ĐẠI HỌC ĐÀ NẴNG
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG VIỆT - HÀN
Vietnam - Korea University of Information and Communication Technology

SYSTEMS ANALYSIS AND DESIGN

Lecturers: Nguyen Thanh Binh – Nguyen Quang Vu – Le Viet Truong – Le Thi Bich Tra – Vo Van Luong – Nguyen Thi Hanh

Faculty of Computer Science

Vietnam - Korea University of Information and Communication Technology (VKU)

http://vku.udn.vn/

Case study

Case study

- Problem
  - A very simple problem to show the use of UML in analysis and design
  - It is taken from the “Applying UML and Patterns” book of Claig Larman
  - A dice game
    - The player rolls 10 times 2 dice. If the total of two dice is 7, he gains 10 points. At the end of the game, the score is saved to the scoreboard

Main Activities of Software Development

Requirements Gathering

Define requirement specification

Analysis

Define the conceptual model

Design

Design the solution / software plan

Implementation

Code the system based on the design

Integration and Test

Prove that the system meets the requirements

Deployment

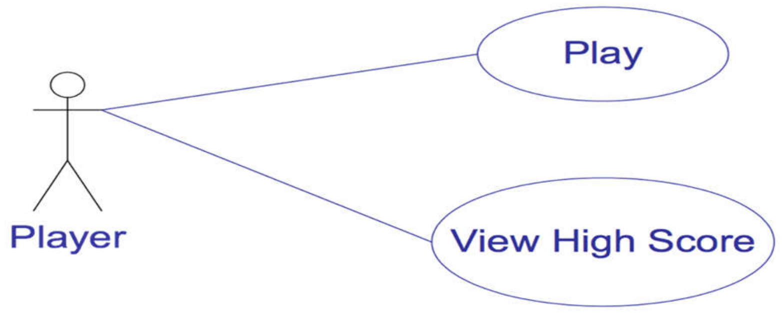
Installation and training

Maintenance

Post-install review
Support docs
Active support

Case study

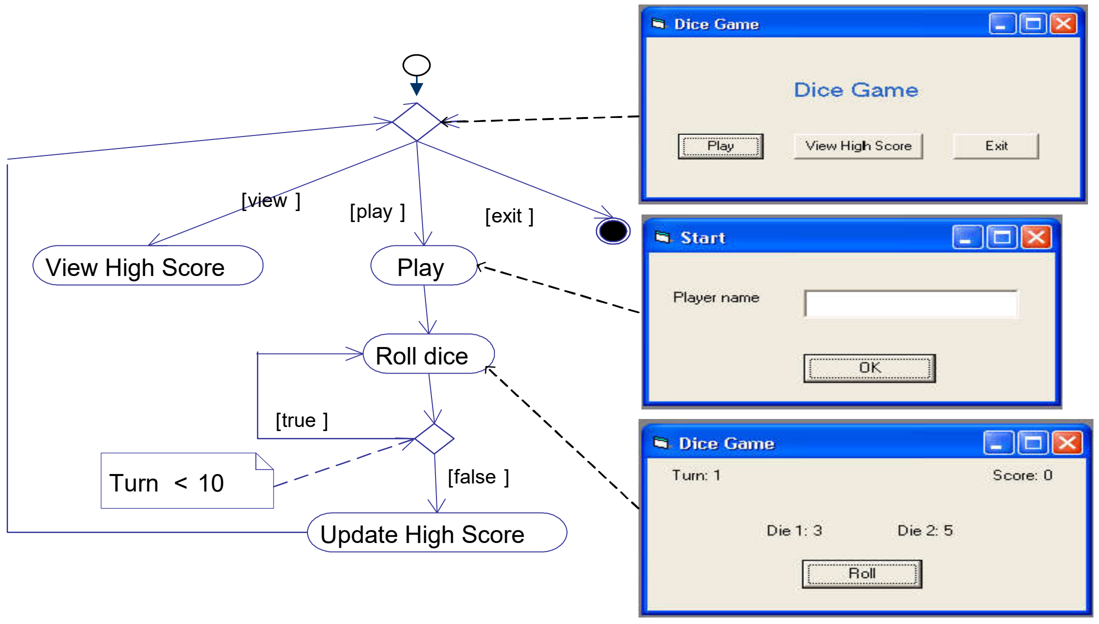
- Requirement analysis
  - Use-case diagram



  - Use-case: Play
    - Description: The player rolls 2 dice 10 times. If each time the total is 7, he receives 10 points.
  - Use-case: View High Score
    - Description: They player consults the scores

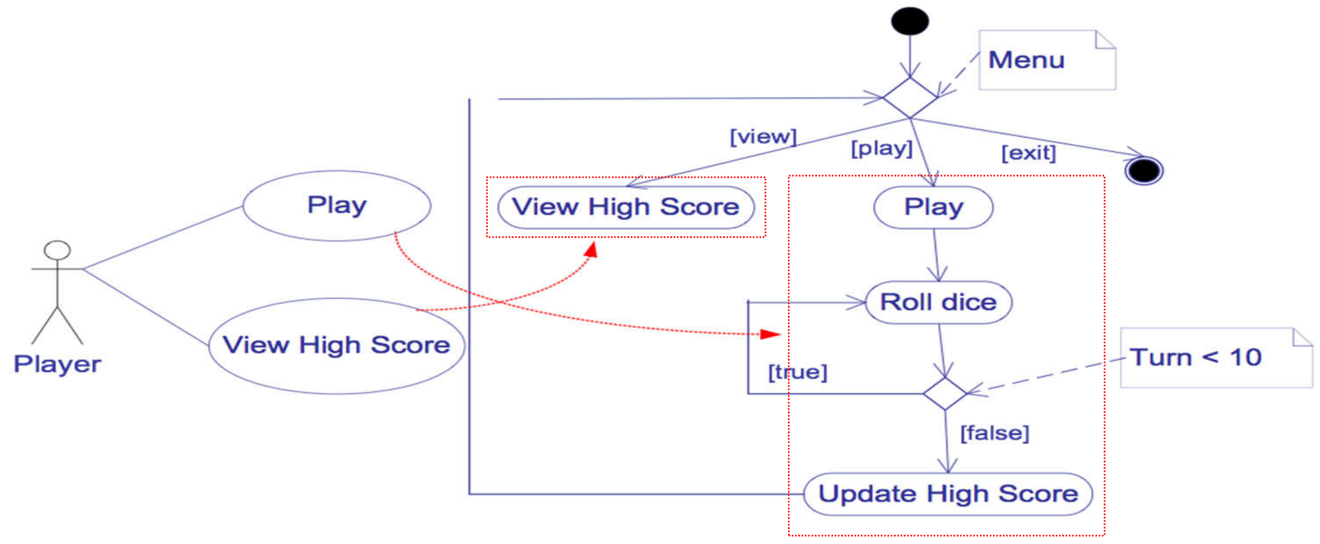
Case study

- Requirement analysis
  - Activity diagram
    - Some activities are linked to the graphical user interface



Use-case

- Requirement analysis
  - Activity diagram
    - The relationship between the use-case diagram and activity diagram



