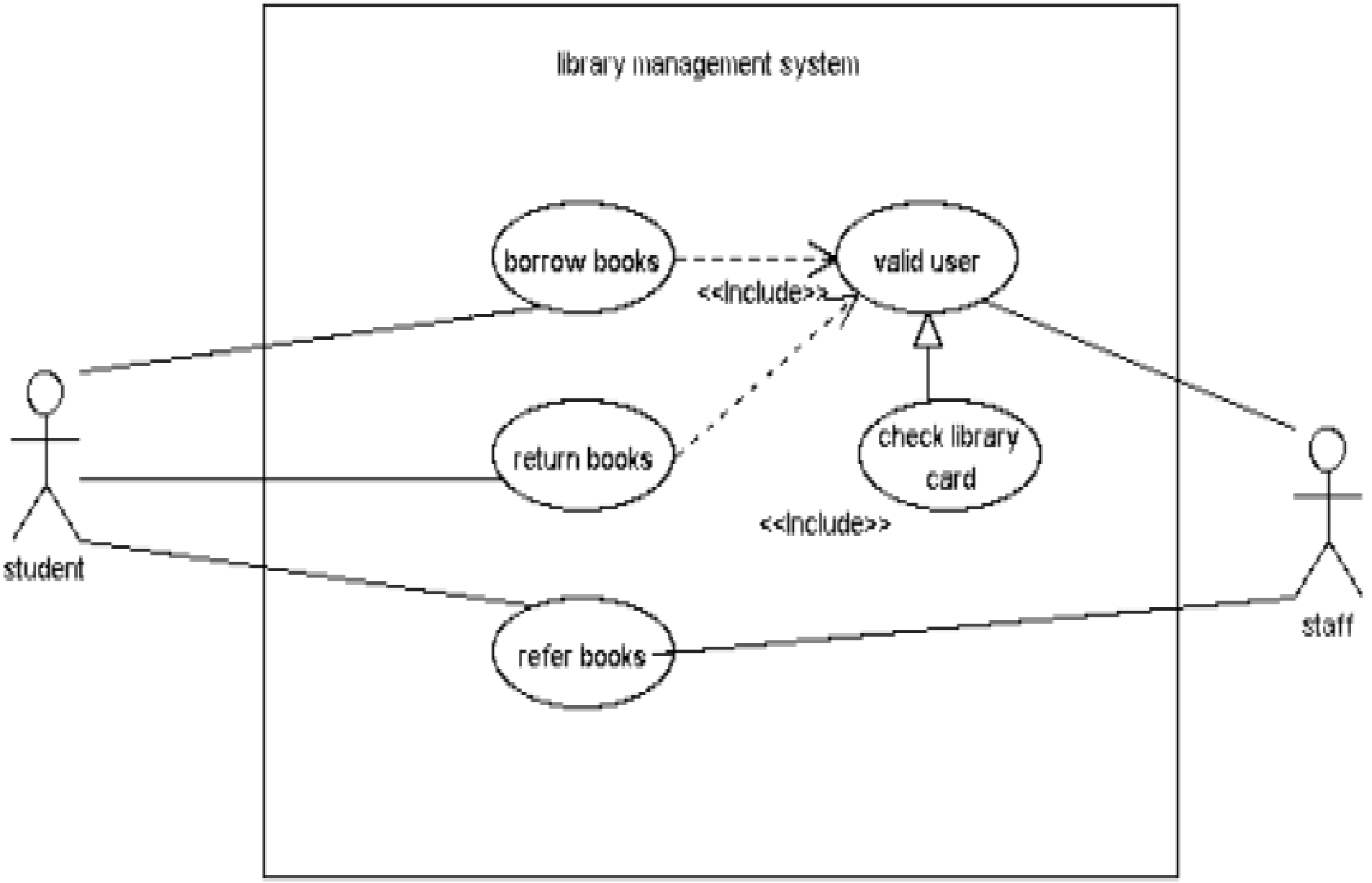
Library System Case Study

- **OBJECTIVE:**
  To draw the UML diagrams for library in which students takes and returns books.

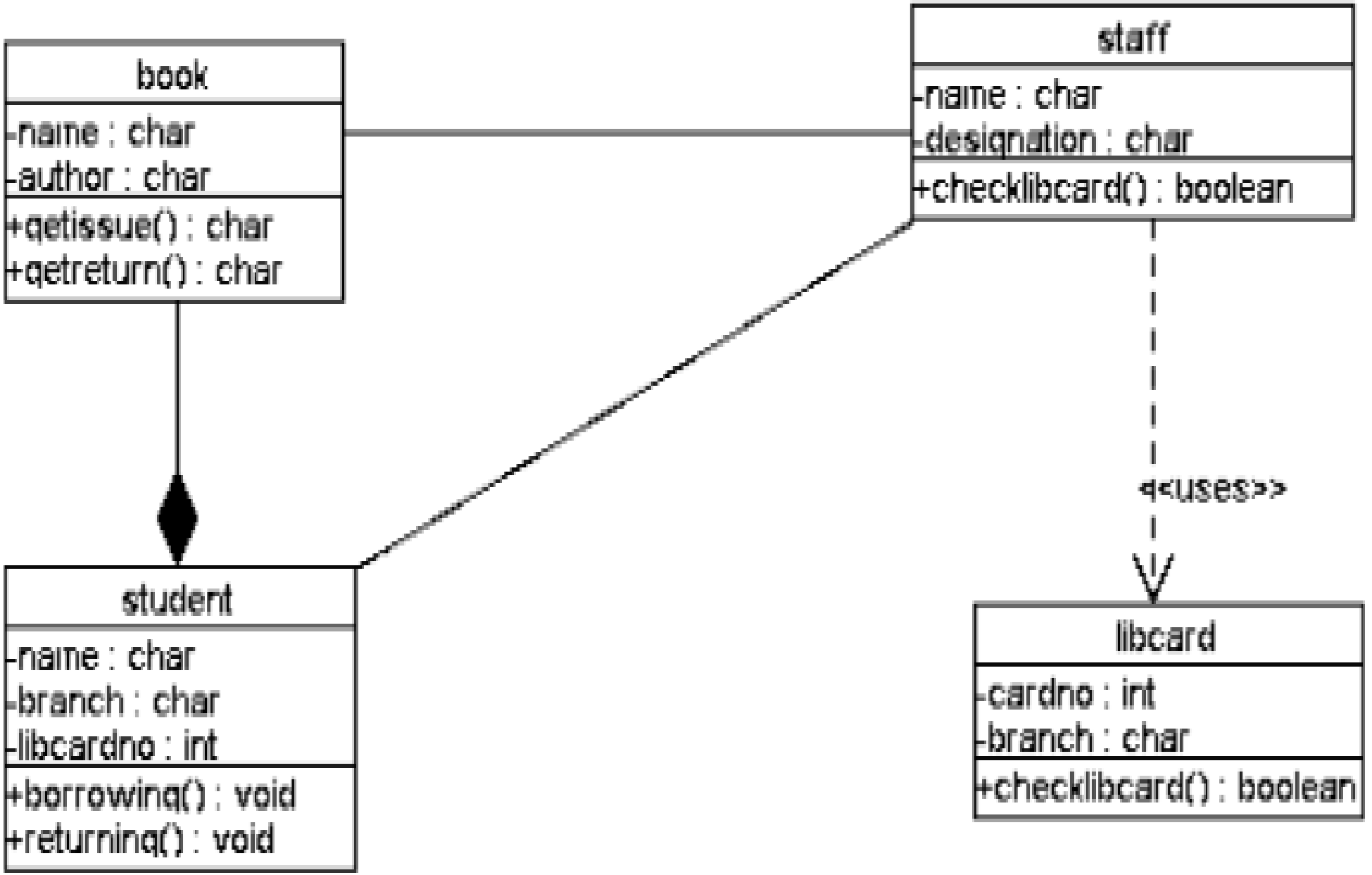
*** DESCRIPTION:**

Generally a college library consists of books and staff. Students and faculty members utilizes the library books. So, they are first made as members and are given passbooks and tickets for taking issues. Library maintains an ID for each library member. Members are allowed to take issues using their respective ID’s and return those books after prescribed time. If some books are issued or returned the library database should be updated accordingly.

