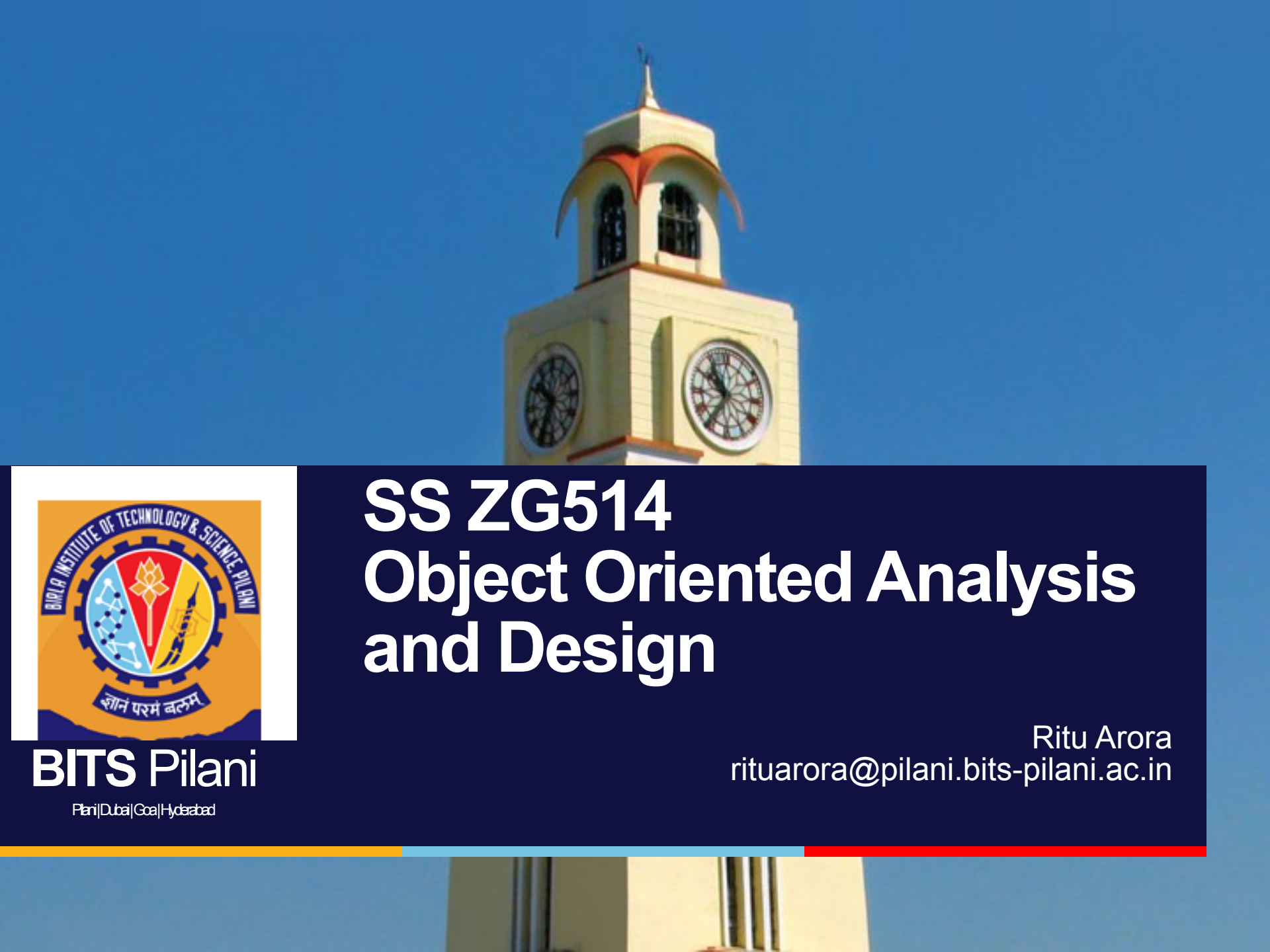


SS ZG514
Object Oriented Analysis and Design

Ritu Arora
rituarora@pilani.bits-pilani.ac.in

BITS Pilani
Pilani|Dubai|Goa|Hyderabad

innovate achieve lead

BITS Pilani

Pilani|Dubai|Goa|Hyderabad

GRASP Patterns

GRASP Patterns

- GRASP stands for General Responsibility Assignment Software Patterns
- Based on Responsibility-Driven Design (RDD) approach
- Software objects have responsibility- which is an abstraction of what they do
- Responsibilities are of two types: doing and knowing
- Doing responsibilities of an object include:
  - Doing something itself, such as creating an object or doing a calculation
  - Initiating action in other objects
  - Controlling and coordinating activities in other objects

GRASP Patterns

innovate achieve lead

- Knowing responsibilities of an object include:
  - Knowing about private encapsulated data
  - Knowing about related objects
  - Knowing about things it can derive or calculate

- A responsibility is not the same thing as method- it's an abstraction- but methods help objects fulfill responsibilities

- Nine GRASP Patterns
  - Creator - Information Expert
  - Low Coupling - Controller
  - High Cohesion - Polymorphism
  - Pure Fabrication - Indirection
  - Protected Variations

Creator Pattern

Problem: Who should be responsible for creating a new instance of some class?

Solution: Assign class B the responsibility to create an instance of class A if one of these is true (the more the better):

- B “contains” or compositely aggregates A
- B records A
- B closely uses A
- B has the initializing data for A that will be passed to A when it is created. Thus B is also an Expert in creating A