Main Activities of Software Development

Requirements Gathering

Define requirement specification

Analysis

Define the conceptual model

Design

Design the solution / software plan

Implementation

Code the system based on the design

Integration and Test

Prove that the system meets the requirements

Deployment

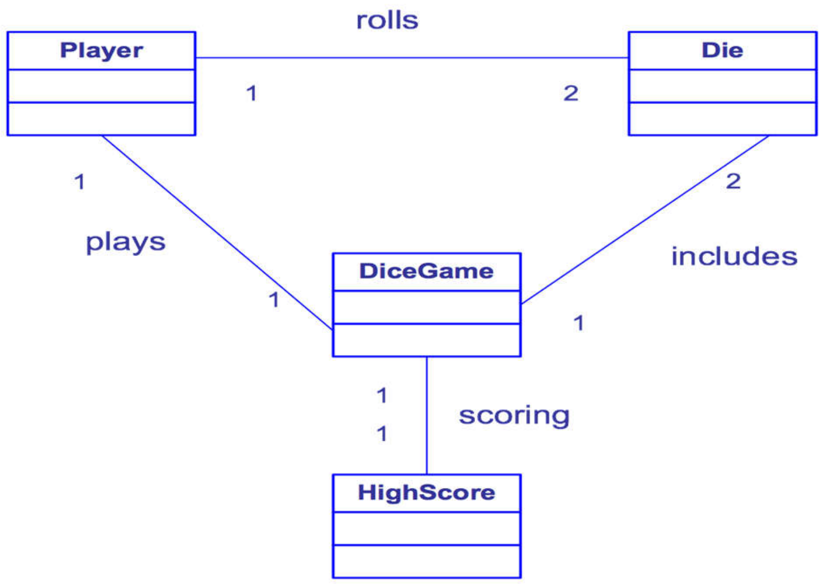
Installation and training

Maintenance

Post-install review
Support docs
Active support

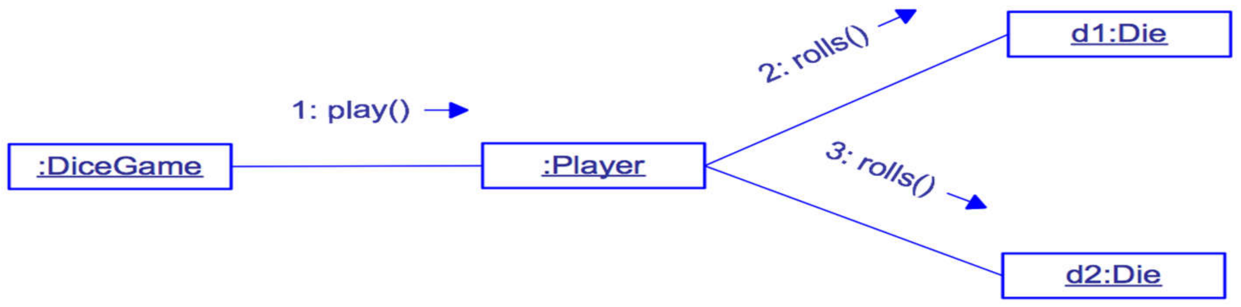
Case study

