Object-Oriented Development and UML

2 – 4 pm Thursday 7/3/2008 @JD2211

1

Announcement

- Class will resume on July 22.
- Try to complete the lab assignments by July 21.

2

Agenda

- Review
- Object-Oriented Analysis
- Object-Oriented Design
- Object-Oriented Programming
- UML

3

Review

- Why object-oriented?
  - Essential difficulties
- Abstraction and encapsulation
- Object
- Class
  - Ready-made class
  - User-defined class
- Attribute
- Method
  - Constructor and `new` operator
  - Set method
  - Get method
- Package
- `import`
- Lab assignments

4

Object-Oriented Development

- Object-Oriented Analysis
- Object-Oriented Design
- Object-Oriented Programming

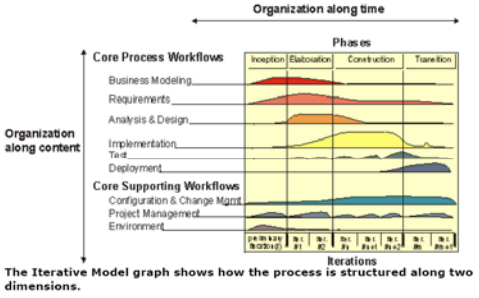
5

Object-Oriented Development

- The one of strengths of object-oriented development is that the transitions between the stages of OOA, OOD and OOP are *seamless*.
- Moving to the next stage involves refining the previous stage by adding detail to existing objects and devising new objects to provide additional functionality.

6

Unified Process



The Iterative Model graph shows how the process is structured along two dimensions.

7

UML is

- UML, the Unified Modeling Language, a *visual modeling language* that uses a combination of graphical and textual notation to express and document designs and processes
- UML is an open industry standard

8

UML is Not

- UML is not a computer programming language
- UML is not a methodology; that is, it does not dictate a particular design process

9

OMG

- The Object Management Group (OMG) is an international, open membership, not-for-profit computer industry consortium.
- http://www.omg.org/

10

UML Diagrams

- Static diagrams
  - Class, Object, Deployment, Component
- Dynamic diagrams
  - Use case, Sequence, Collaboration, Statechart, Activity

11

Object-Oriented Analysis

- Use case description and use case diagram
- OOA-level class diagram
- OOA-level sequence diagram

12

Use Case

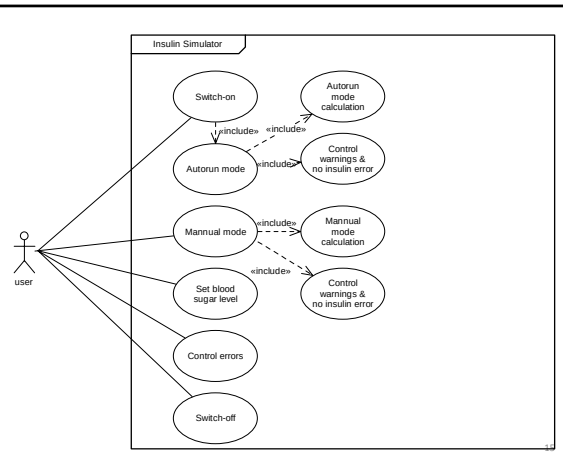
- Projects begin with use cases
  - A set of use cases describe all possible functions in the system
- Use case is a scenario-based technique for elaborating and verifying requirements
- Based on the use case diagram and their associated user case descriptions, the rest of UML diagrams can be developed.

13

Process of Development Use Case Descriptions and Diagram

- Find actors and use cases
- Prioritize use cases
- Develop each use case (starting with the priority ones)
- Structure the use case model

14



Use Case Description

UC1: Switch On

Characteristic Information

Goal In Context	Power-up and initialize the simulator
Pre-Condition	Simulator must be in the OFF mode
Success End Condition	Simulator is powered up and has been initialized
Primary Actor	Simulator user

16

Use Case Description

Main Success Scenario

Step	Actor/System	Action Description
1	User	Push the power on button
2	Simulator	Perform a self-test
3	Simulator	Set the simulator in AUTORUN mode
4	Simulator	Set the blood sugar-level to SAFE
5	Simulator	Run the clock

Scenario Extensions

Step	Condition	Action Description
2a	self-test fails	Set an alarm and notify the user to correct the problem
3a	Mode setting failure	Set an alarm and notify the user to correct the problem
4a	BSL initialization failure	Set an alarm and notify the user to correct the problem

17

Class Diagram

- UML class diagrams show the static structure of the system, that is, what classes there are and how they are related
- Building class diagrams starts identifying the classes and their relationships among them

18

Class Diagram (cont'd)

- The production of class diagrams is an ***iterative*** process.
- At the beginning, only a rudimentary "wire-frame" class diagram may be produced reflecting the requirements of the system being modeled.
- These diagrams can then be refined through an iterative process of review and further development.