USECASE DIAGRAM



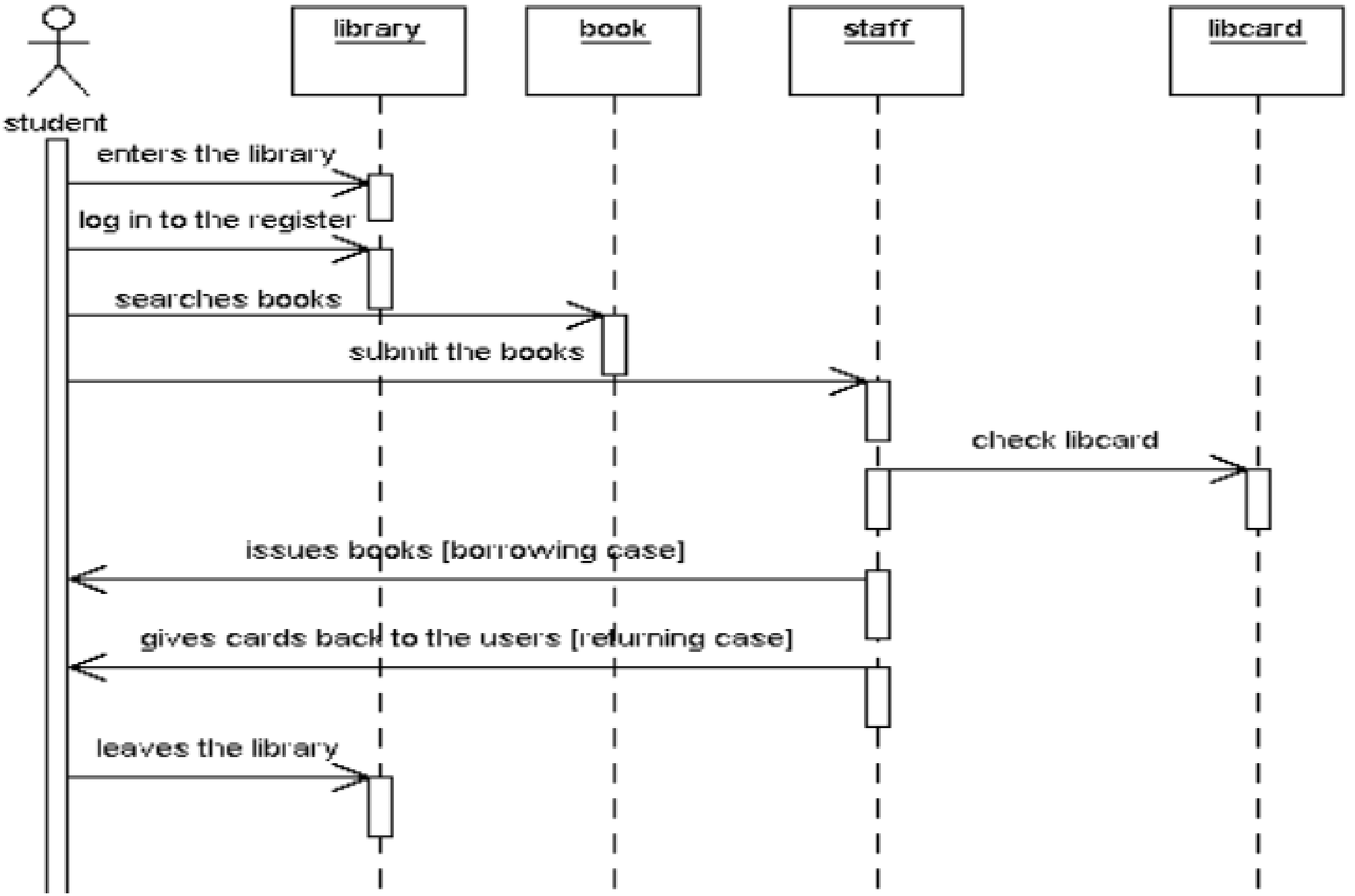
CLASS DIAGRAM



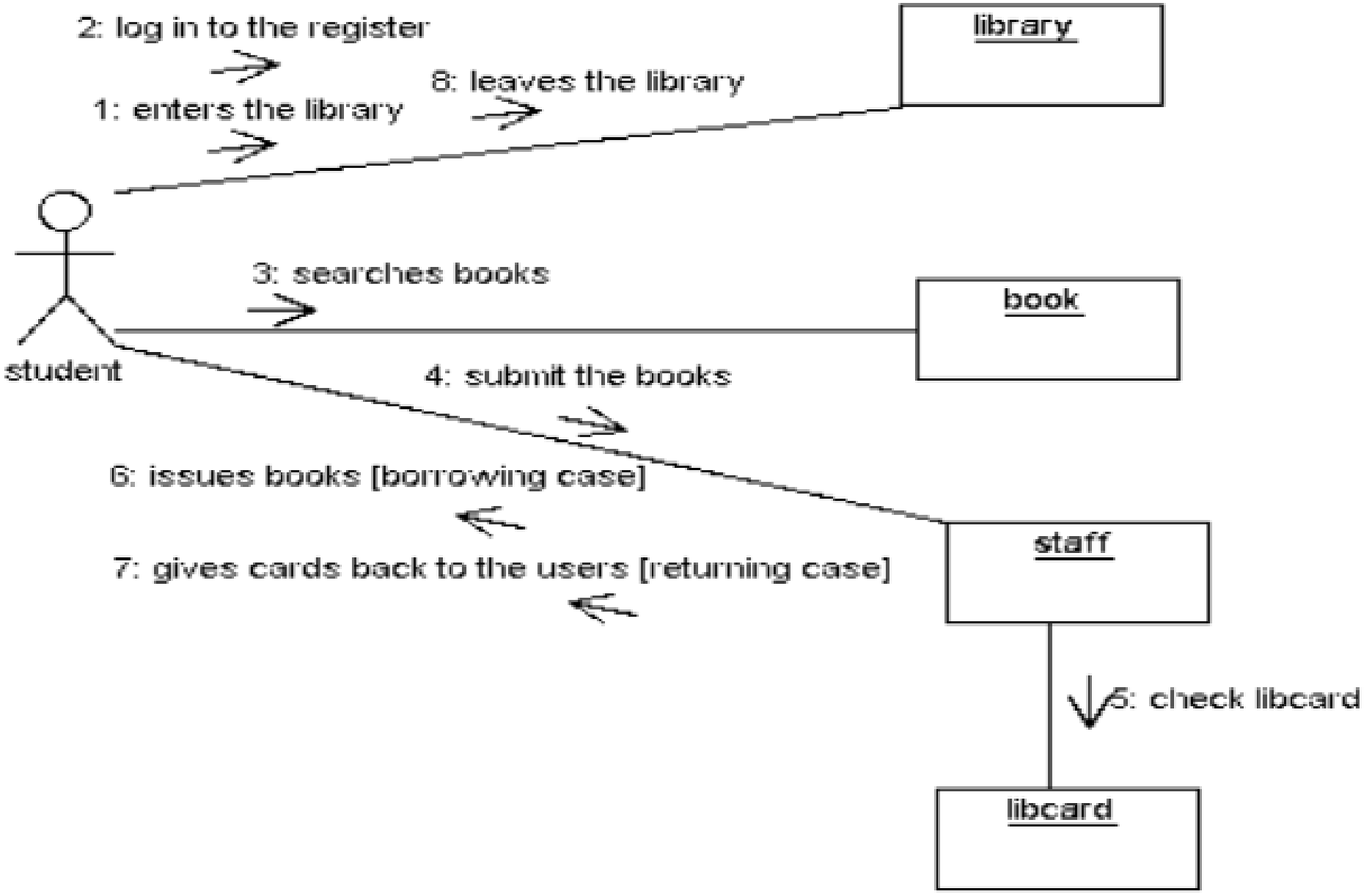
CONTINUATION

CLASS NAME	ATTRIBUTES	OPERATION
Books	name, author	get issue(), get return()
Student	Name,branch,libcardno	Borrowing(),returning()
Libcard	Cardno,branch	Checklibcard()
Staff	Name,designation	Checklibcard()