- Modeling of conceptual class diagram



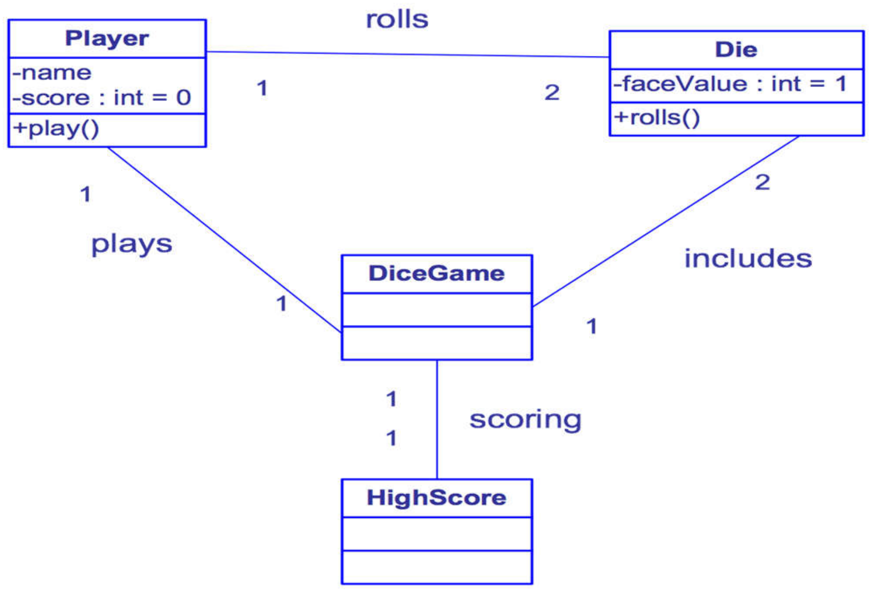
Case study

- A first collaboration diagram



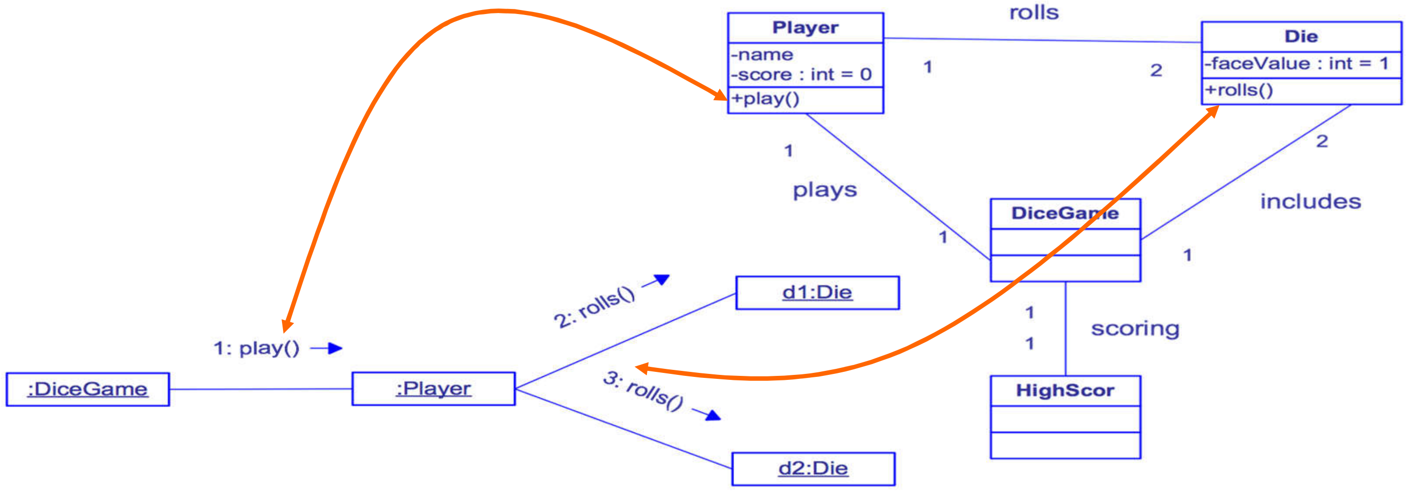
Case study

- A first class diagram



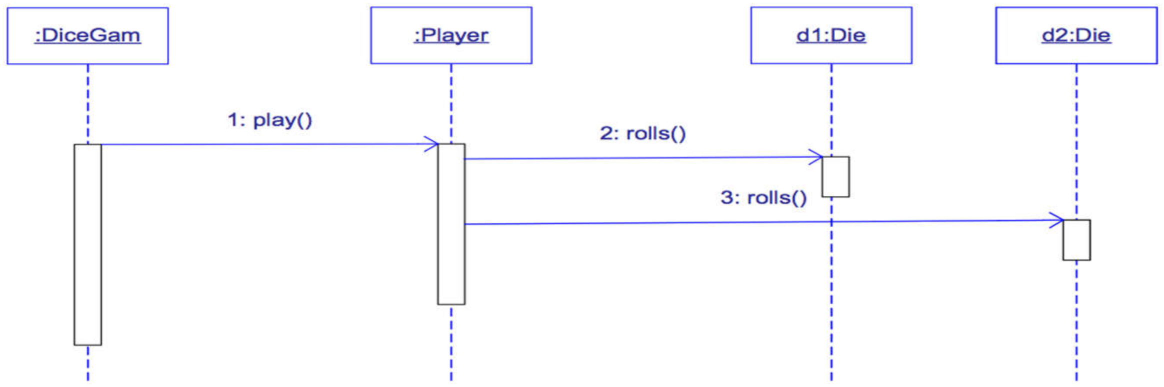
Case study

- Collaboration diagram and class diagram



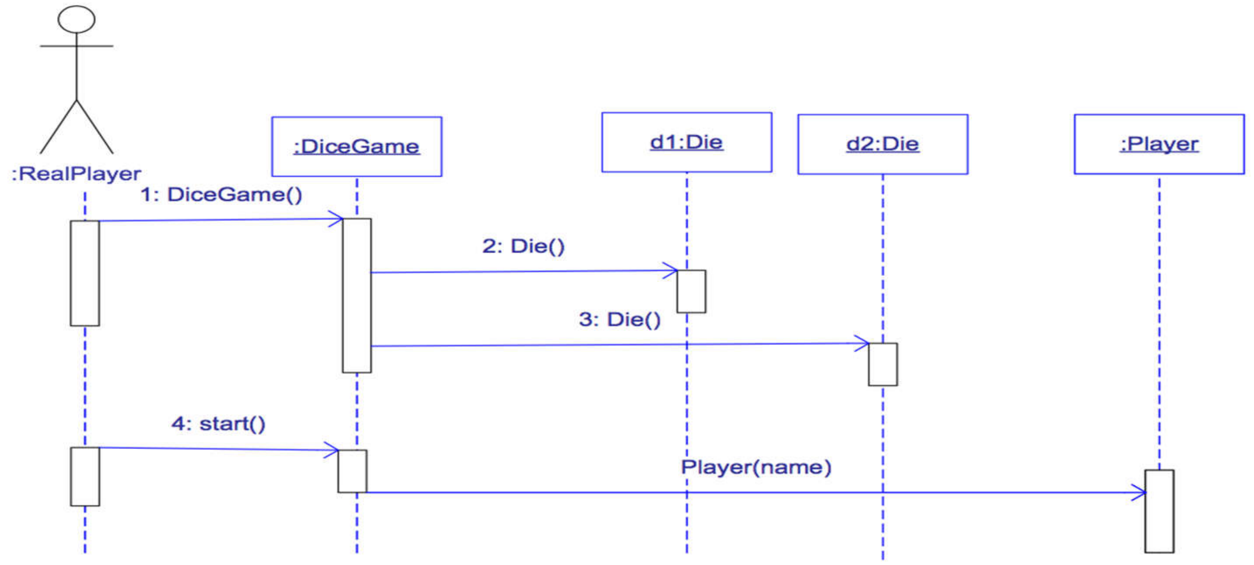
Case study

- Sequence diagram



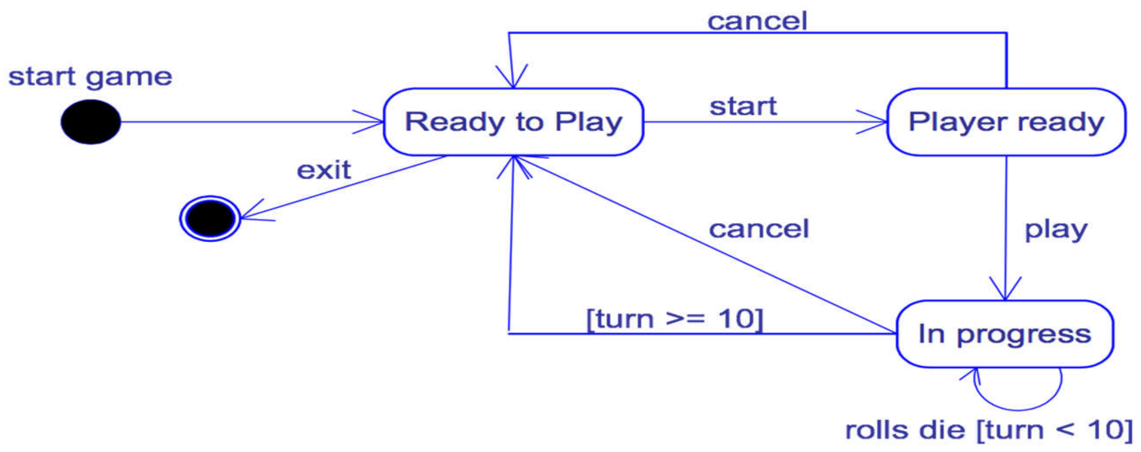
Case study

- The creation of objects at the beginning of the game (DiceGame) for a player



