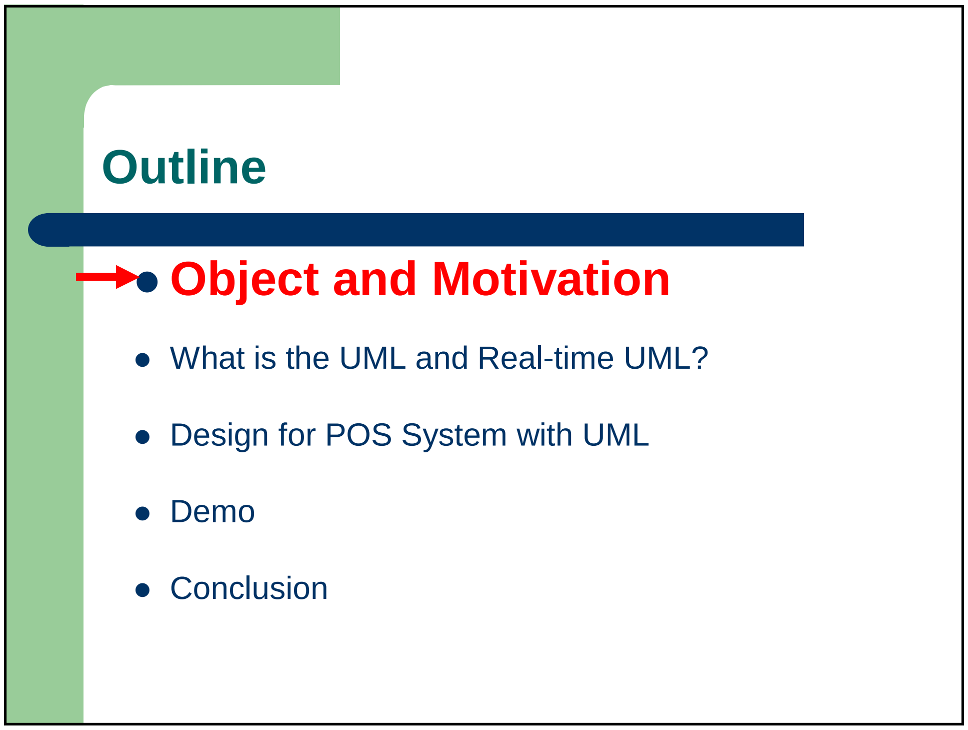
The research for Real-time UML and Development of POS System

Sumiyasu Ikeda

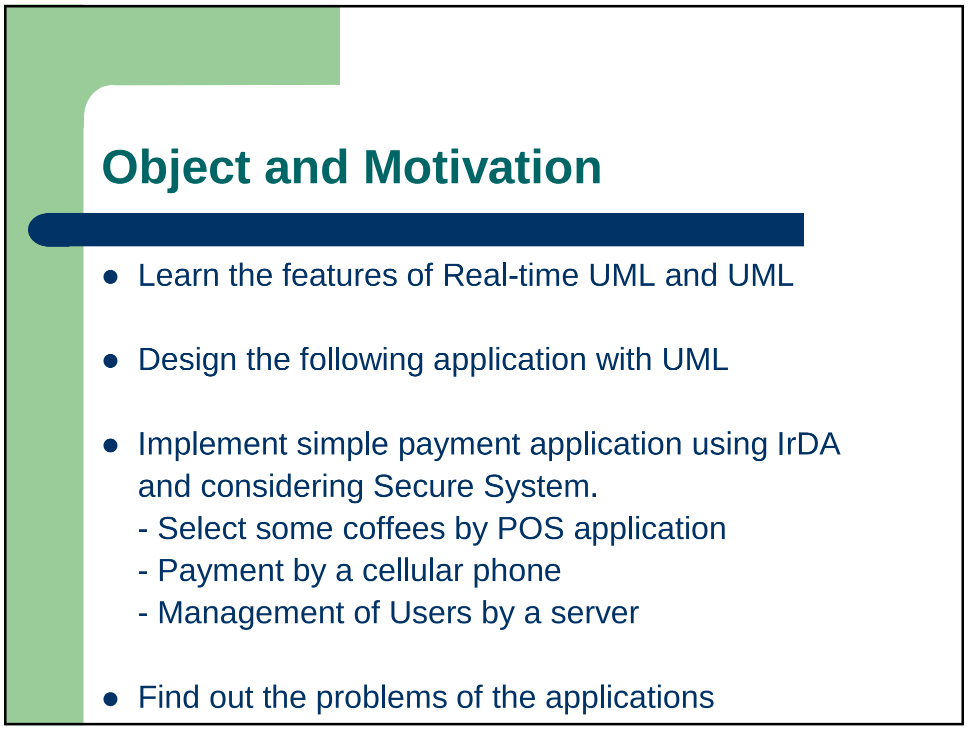
Outline

- Object and Motivation
- What is the UML and Real-time UML?
- Design for POS System with UML
- Demo
- Conclusion