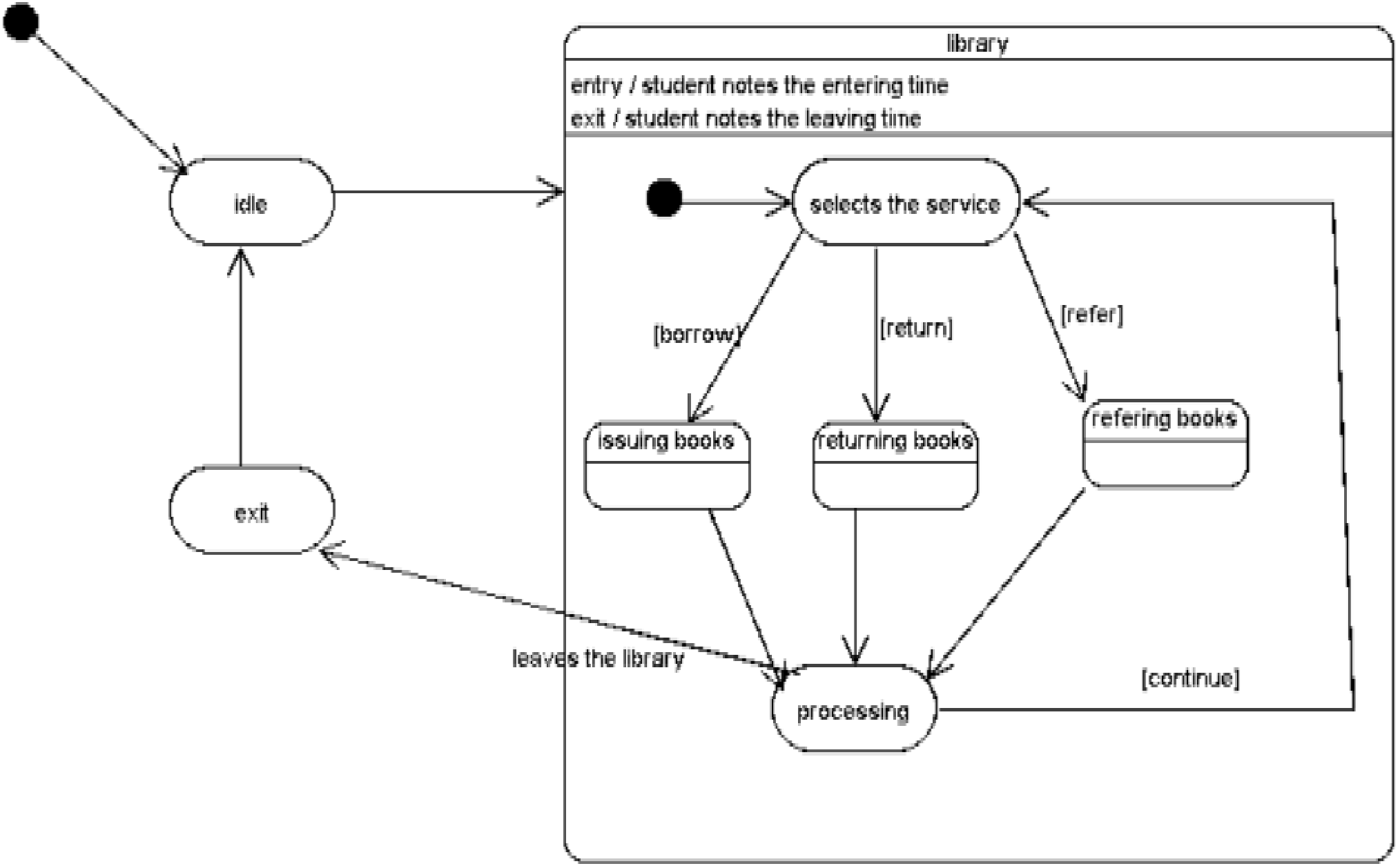
SEQUENCE DIAGRAM



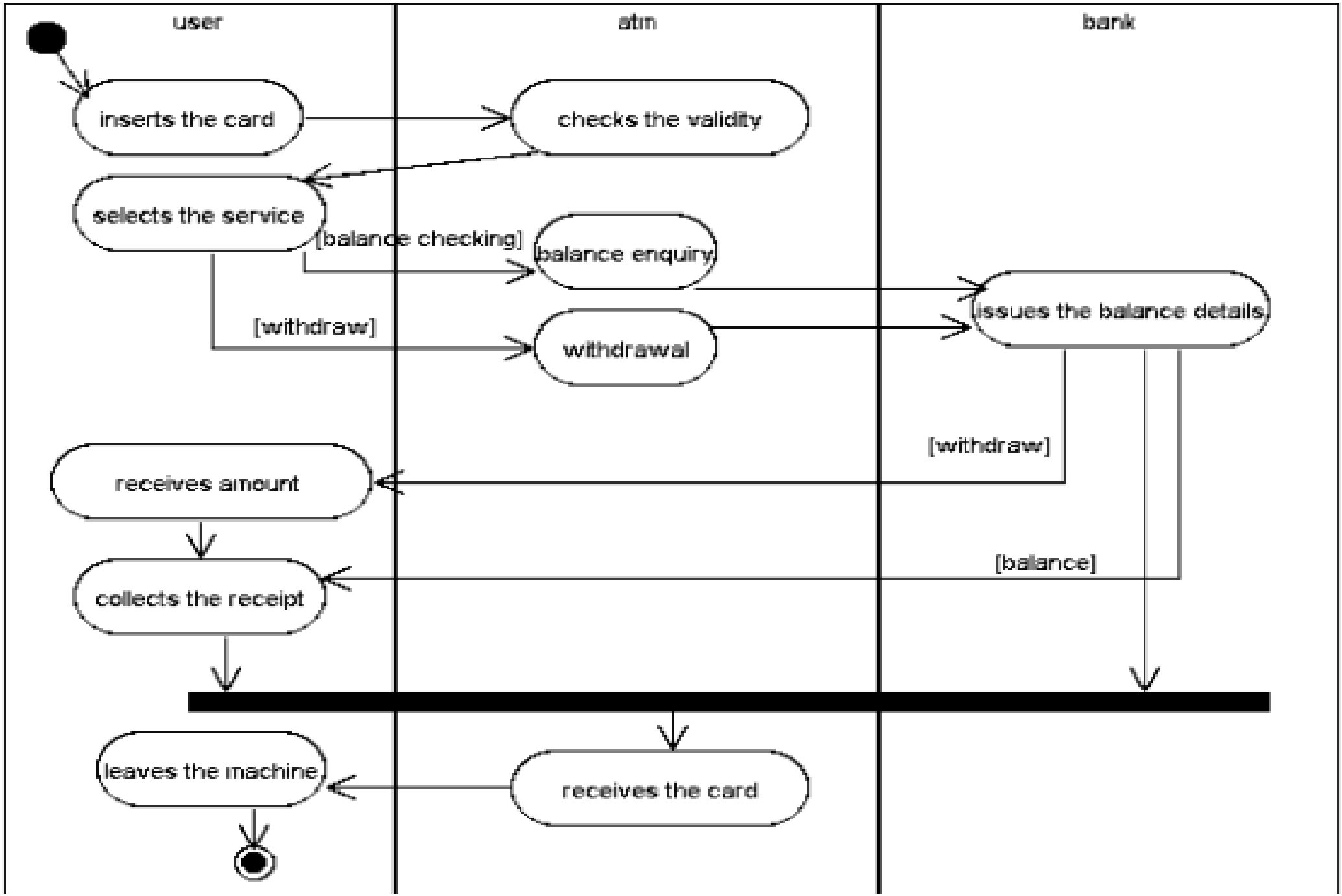
COLLABORATION DIAGRAM



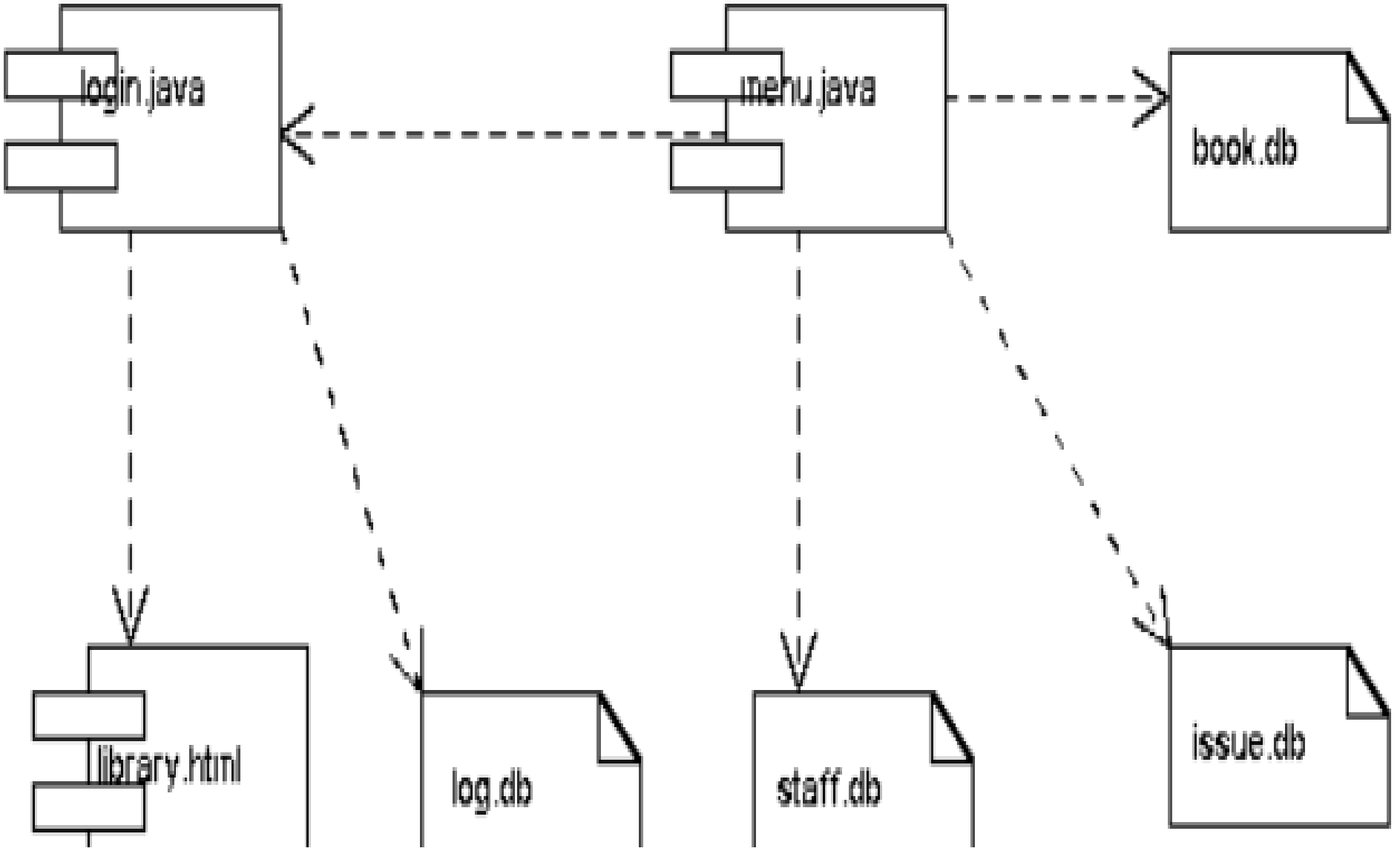
STATE DIAGRAM



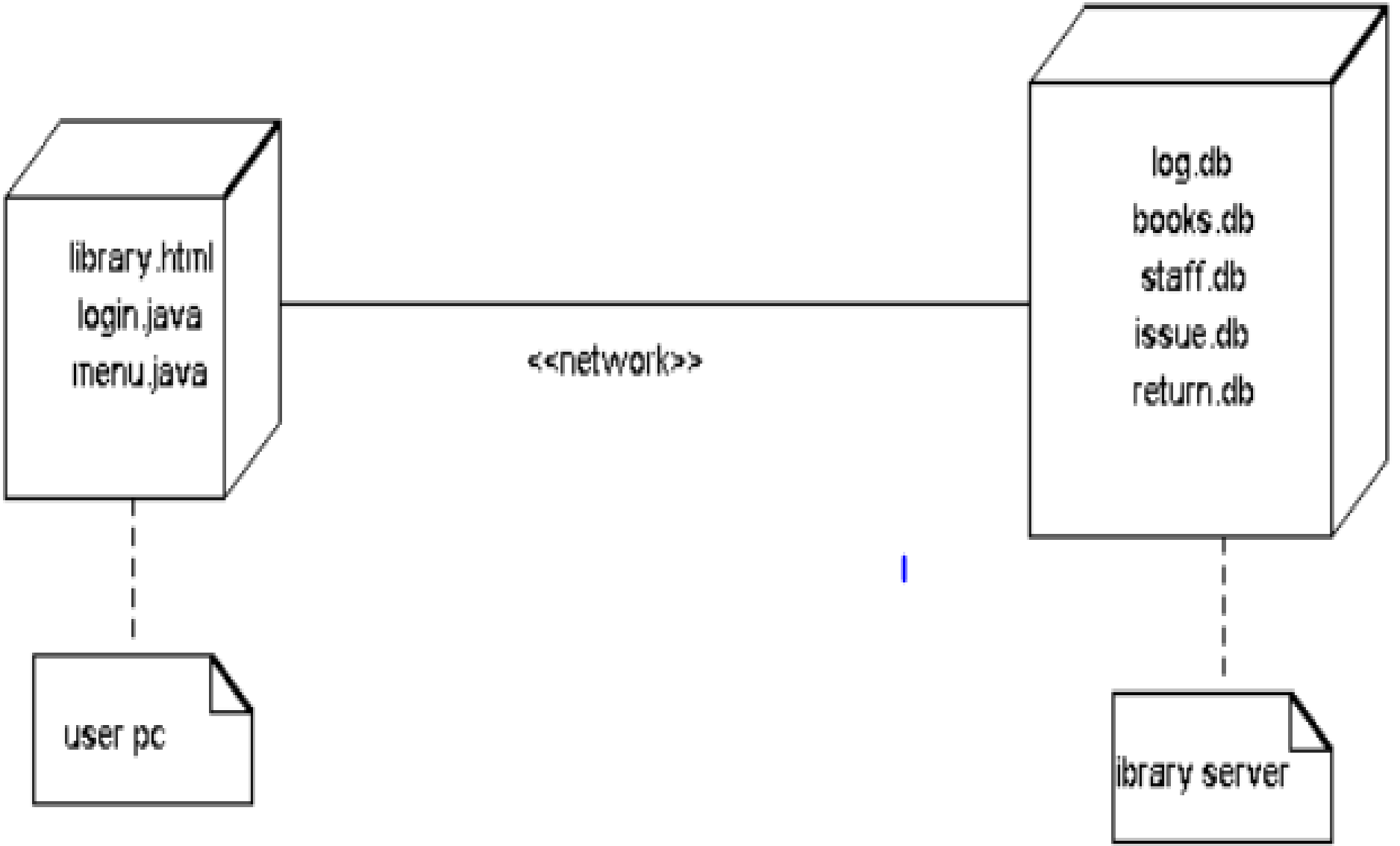
Activity Diagram



Component Diagram



Deployment Diagram



ATM