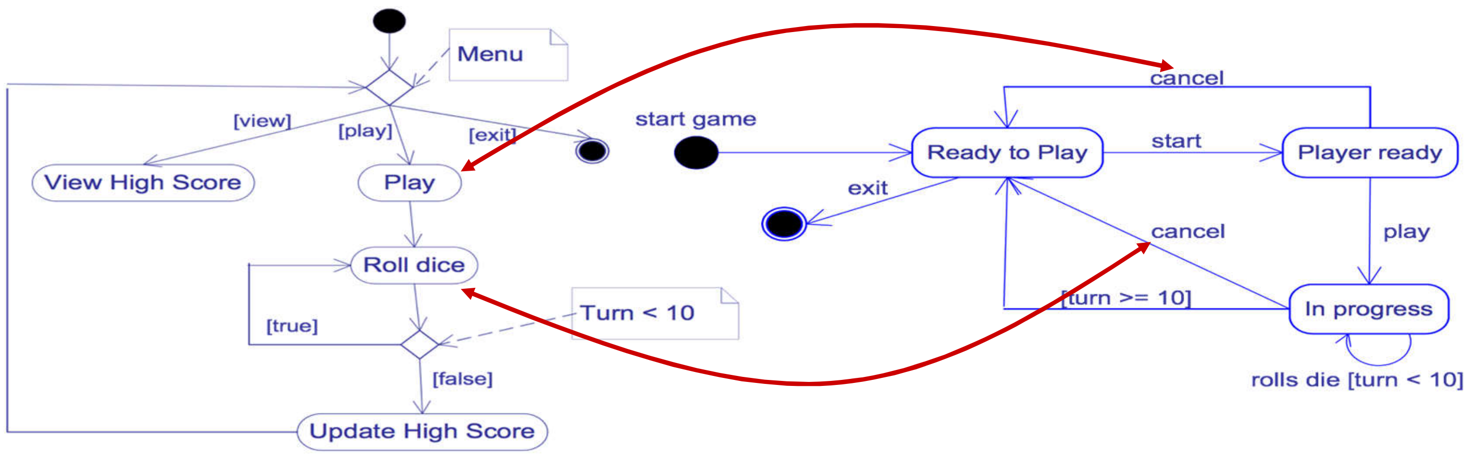
Case study

- State diagram: modelling the states of the DiceGame



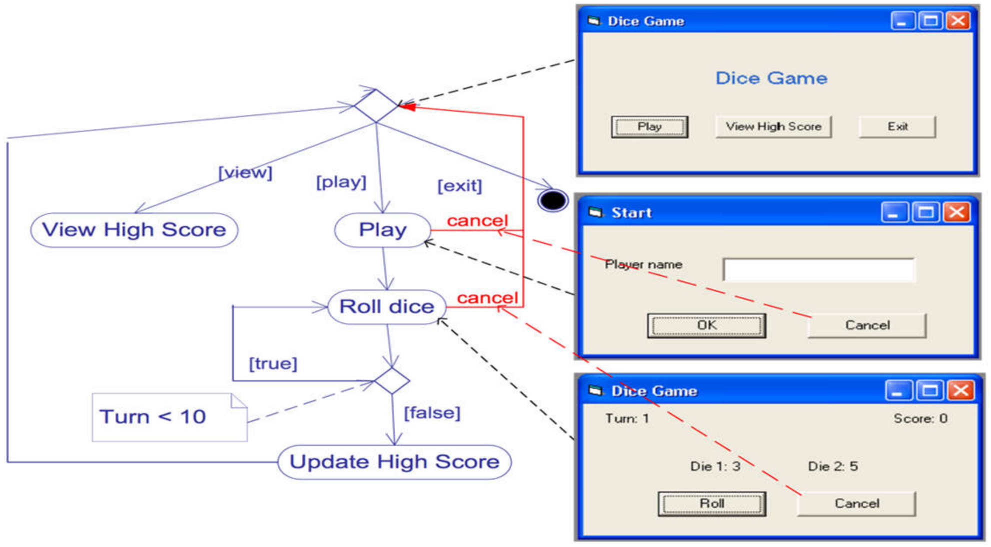
Case study

- Detection of inconsistency between the activity diagram and the state diagram



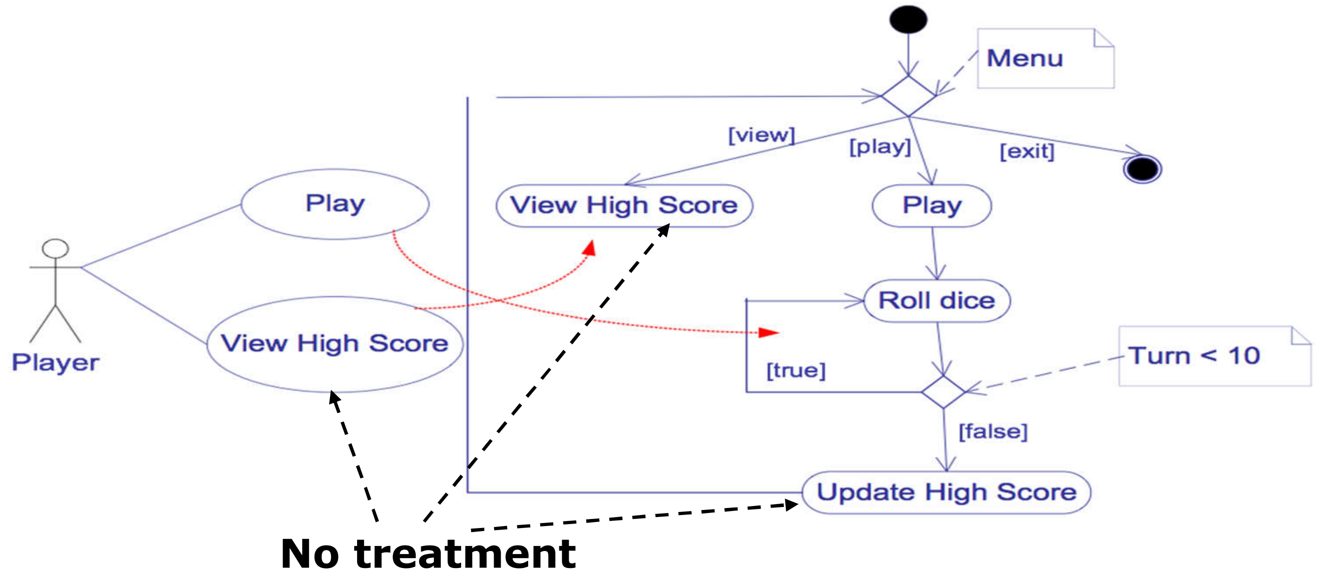
Case study

- Modification of the activity diagram as well as the envisaged graphical user interface



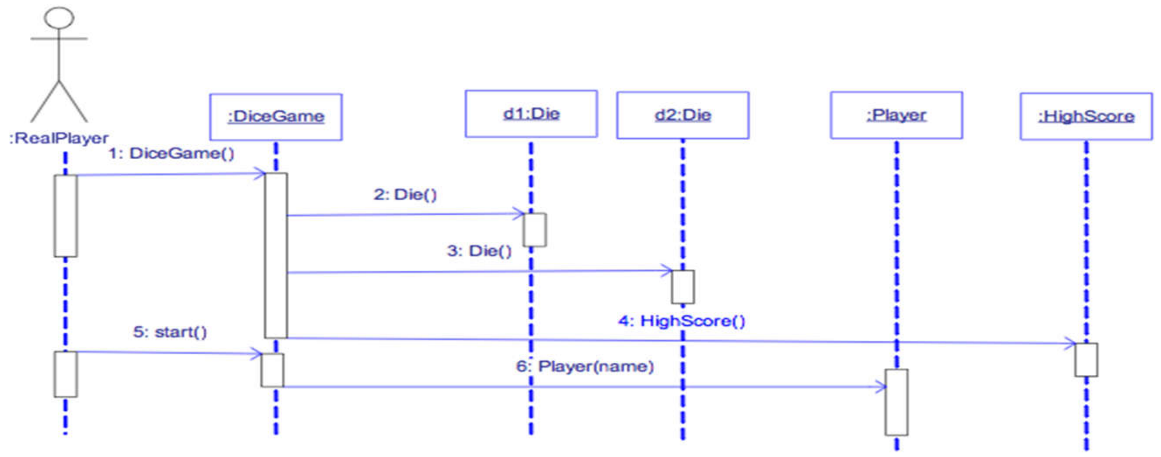
Case study

- The treatment of the scoreboard must be taken into account: the update and the creation