Outline

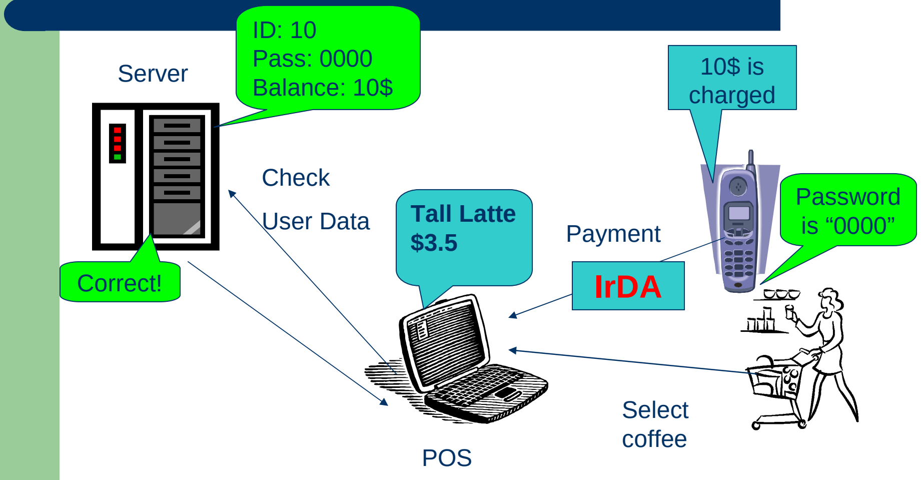
- **Object and Motivation**
- What is the UML and Real-time UML?
- Design for POS System with UML
- Demo
- Conclusion



Object and Motivation

- Learn the features of Real-time UML and UML
- Design the following application with UML
- Implement simple payment application using IrDA and considering Secure System.
  - Select some coffees by POS application
  - Payment by a cellular phone
  - Management of Users by a server
- Find out the problems of the applications

Overview of POS System



Why choose IrDA?

- IrDA is abbr. for Infrared Data Association.
- can send data **without a fee**
  - HTTP communication impose us to pay fee.
- Many cellular phone has IrDA, but there are a few application which use IrDA.

Outline

- ✓ Object and Motivation
- ➔ ● **What is the UML and Real-time UML?**
- Design for POS System with UML
- Demo
- Conclusion

What is the UML?

- abbr. for Unified Modeling Language.
- for specifying, visualizing, constructing, and documenting the artifacts of software system, as well as for business modeling and other non-software systems.
- a collection of best engineering practices.
- has 13 types of notation (version 2.0).

UML Diagrams

Activity Diagram	Depicts high-level business processes, including data flow, or to model the logic of complex logic within a system.
Class Diagram	Shows a collection of static model elements such as classes and types, their contents, and their relationships.
Communication Diagram	Shows instances of classes, their interrelationships, and the message flow between them. Communication diagrams typically focus on the structural organization of objects that send and receive messages.
Component Diagram	Depicts the components that compose an application, system, or enterprise. The components, their interrelationships, interactions, and their public interfaces are depicted.

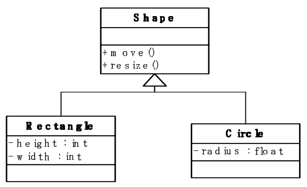
UML Diagrams *(cont.)*

Composite Structure Diagram	Depicts the internal structure of a classifier (such as a class, component, or use case), including the interaction points of the classifier to other parts of the system.
Deployment Diagram	Shows the execution architecture of systems. This includes nodes, either hardware or software execution environments, as well as the middleware connecting them.
Interaction Overview Diagram	A variant of an activity diagram which overviews the control flow within a system or business process. Each node/activity within the diagram can represent another interaction diagram.
Object Diagram	Depicts objects and their relationships at a point in time, typically a special case of either a class diagram or a communication diagram.