Note: If more than one option applies, usually prefer a class which aggregates or contains class A.

Snakes And Ladders

Snakes And Ladders Game Project: Study this case study and answer the questions that follow:

The aim of this project is to develop an online application where the players can create and play a customized game of Snakes and Ladders. The GameOwner is required to choose the number of snakes and ladders he wants to be placed on the board, along with the starting and ending point of each snake and ladder (not more than 10 snakes and 10 ladders can be placed on the board). After customization, the GameOwner is displayed a board drawn as a 10X10 matrix (with numbers) and snakes and ladders drawn over it as directed arrows (as shown in Figure 1).

Next, the GameOwner can invite players to play the game. At any instance of time, not more than 4 players (including GameOwner) and not less than 2 players (including GameOwner) can play the game. Individual player coins are drawn as colored circles and placed on the board. There is no differentiation between the coin of the GameOwner and that of the other players. Finally, each player as well as the GameOwner gets turns to roll the dice (a random number between 1 and 6) and play the game.

Snakes And Ladders

Even\Odd mode:

The game can also be customized to be played in an even or odd mode. A player can choose an even or odd mode, while joining the game. In even mode, the player moves his coin only if he gets an even number on his dice. Similarly, in odd mode, the player moves his coin on getting an odd number. GameOwner also has the option to choose between even\odd modes.