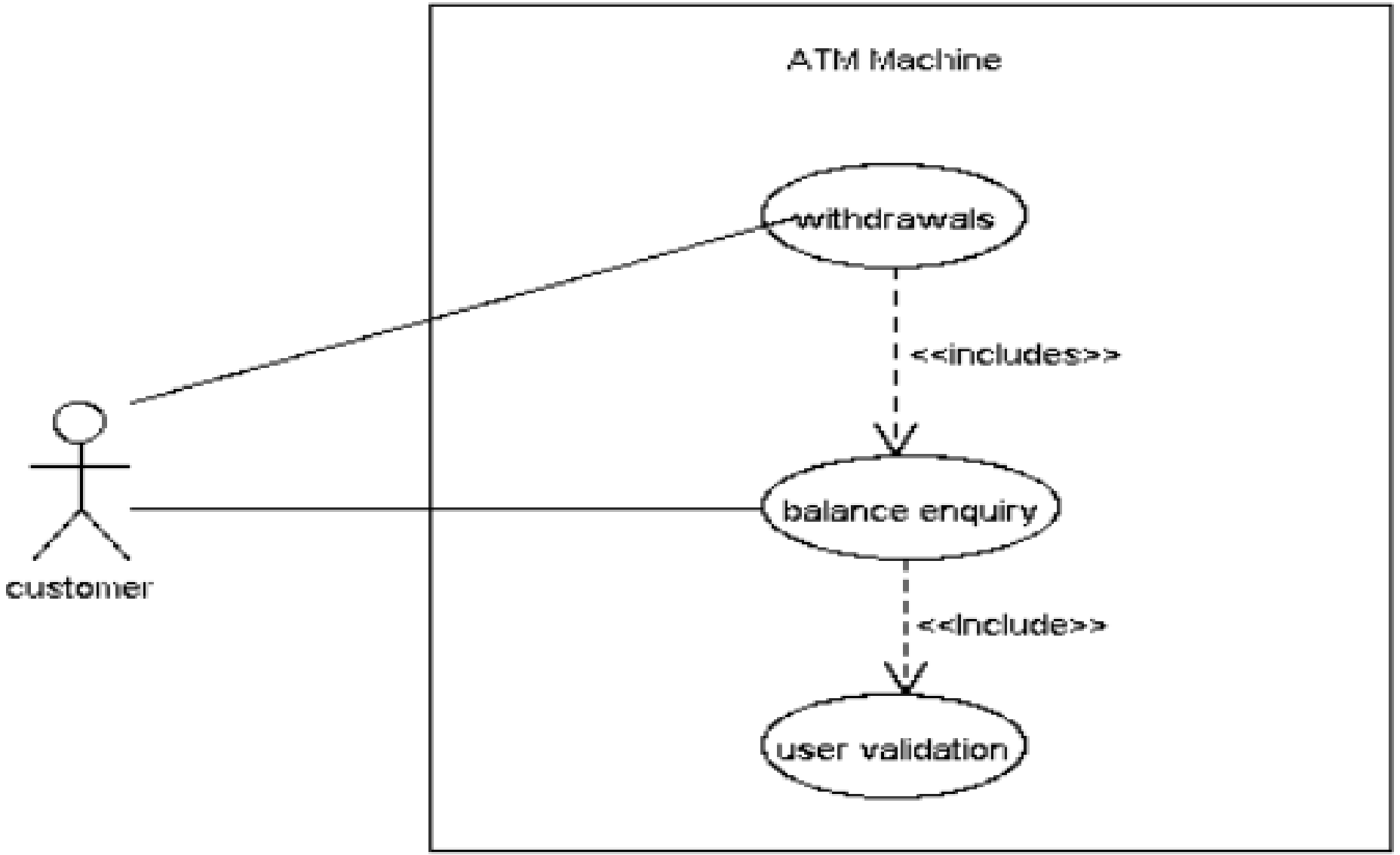
States involved:

user, idle, active, selecting services, processing. Selecting service
as substates-withdrawl balance enquiry

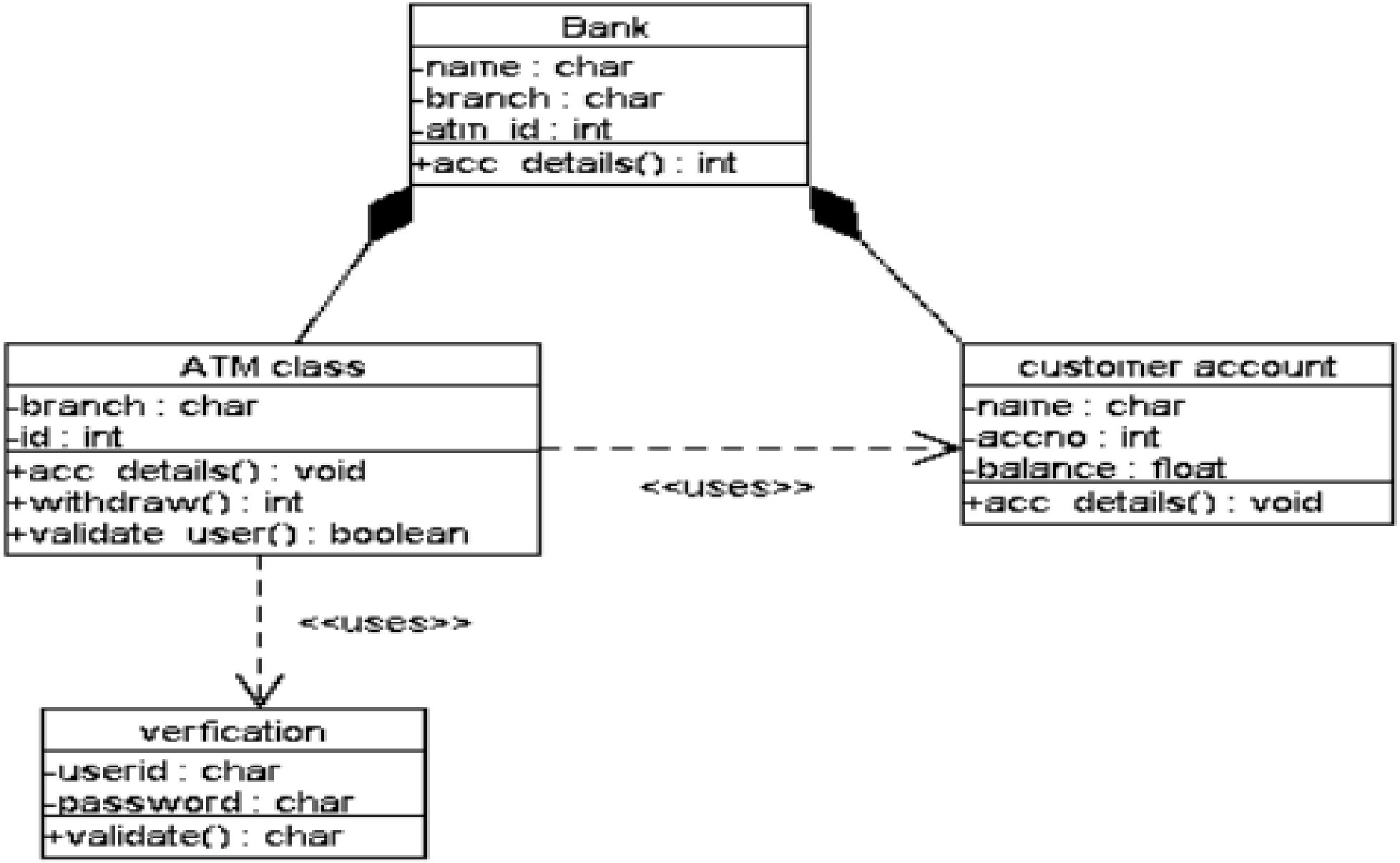
Description:

Initially, user is in idle state, and then he will insert the card and becomes active. He enters the pin number and selects the service he needs to take. If he need to withdrawl, he enters the amount to be debited and he receives the amount and then he receives the card and quits from ATM. If he need the balance enquires, ATM process and gives the receipt having information regarding balance.