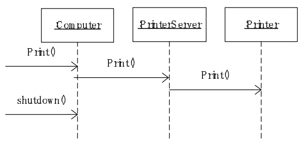
UML Diagrams (*cont.)*

Package Diagram	Shows how model elements are organized into packages as well as the dependencies between packages.
Sequence Diagram	Models the sequential logic, in effect the time ordering of messages between classifiers.
State Machine Diagram	Describes the states an object or interaction may be in, as well as the transitions between states.
Timing Diagram	Depicts the change in state or condition of a classifier instance or role over time. Typically used to show the change in state of an object over time in response to external events.
Use Case Diagram	Shows use cases, actors, and their interrelationships.

Examples of UML diagrams

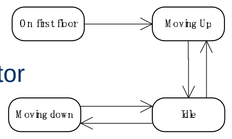


Class diagram

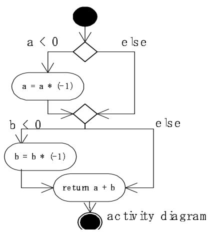


Sequence Diagram

Elevator



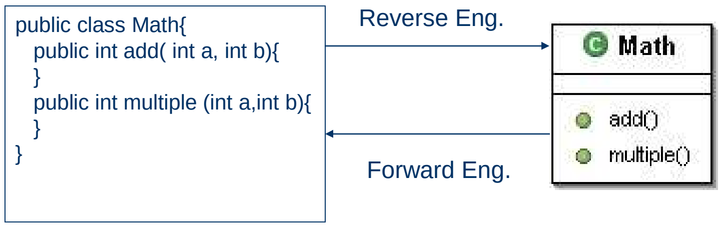
State chart Diagram



Activity Diagram

Other features

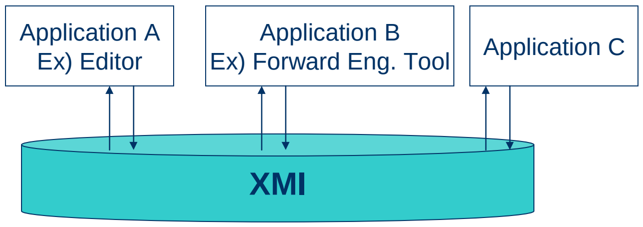
- Forward and Reverse Engineering
  - Forward Engineering:
    Generating source code from diagrams
  - Reverse Engineering:
    Generating diagrams from source code

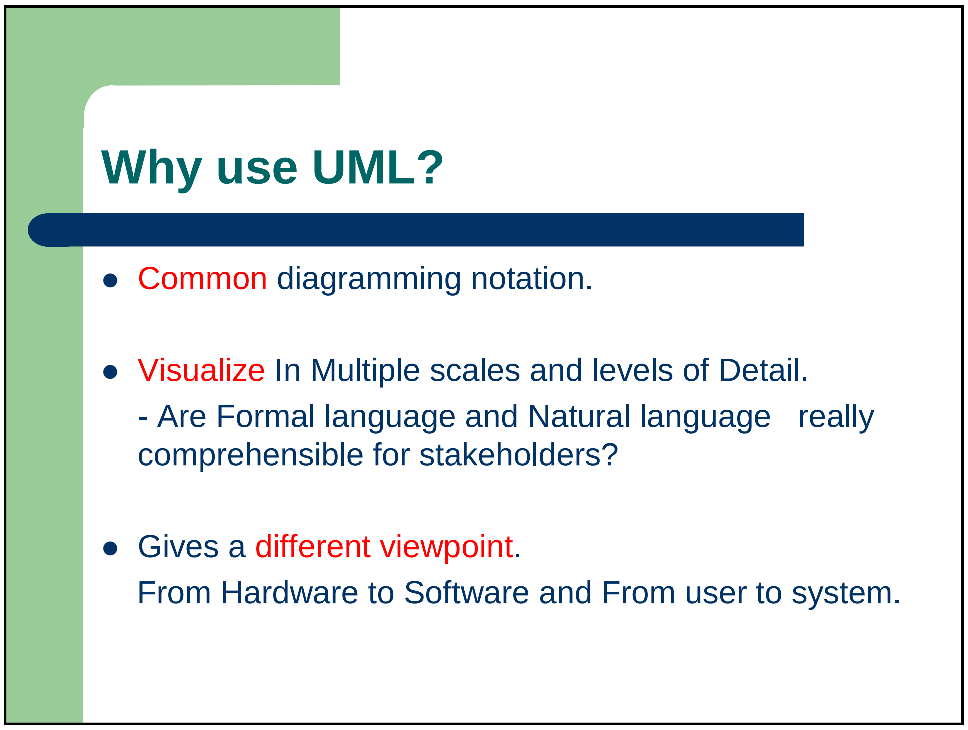


```
public class Math{
   public int add( int a, int b){
   }
   public int multiple (int a,int b){
   }
}
```