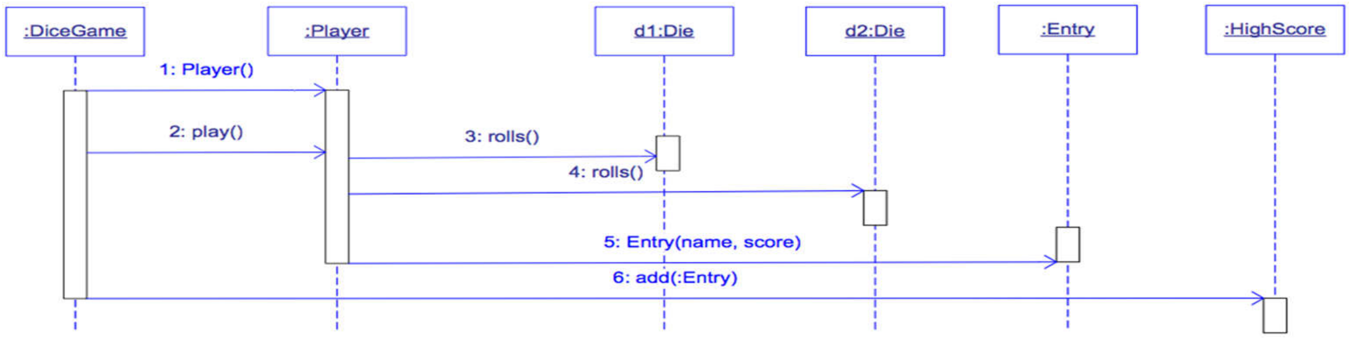
Case study

- Sequence diagram: manage high score, create new player



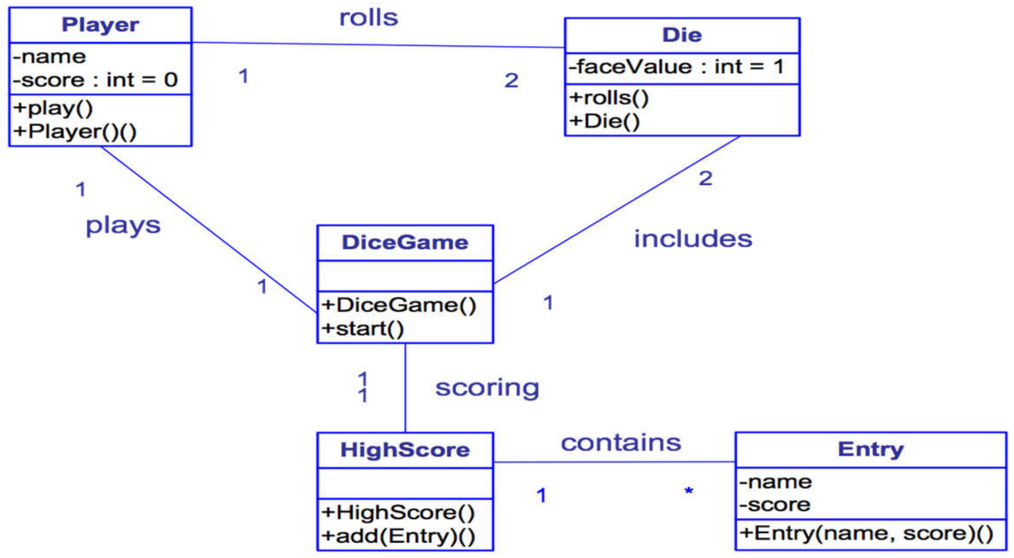
Case study

- Sequence diagram: add high score to score board



Case study

- Class diagram



Main Activities of Software Development

Requirements Gathering

Define requirement specification

Analysis

Define the conceptual model

Design

Design the solution / software plan

Implementation

Code the system based on the design

Integration and Test

Prove that the system meets the requirements

Deployment

Installation and training

Maintenance

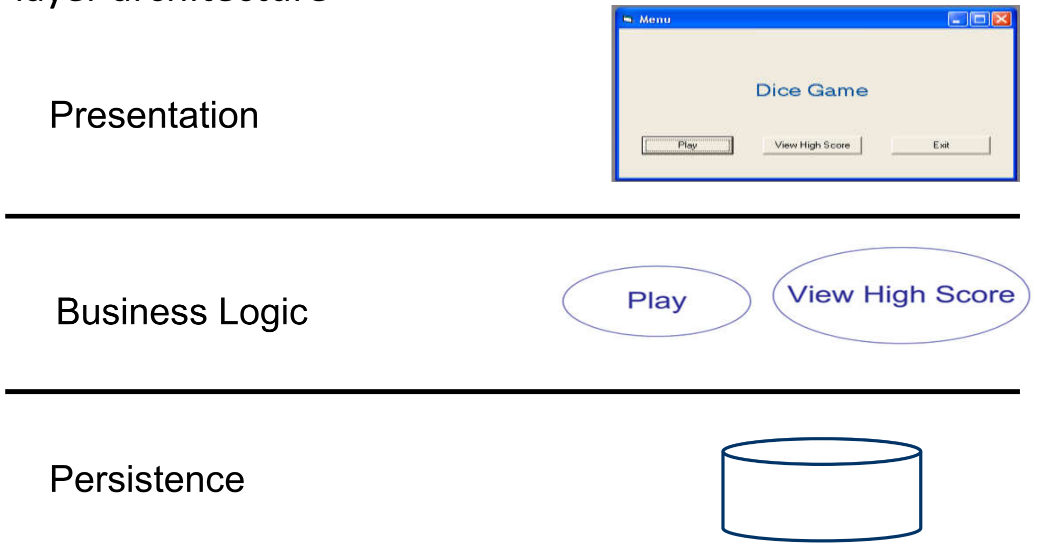
Post-install review
Support docs
Active support

Case study

- Design
  - Take into account the implementation
    - Manage the graphical user interface part
    - Manage the persistence of scoreboard
  - Define the logical architecture
  - Define the physical architecture
  - Introduce the technical class permitting to implement the architecture

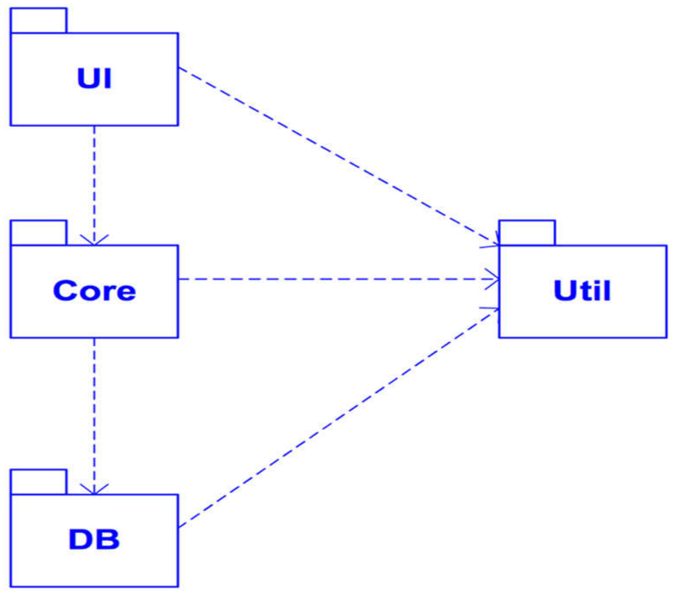
Case study

- General architecture
  - Classical three layer architecture



Case study

- A package diagram corresponds to the architecture



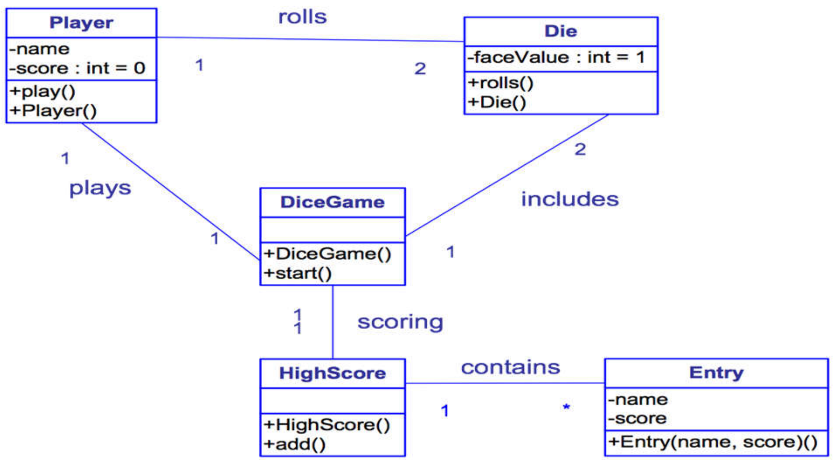
UI	: presentation layer
Core	: Business logic layer
DB	: Persistence layer
Util	: utility services/classes/functionalities

Case study

- Use design patterns to improve the classes of “Core” package

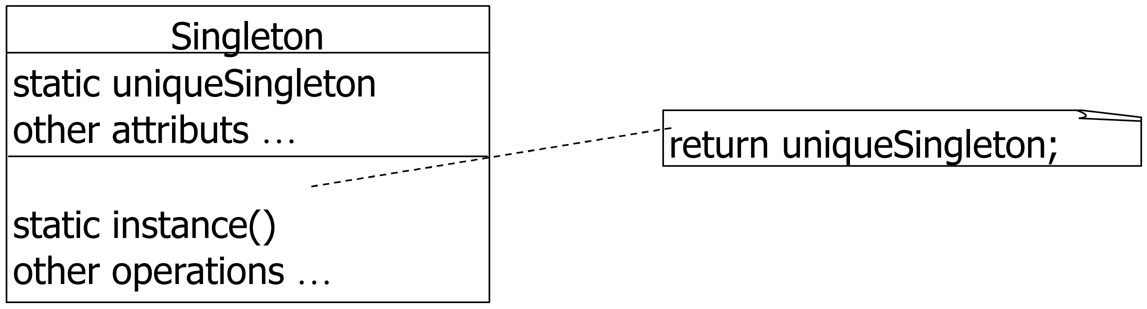
Class DiceGame has only one object
Class HighScore has only one object

Design pattern : Singleton



Case study

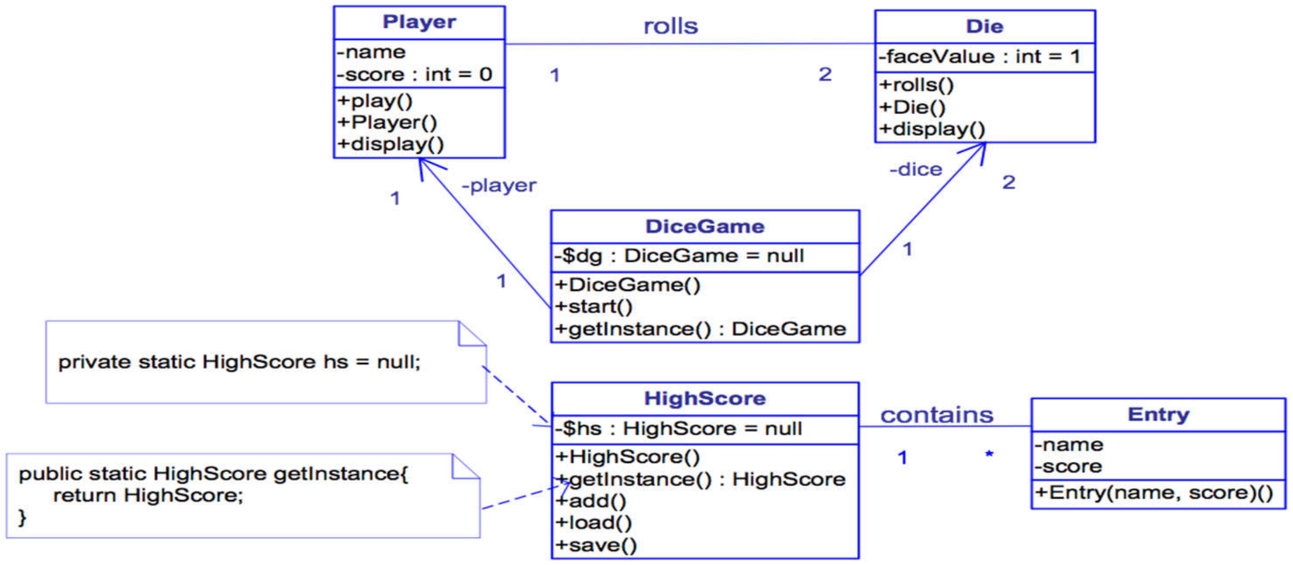
- **Singleton** design pattern



- Application to **DiceGame** and **HighScore**.