ATM MACHINE


ATM Machine
withdrawals
<<includes>>
balance enquiry
<<Include>>
user validation
customer

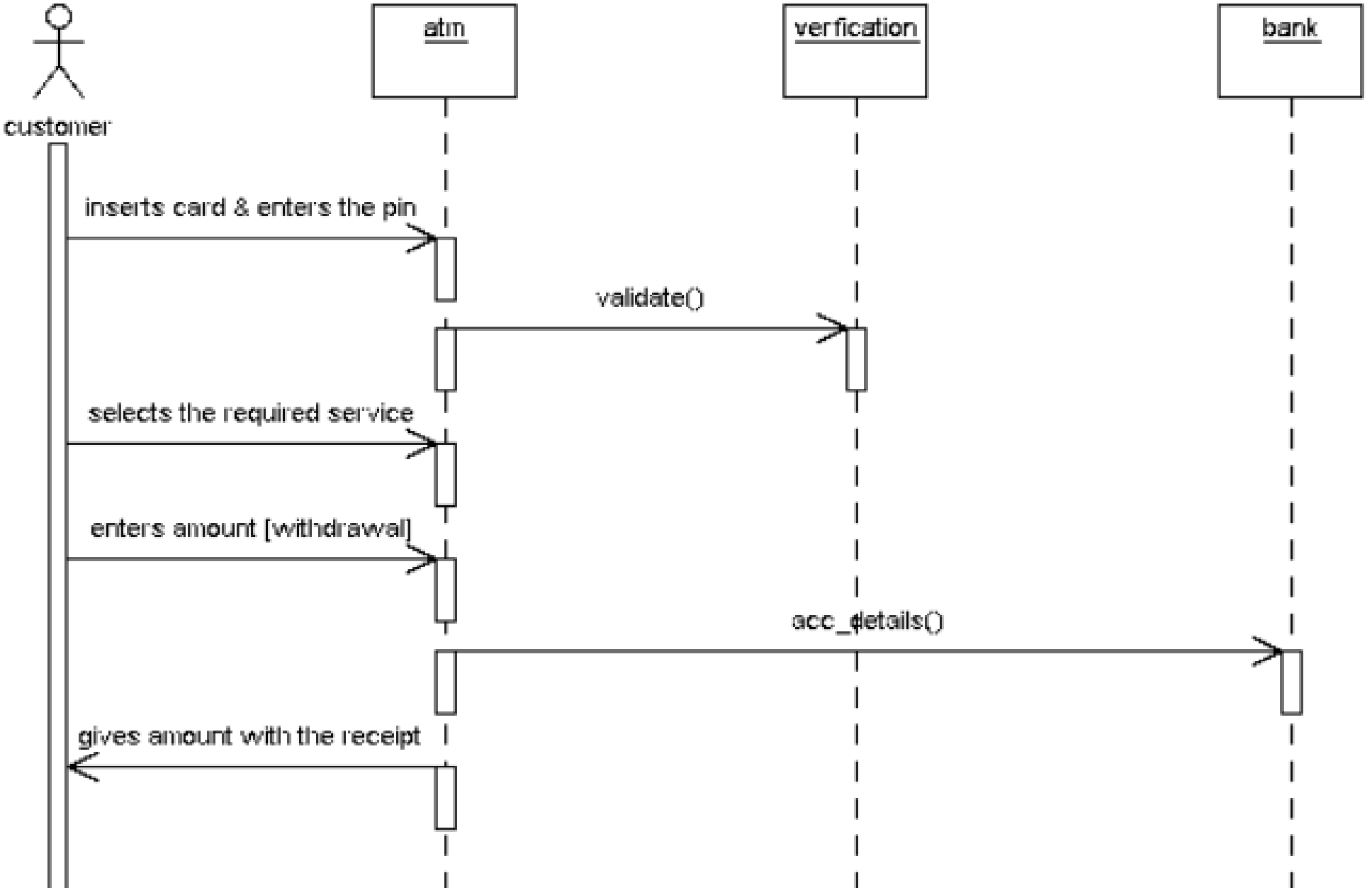
CLASS DIAGRAM



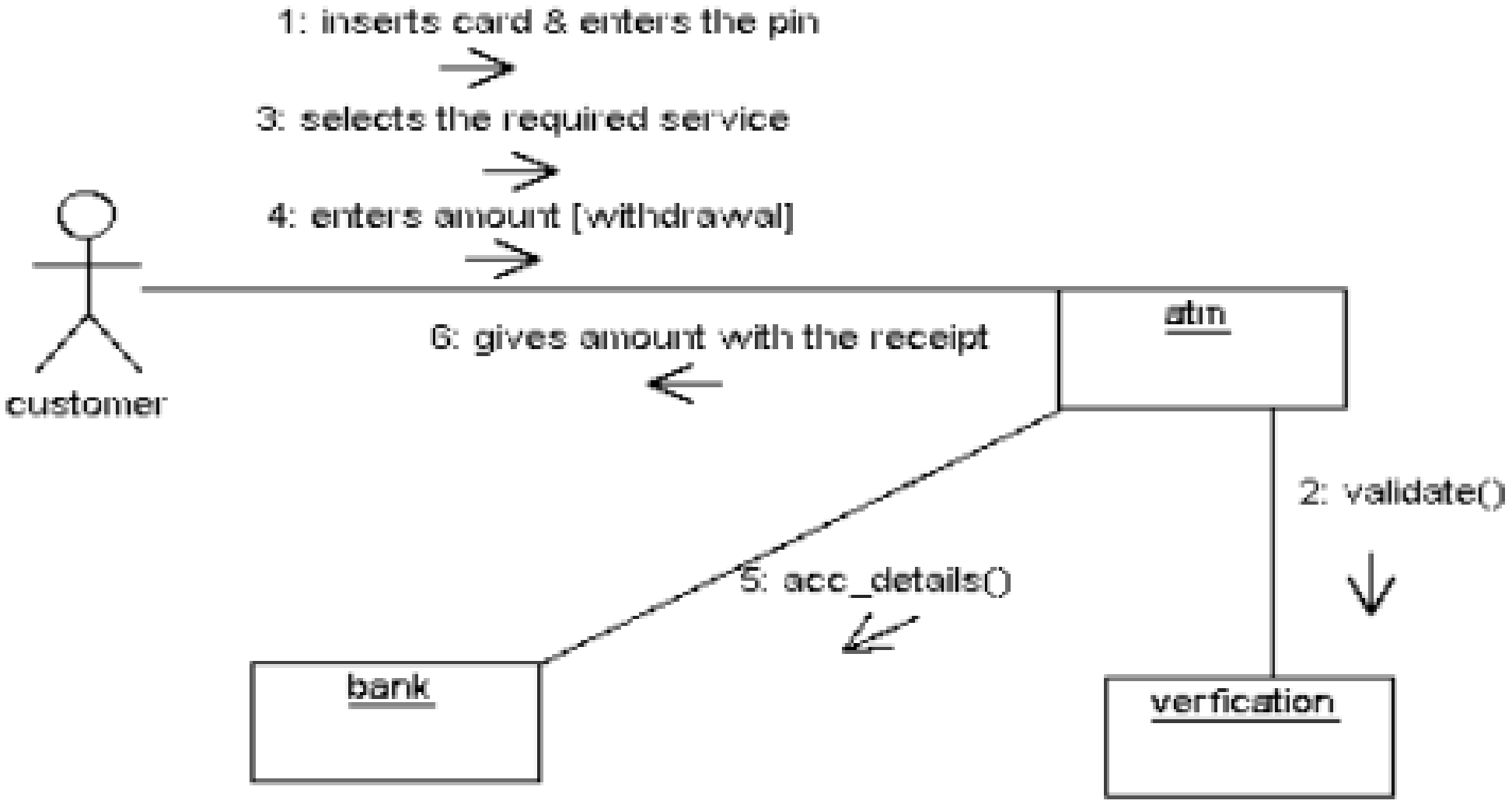
CONTINUATION

Class name	Attributes	Operation
Bank	Name, branch, atmid	Acc details()
ATM class	Branch, id	Accdetails(),withdraw(),validate user()
Customer account	Name, accno, balance	Acc details()