Other features *(cont)*

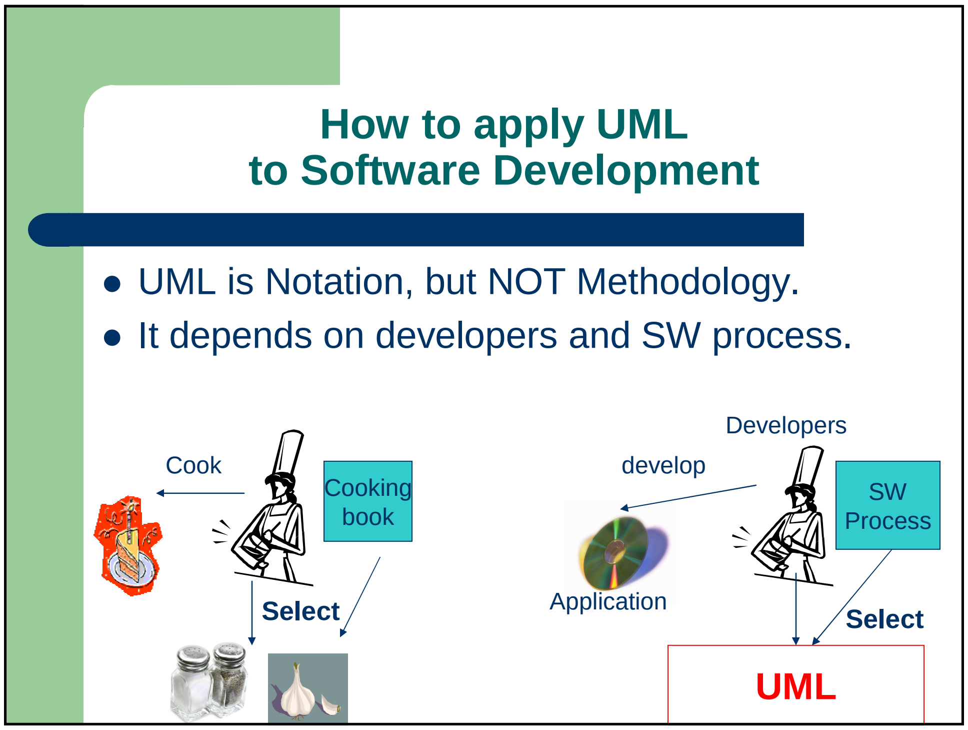
- XMI (XML Metadata Interchange)
  is standard XML format used to export UML diagrams into XML.
- Each application share the defined UML diagrams.





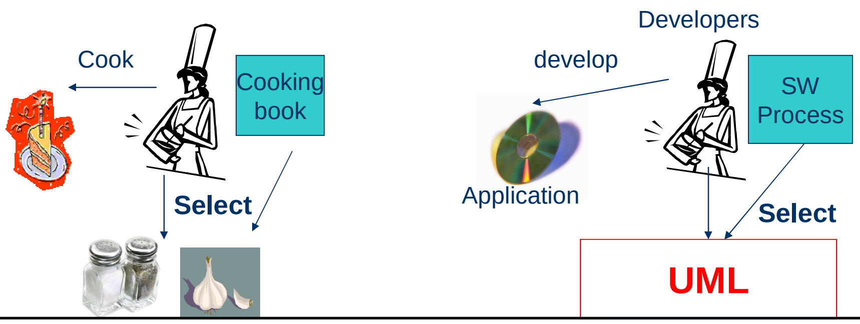
Why use UML?

- Common diagramming notation.
- Visualize In Multiple scales and levels of Detail.
  - Are Formal language and Natural language really comprehensible for stakeholders?
- Gives a different viewpoint.
  From Hardware to Software and From user to system.



How to apply UML to Software Development

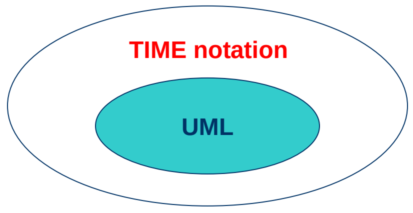
- UML is Notation, but NOT Methodology.
- It depends on developers and SW process.