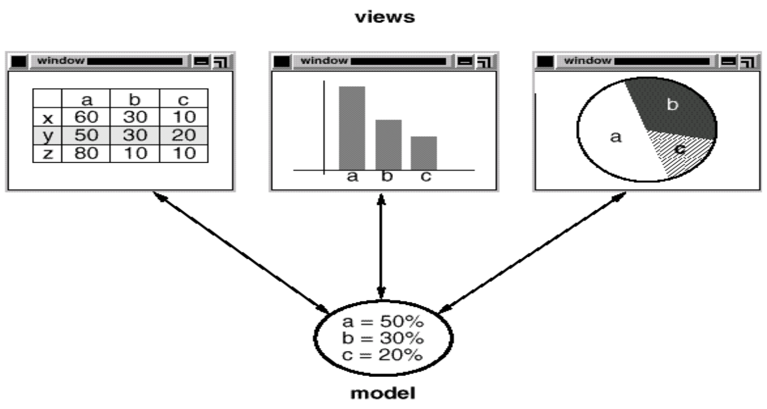
Case study

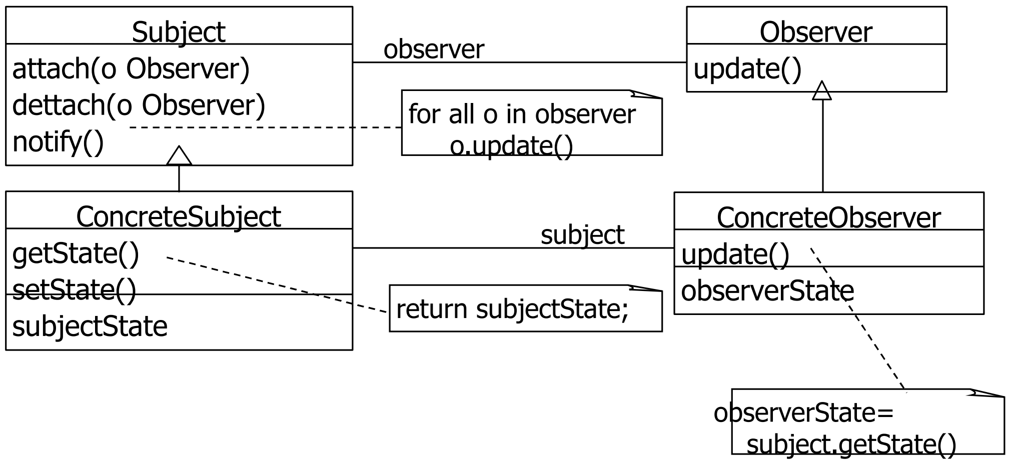
- Modified class diagram



Case study

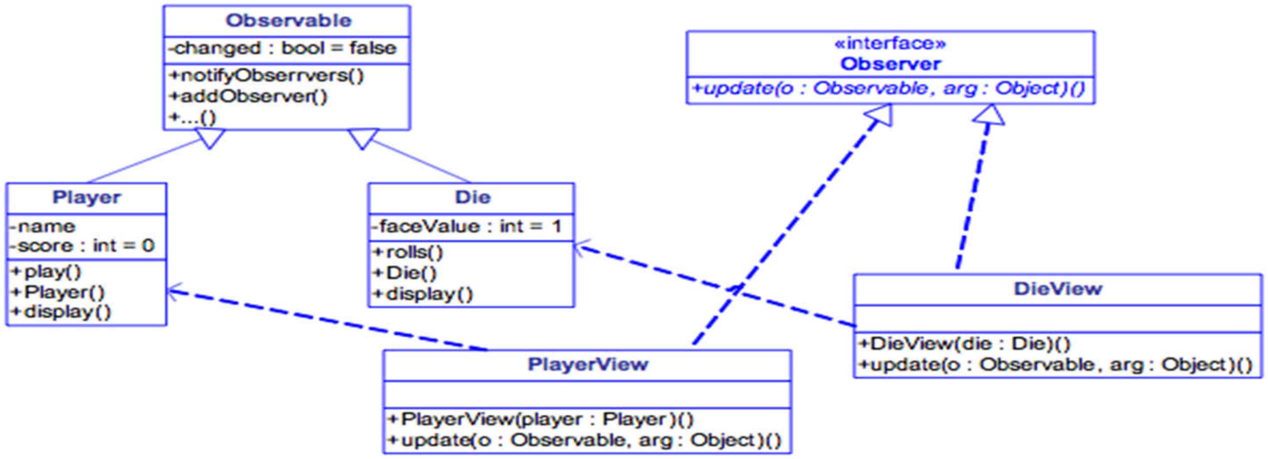
- **Observer** design pattern





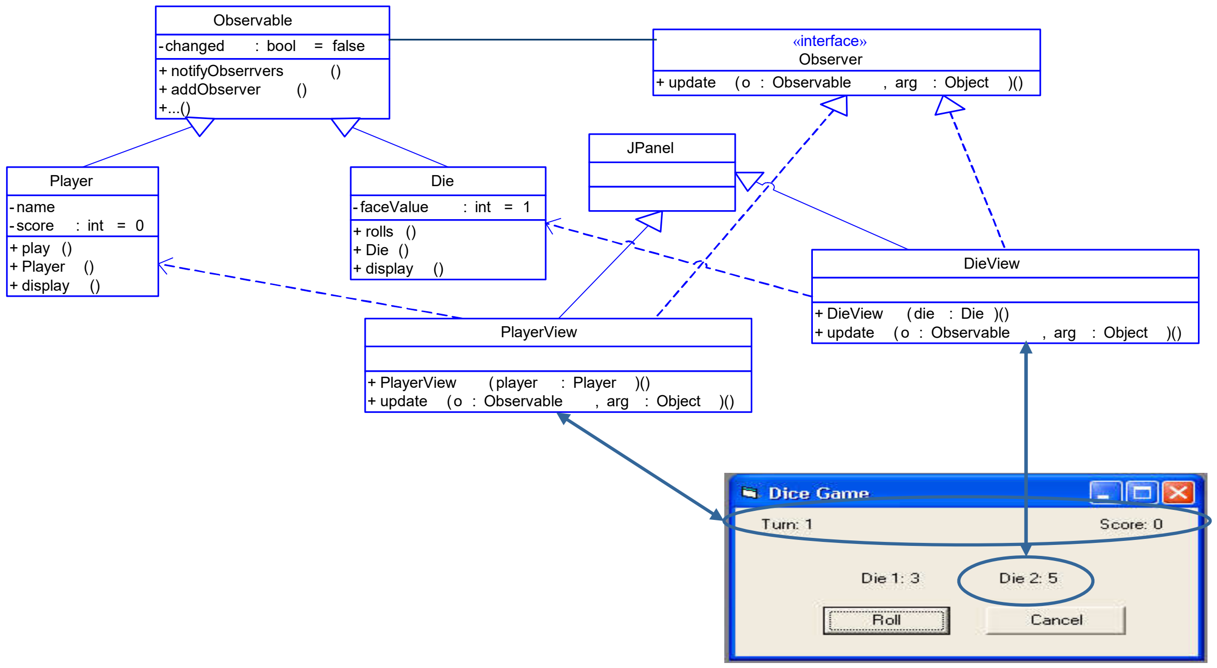
Case study

- Application of **Observer** design pattern to improve the class diagram
  - Decouple the graphical views and objects for the dice and players
  - Application of **Observer** pattern
    - **Die** and **Player** classes are **ConcreteSubject** class
    - Introduce **DieView** et **PlayerView** as **ConcreteObserver** classes