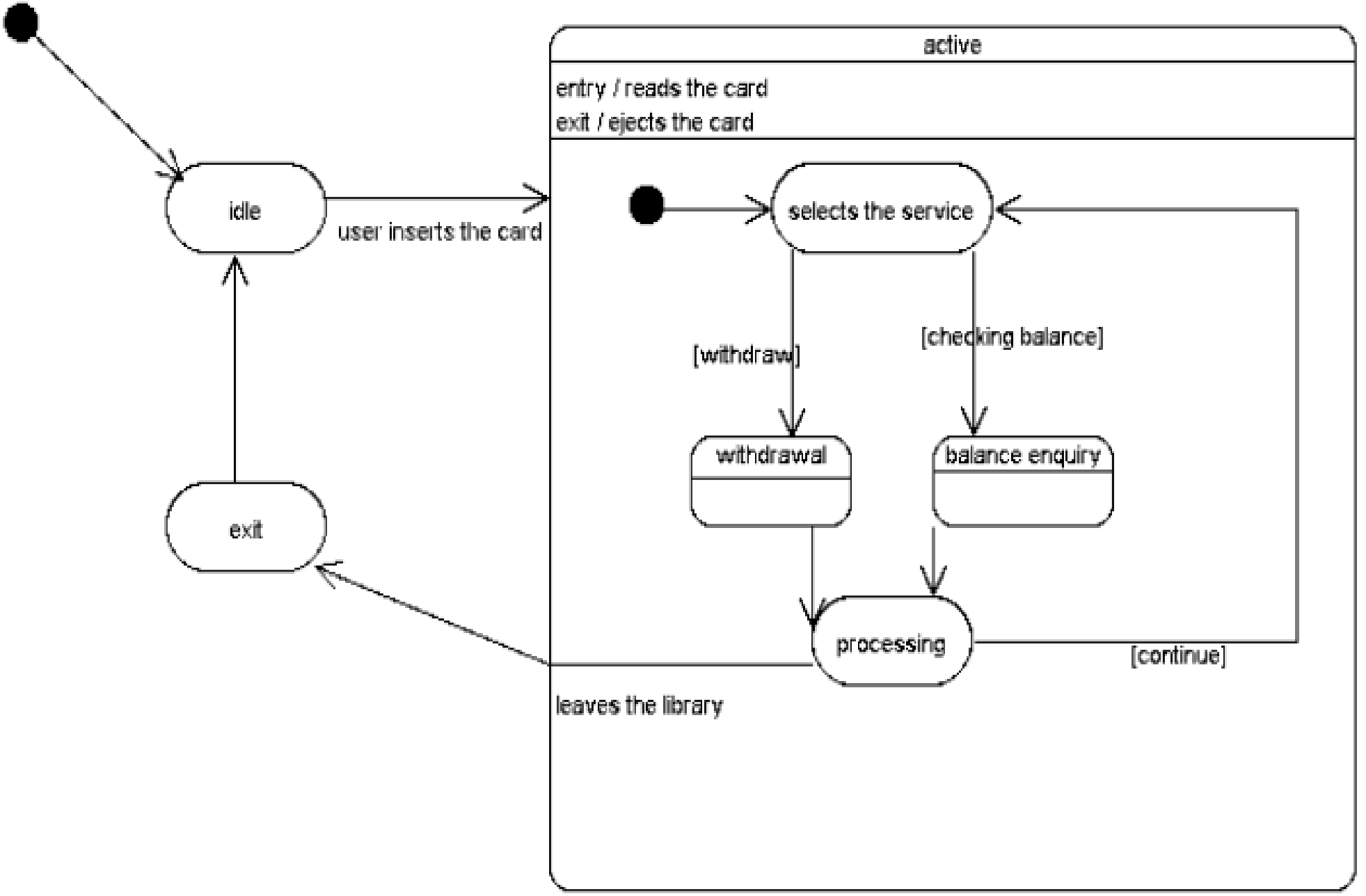
SEQUENCE DIAGRAM



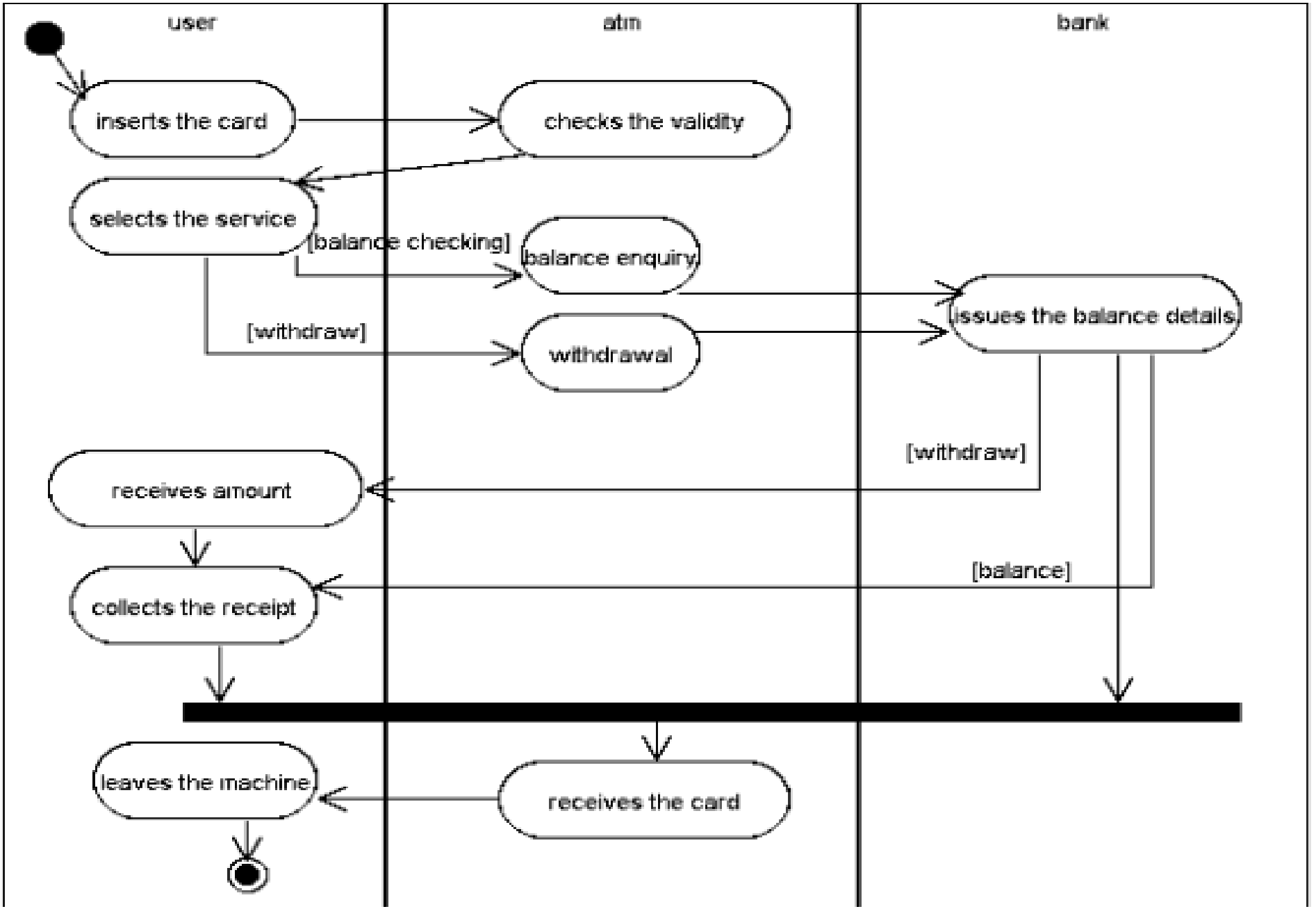
COLLABORATION DIAGRAM



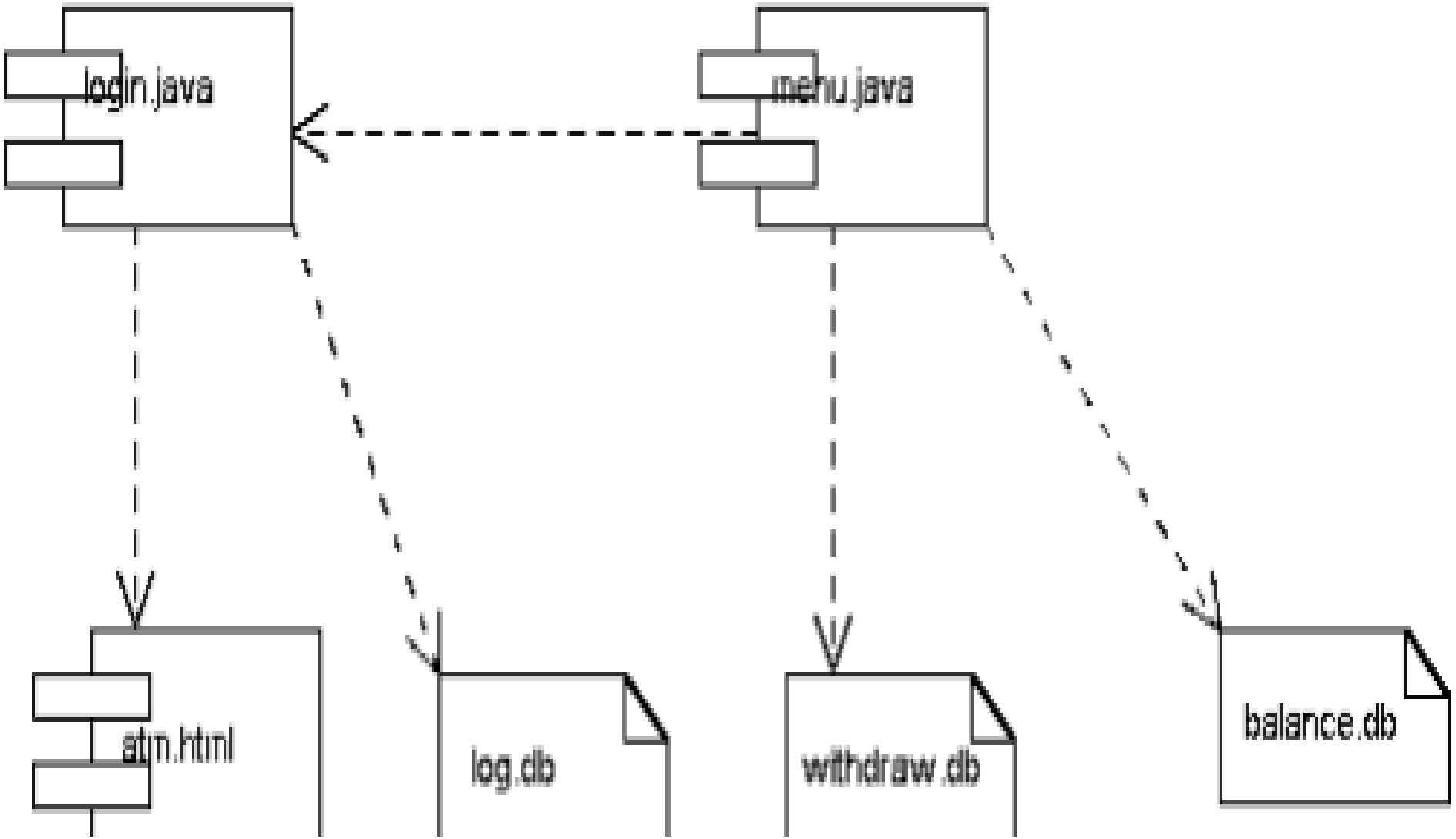
STATE DIAGRAM



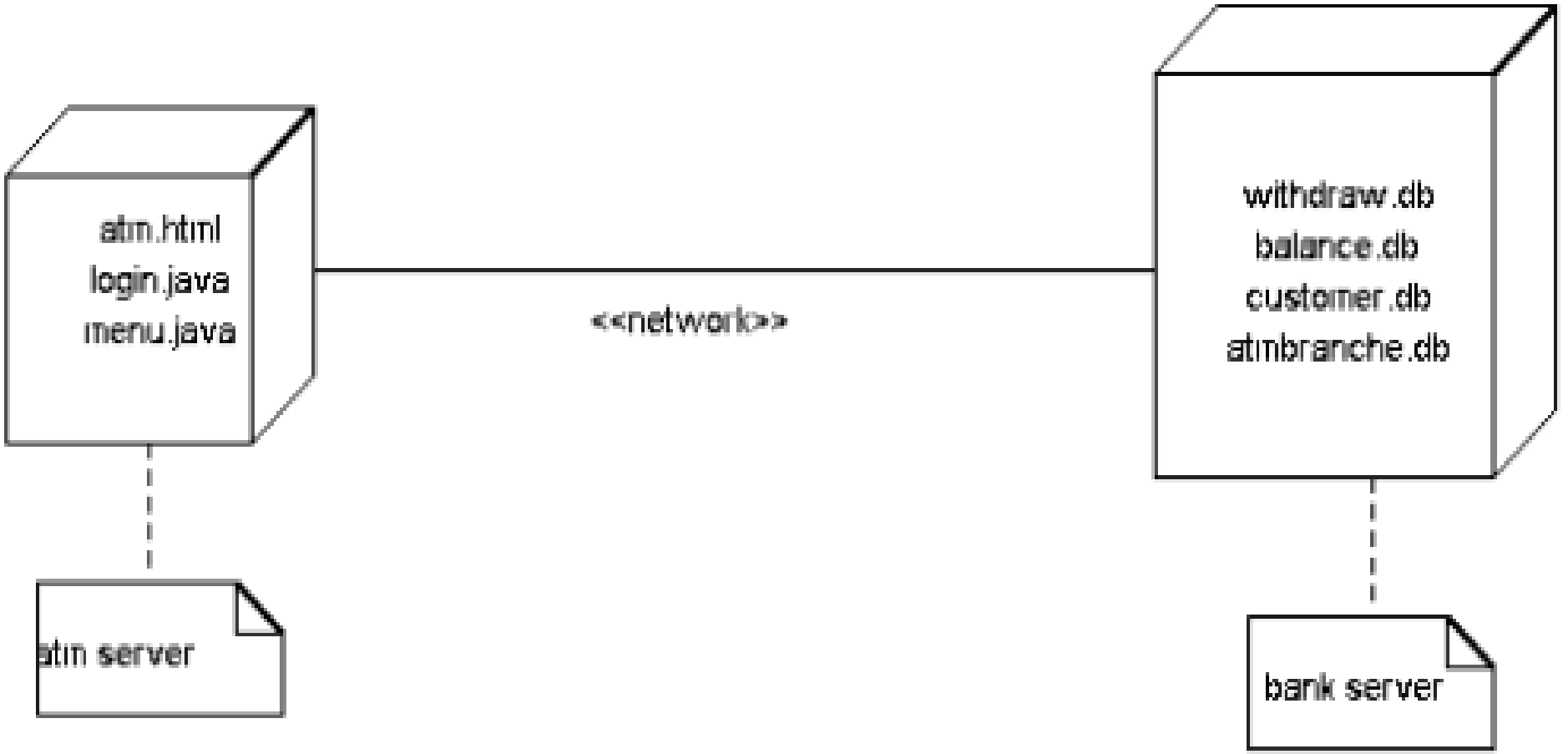
ACTIVITY DIAGRAM



Component Diagram



Deployment Diagram



Rules of UML

- Models must be well-formed:
- • Names
- • Scope
- • Visibility
- • Integrity
- • Execution

Common Mechanisms

- • Specifications
- • Adornments
- • Common divisions
- • Extensibility mechanisms

Extensibility Mechanisms

- • Stereotypes (<< >>)
- • Extend UML vocabulary
- • Used for classes, components, relationships

Key Definitions

- System: Organized subsystems for a purpose
- Subsystem: Group of related elements
- Model: Simplified abstraction of reality
- View: Projection of a system model