What is the Real-time UML?

- Focus on Real-time application, such as RTOS and device SW.
- Current UML, version 1.5, does NOT have a notation about TIME.
  → Add TIME notation into UML = Real-time UML

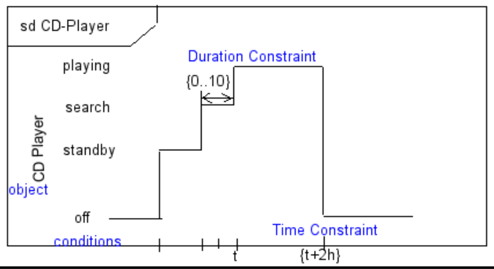
Real-time UML

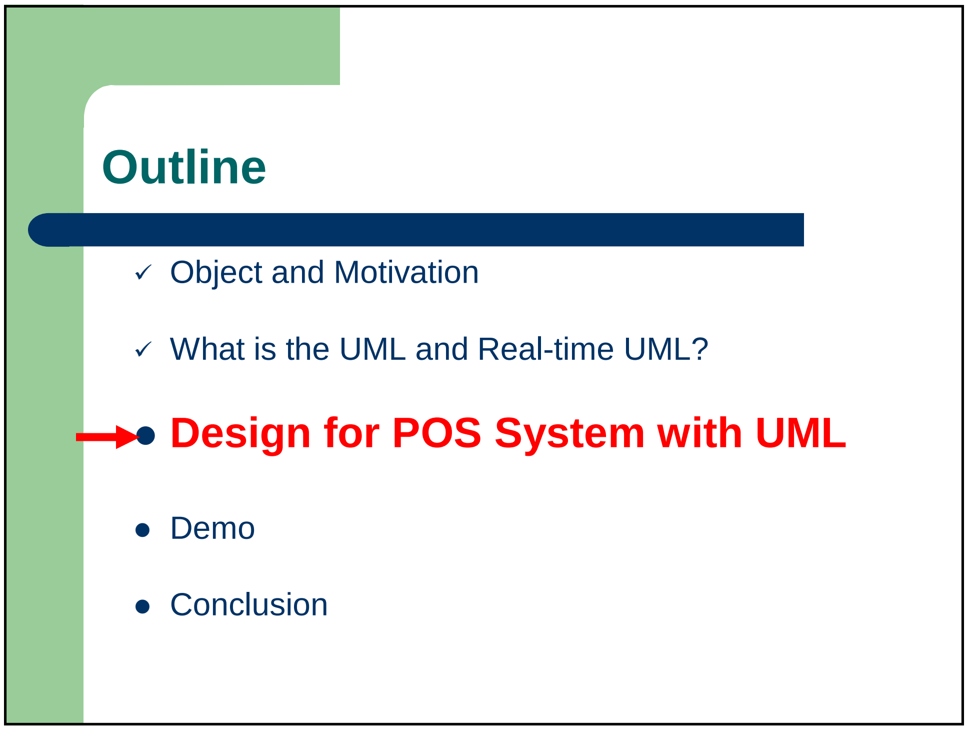


TIME notation of Real-time UML

- Timing Diagram (will be included in UML2.0)