19

Class Diagram Development Process

The process for developing a class diagram:

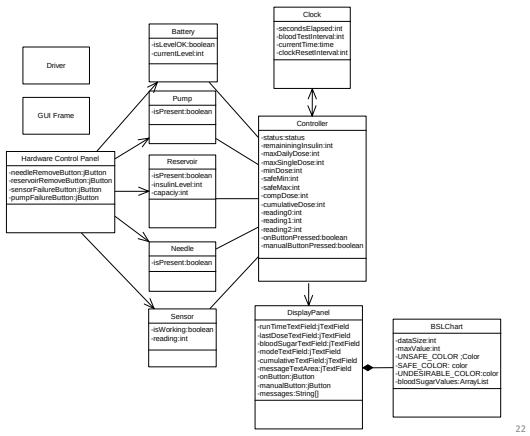
1. Identify and define the classes and their attributes
2. Identify and define the relationships and their attributes
3. Identify and define the methods

20

Class Diagram Development Process (cont'd)

4. Assign the methods to classes
5. Design the complex methods (using pseudo code or activity diagram)
6. Extend the class attributes with visibility, type, etc.
7. Extend the relationships with navigability, name, multiplicity, etc.
8. Validate the class diagram and go back to the previous steps, if necessary.

21



22

Interaction Diagrams

- Emphasize the communication between objects
  - Sequence diagram
  - Communication diagram
  - Timing diagram

23

Use Case Realization

- During the analysis workflows, more details are added to each use case, including a descriptions of the objects involved in carrying out the use case.
- The process of extending and refining use cases is called use-case realization.

24

Use Case Realization

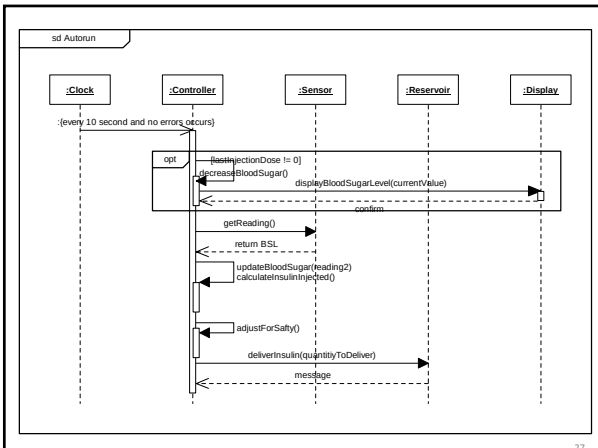
- An interaction diagram (sequence diagram or communication diagram) depicts the realization of a specific scenario of the use case.
- The primary contribution of use case realization is to help developers to make a complete class diagram, in particularly by identifying objects that collaborate and messages (that become methods or functions for classes) and assigning them to specific classes.

25

Process for Developing Sequence Diagrams

1. Identify and decide the objects that collaborate
2. Identify and decide messages between objects
   - Messages
     - Methods calls (synchronous)
     - Return signal
     - Sending a signal (asynchronous)
     - Creating an object
     - Destroying an object

26



27

Class Relationships

- Association
- Aggregation
- Composition
- Generalization (Inheritance)

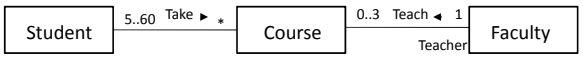
28

Association

- Association is a general binary relationship that describes an activity between two classes. For example, a `student` taking a course is an association between the Student class and the `Course` class, and a faculty member teaching a course is association between the `Faculty` class and the `Course` class

29

Association -- UML



30

```
Public class Student {
       private Course [] courseList;

       public void addCourse (Course s)

}

 Public class Course {
        private Student [] classList;
        private Faculty faculty;

        public void addStudent (Student student)
        public void setFaculty (Faculty faculty)

 }
 Public class Faculty {
        private Course [] courseList;

        public void addCourse (Course s)

 }
```

31

Aggregation

- Aggregation is a special form of association that represents an ownership relationship between two objects.
- Aggregation models *has-a* relationship.

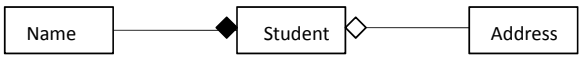
32

Composition

- If an object is *exclusively* owned by an aggregating object, the relationship between the object and its aggregating object is referred to as composition

33

Aggregation and Composition -- UML Notation



34

Aggregation and Composition – Java Code

```
Public class Student {
       private Name name;
       private Address address;

       public Student ()
       {
              name = new Name (.....);
              address = new Address (.....);
       }
       ....
}

Public class Address {
       ……
}
```

35

The `this` Reference

- The word this is reserved in Java
- It allows an object to refer to itself

36

Lab Assignment

- http://www.csun.edu/~twang/Java/Lab/7-3-Lab.pdf

37