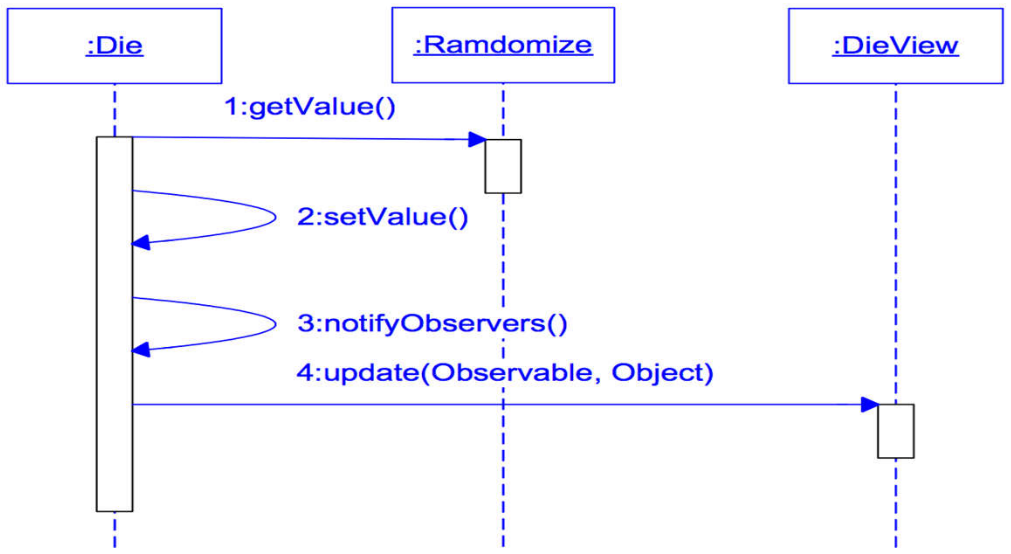
Case study

- User view are instances of *javax.swing.JPanel.java*



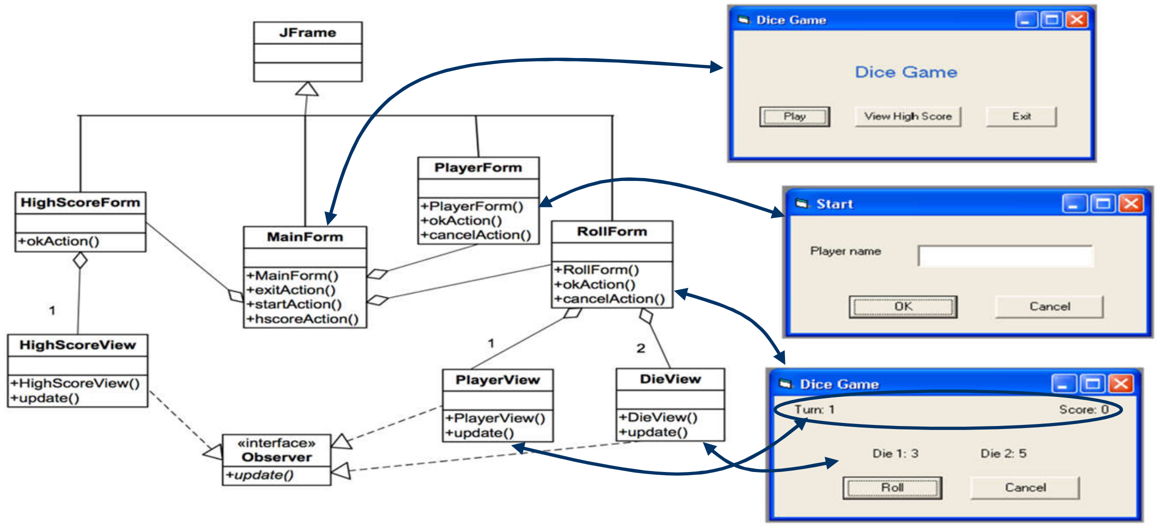
Case study

- Sequence diagram describes the interactions between **Die** object the its view