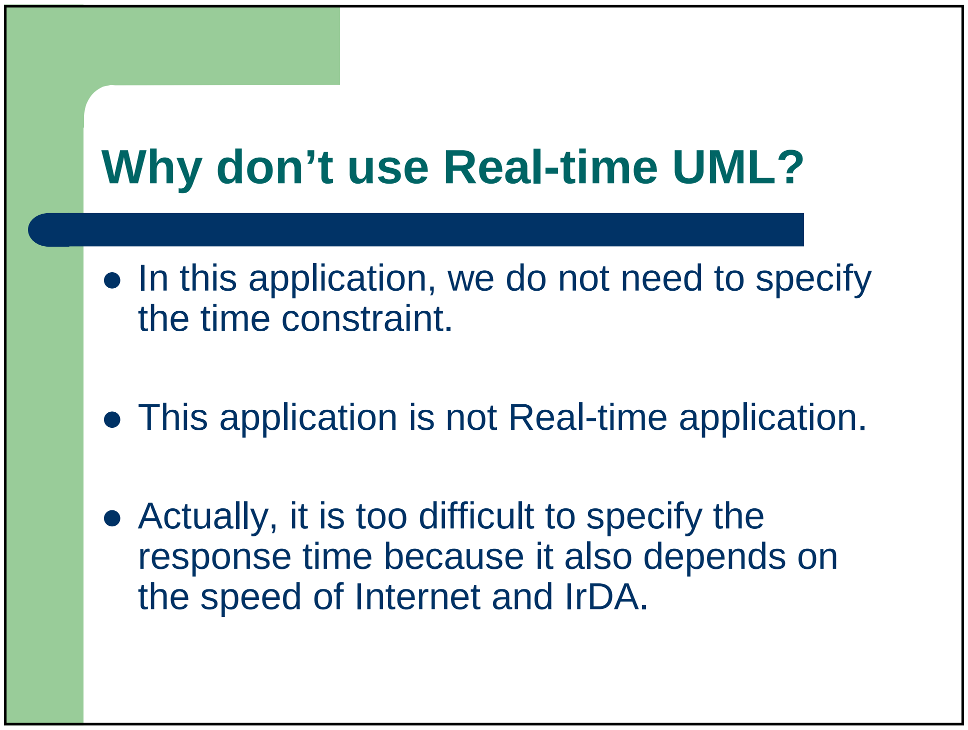
1. User turn on the CD-player.
2. User insert CD.
3. CD-Player searches music and play.
4. After 2 hours passes, CD-player is turned off automatically





Outline

- ✓ Object and Motivation
- ✓ What is the UML and Real-time UML?
- → **Design for POS System with UML**
- Demo
- Conclusion

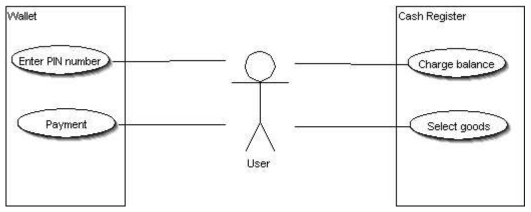


Why don’t use Real-time UML?

- In this application, we do not need to specify the time constraint.
- This application is not Real-time application.
- Actually, it is too difficult to specify the response time because it also depends on the speed of Internet and IrDA.

Use Case Diagram

- Shows use cases, actors, and their interrelationships from **view of USER**.
- For example, “Check PIN number” is the function from view of the system.



Use Case Diagram *(cont)*

- Only use case diagram is very ambiguous, so we need to write explanations with NL.

Example:

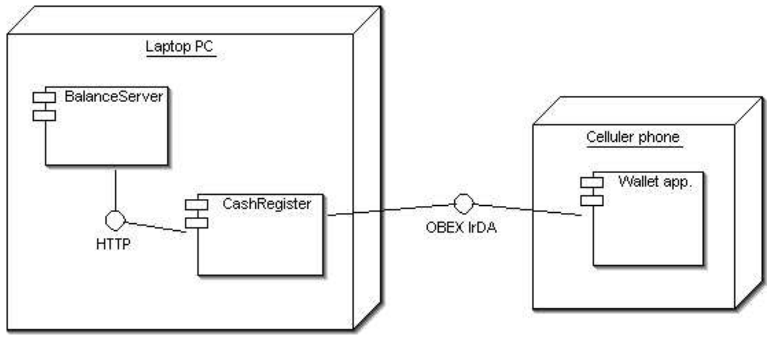
Use Case – Enter PIN number:

This use case is used to avoid dishonest access. For COMFIRMATION.

1. User enters PIN number
2. If correct, wallet app. shows Success message.
3. If not, it shows failure message.

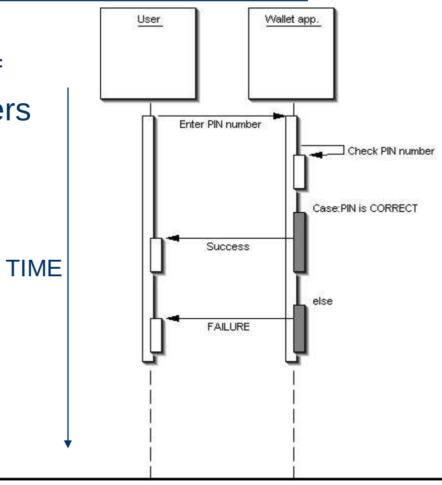
Deployment Diagram

- Shows the execution architecture of systems

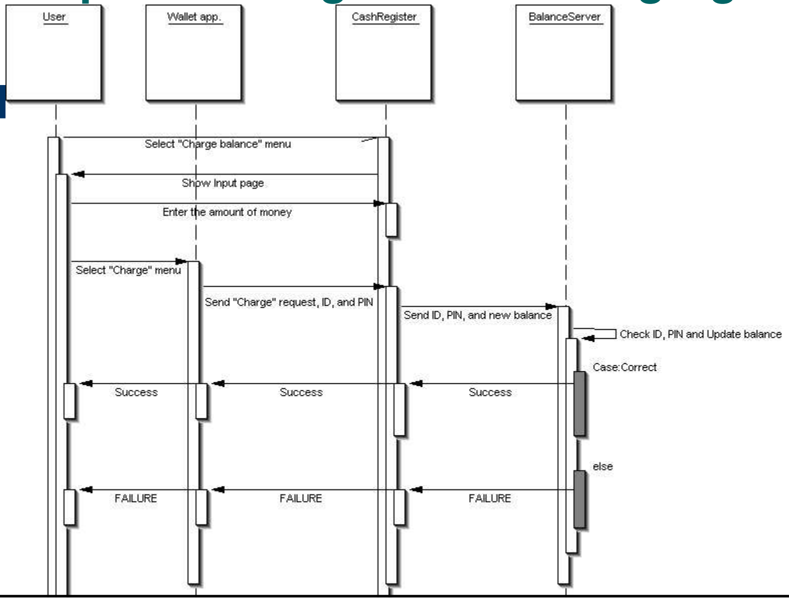


Sequence Diagram for checking PIN

Models the sequential logic, in effect the time ordering of messages between classifiers

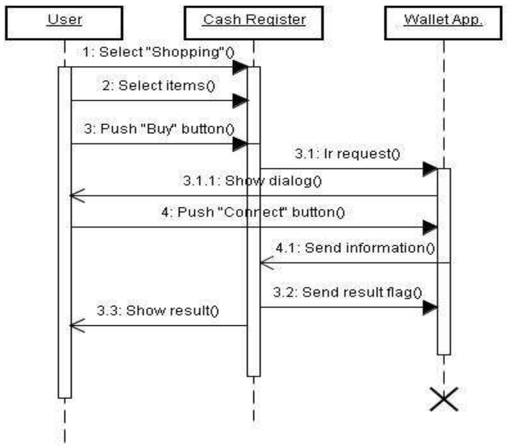


Sequence Diagram for Charging

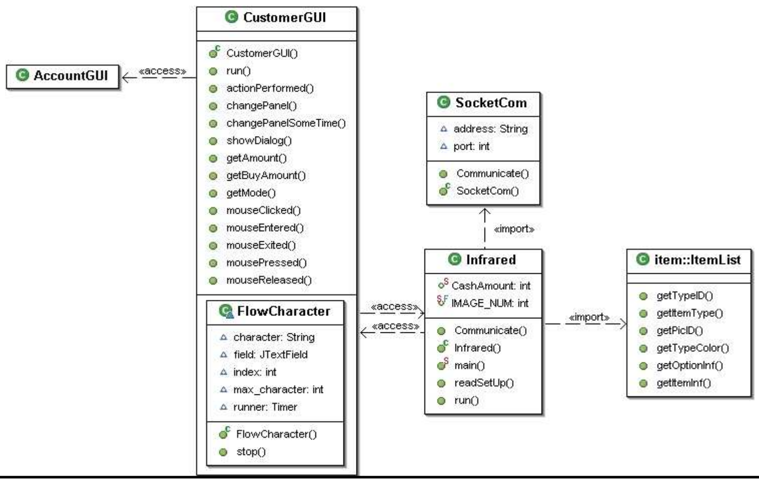


User
Wallet app.
CashRegister
BalanceServer
Select "Charge balance" menu
Show Input page
Enter the amount of money
Select "Charge" menu
Send "Charge" request, ID, and PIN
Send ID, PIN, and new balance
Check ID, PIN and Update balance
Case:Correct
Success
Success
Success
else
FAILURE
FAILURE
FAILURE