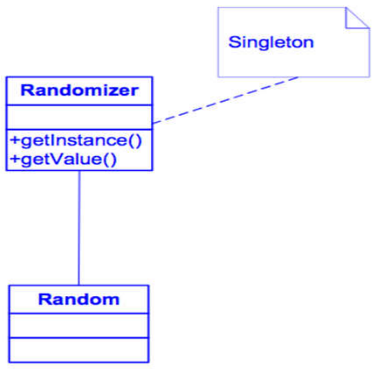
Case study

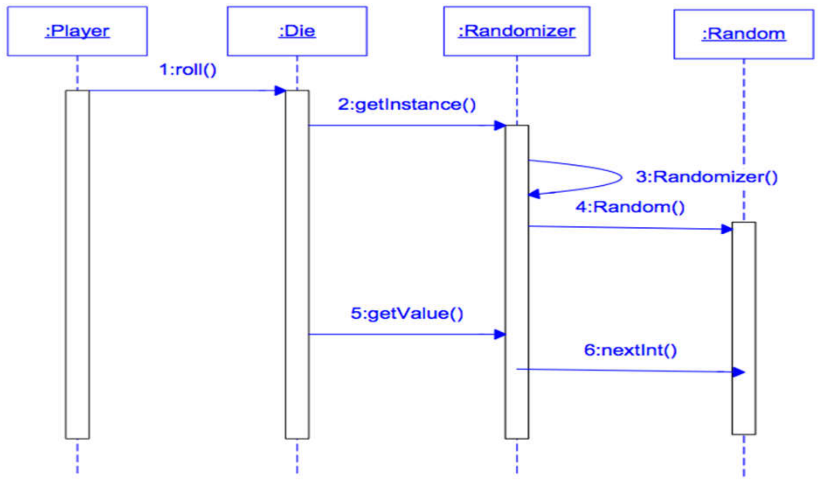
- The design of "UI" package



Case study

- The design of “Util” package

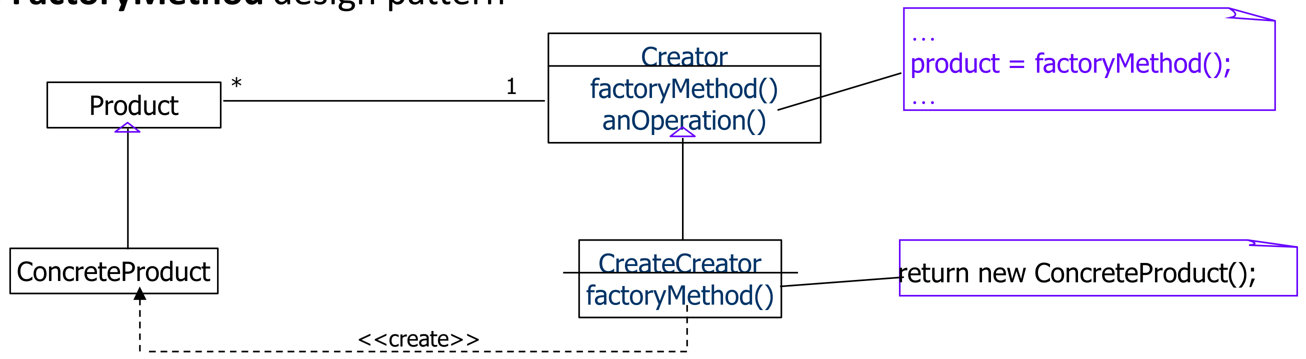




Case study

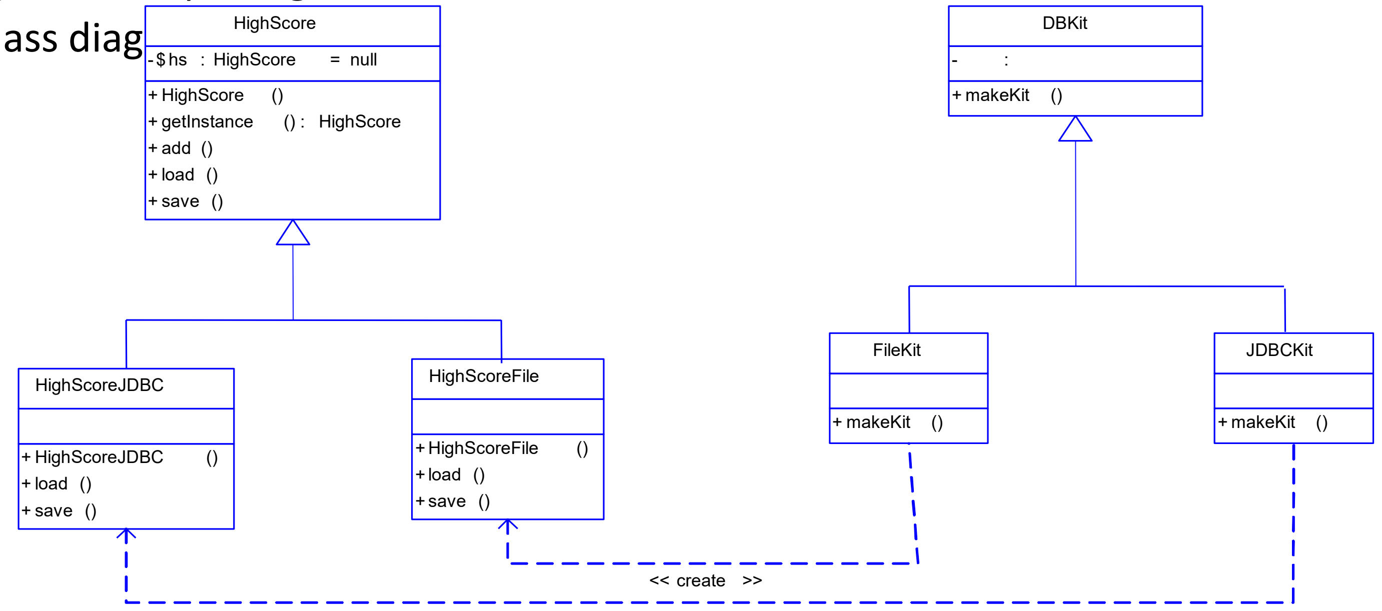
- The design of “DB” package
  - How to ensure the independence between “Core” and “DB” package
    - In order to be able to use several persistence types
      - File (serialisation)
      - Relation Database Management System (via JDBC)

    - Use **FactoryMethod** design pattern



Case study

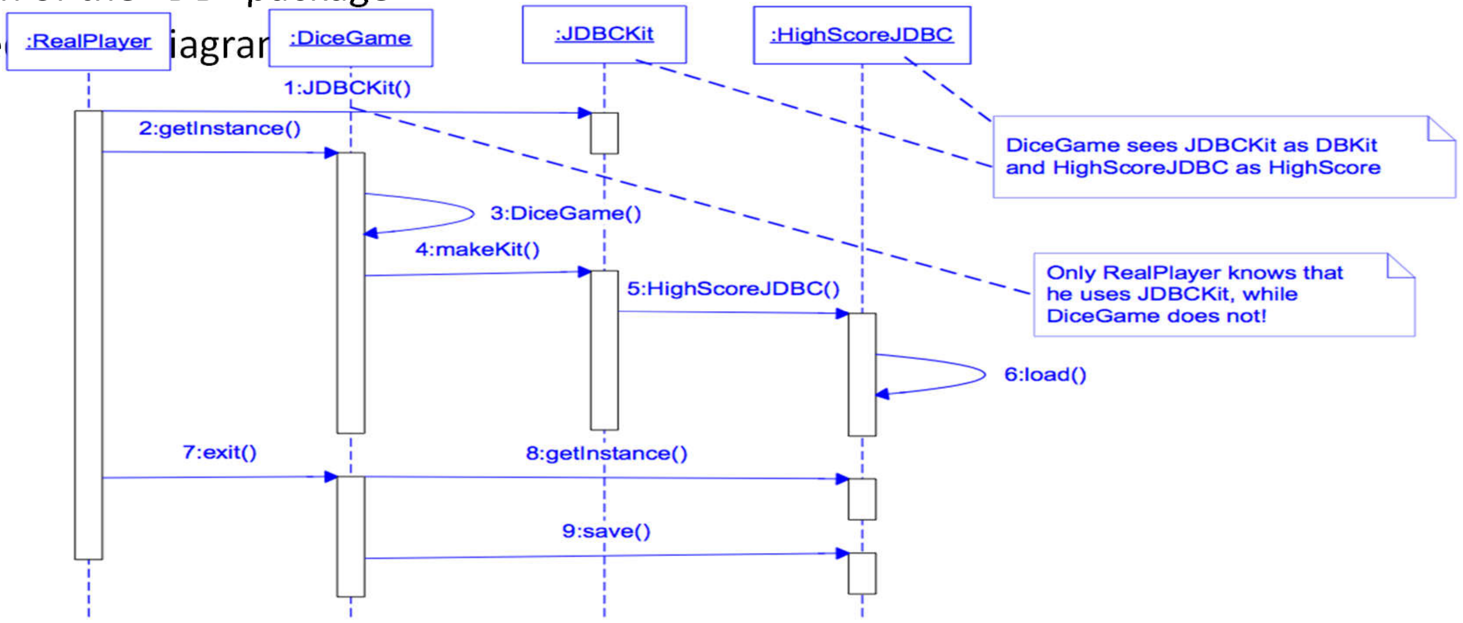
- The design of “DB” package
  - Class diag



Note : HighScore class is a Singleton

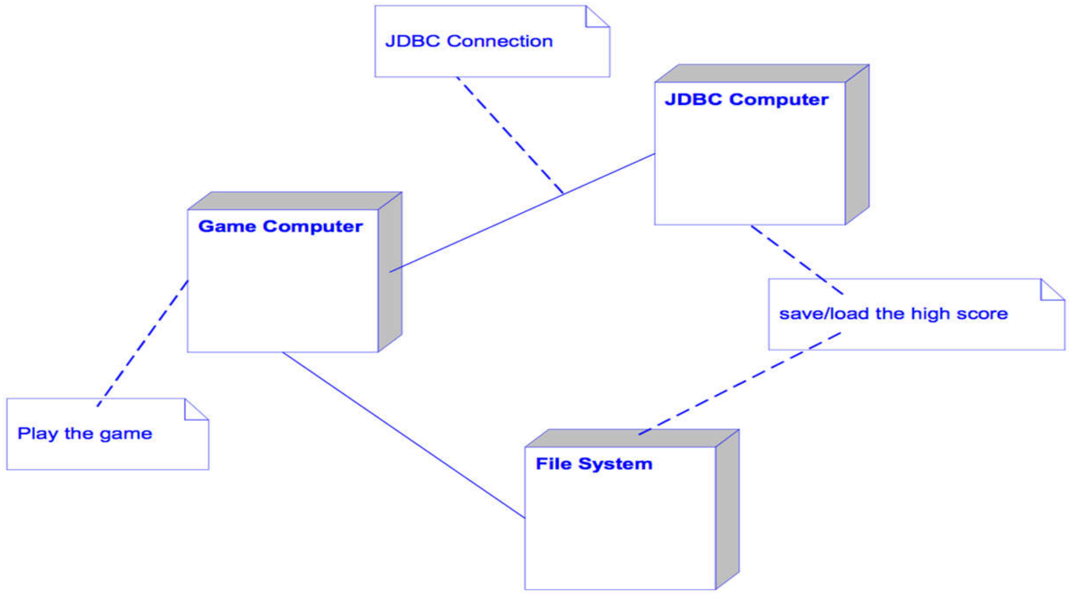
Case study

- The design of the “DB” package
  - Se iagrar



Case study

- Deployment diagram



Main Activities of Software Development

Requirements Gathering

Define requirement specification

Analysis

Define the conceptual model

Design

Design the solution / software plan

Implementation

Code the system based on the design

Integration and Test

Prove that the system meets the requirements

Deployment

Installation and training

Maintenance

Post-install review
Support docs
Active support

Case study

- Complete the interaction diagrams
- Generate the code

Conclusions

Conclusions

- Distinction between functional approach and object-oriented approach
- Master the basic object-oriented concepts

- UML: a modelling language
  - Need a development process
  - Different views
  - Different models
  - Use of the models in different development activities

- Master the main diagrams
  - Use-case diagram
  - Class diagram
  - Interaction diagram

Conclusions

- The UML concepts can be extended
    - The extensions
- Transformation of models to code
    - Models independent of programming language
- The automatic code generation is only a supplement
    - The models guide the coding process
- Master design principles
    - GRAPS principles/patterns
    - Some design patterns

Chapter 10.
Case Study