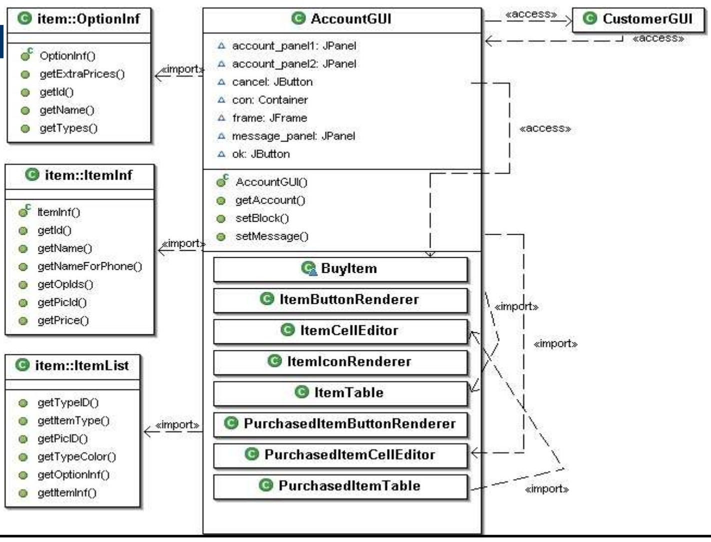
Sequence Diagram for Shopping



Class Diagram for Cash Register



Class Diagram for Cash Register *(cont)*



Outline

- ✓ Object and Motivation
- ✓ What is the UML and Real-time UML?
- ✓ Design for POS System with UML
- → ● **Demo**
- ● Conclusion

Demonstration environment

Laptop PC
- – Pentium® M processor 1.4GHz and 504MB RAM
- – Windows XP Professional

- ● Wallet application
  - i-appliTool for Doja-3.5(DoJa-3.5 Emulator)
  © NTT DoCoMo, Inc.

- ● Cash Register application and Balance server
  - Java™ 2 Runtime Environment,
  Standard Edition (build 1.4.2_3-B02)

Modification for Testing and Demo

- **Problem**

  Either applications locks the one IrDA port, so we cannot run both application on one PC.
- **Solution**

  Use TCP connection.

  Fortunately, OBEX supports not only IrDA, but also TCP connection, so no needs to change interface.

$\downarrow$

can adapt them with a few efforts

How to reduce the code size

- Rename classes, methods and fields shorter name.
- Reduce the number of classes, methods and fields.
  - Inline methods
  - Merge classes
- Merge some exception handlers together.

$\downarrow$ **However**

They also **reduce the Readability**

The application is **NOT** Object-Oriented

Outline

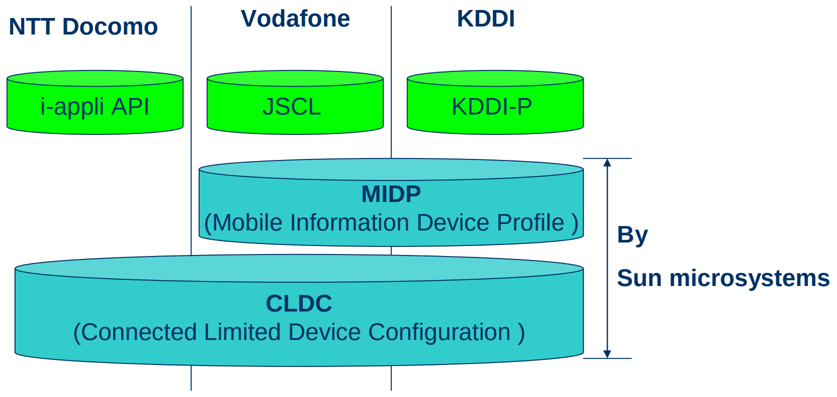
- ✓ Object and Motivation
- ✓ What is the UML and Real-time UML?
- ✓ Design for POS System with UML
- ✓ Demo
- → ● **Conclusion**

Conclusion

- UML
  - The features of Real-time UML will be combined into UML2.0, so Real-time UML might be discarded
  - Because of the ambiguous of diagrams, combination of UML and NL or SNL is necessary
  - For personal and small application, the advantage of UML will decrease
  - Appropriate SW process make UML more useful
- **Application**
  - satisfies the goal, but Needs much securer methodology. Ex.) Encryption
  - The implementation of phone application is **almost** same that of the application running on PC
  - Necessity of Error Handling

Thank you

Java API hierarchy of Japanese cellular phone


NTT Docomo
Vodafone
KDDI
i-appli API
JSCL
KDDI-P
MIDP
(Mobile Information Device Profile)
CLDC
(Connected Limited Device Configuration)
By
Sun microsystems