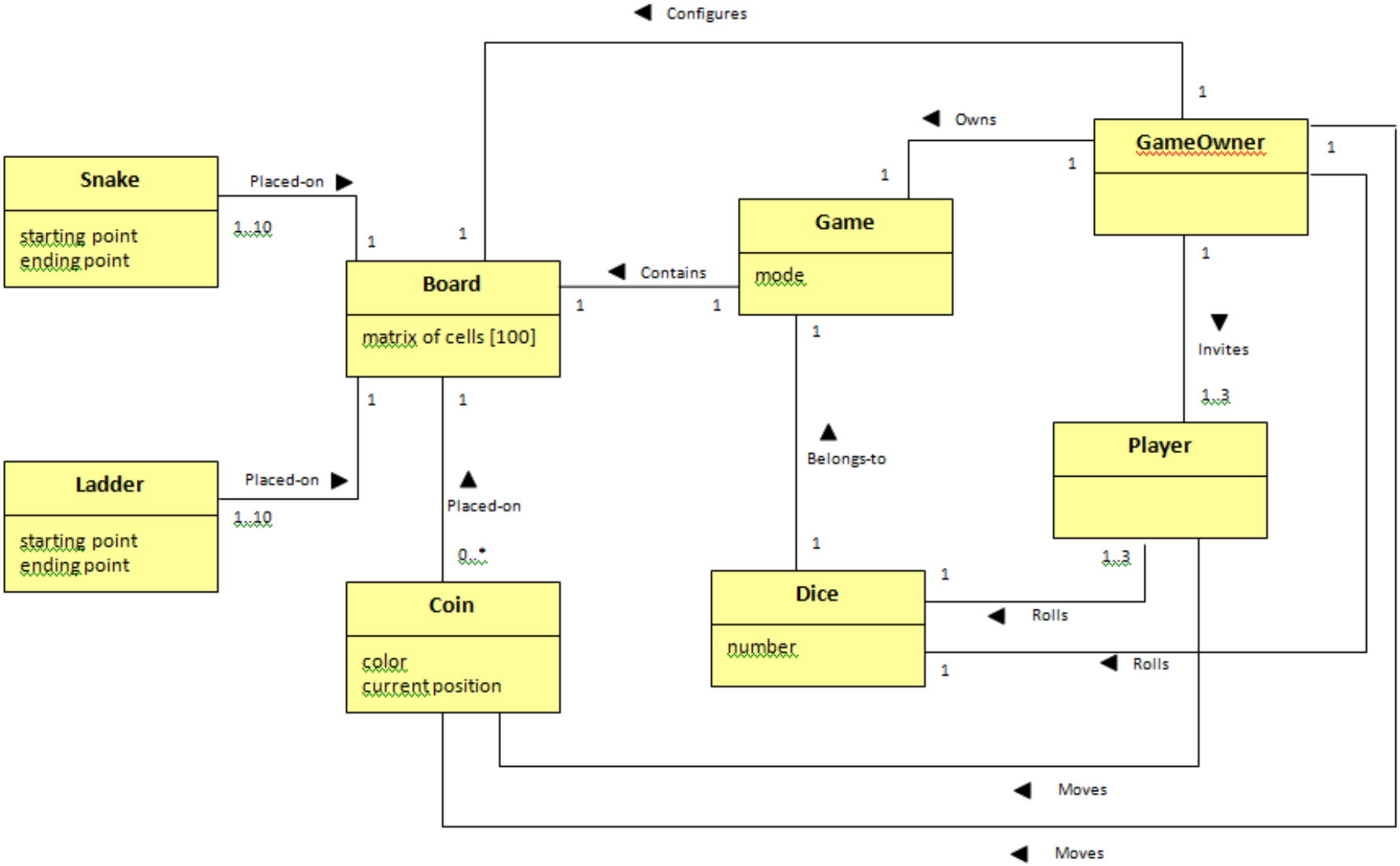
Multiple coins mode:

The game can also be customized to be played in multiple coins mode. For a particular player, 2 to 4 coins can be placed on the board at the same time. However, the number of coins for all the players should be same. A player wins only after all his coins pass the 100^{th} cell of the board. All the coins for a particular player are of the same colour.

Hybrid mode:

The game can also be played in hybrid mode, which is a combination of the even\odd mode and the multiple coins mode.

Snakes And Ladders



Creator Pattern: Example

In the SnakesAndLadders example,

- Who should be responsible for creating the objects of the Snake class?

- Who should be responsible for creating the objects of the Dice class?

Information Expert Pattern

Problem: What is the general principle of assigning responsibility to objects?

Solution: Assign responsibility to the information expert- the class that has the information necessary to fulfill the responsibility.

Information Expert : Example

In the SnakesAndLadders example,

- Who should be responsible for finding whether the tail or mouth of a Snake has been reached, after a move?
- Who should be responsible for knowing whether a player operates in even or odd mode?
- Who is responsible for knowing whether a coin has crossed 100^{th} square or not?

Low Coupling Pattern

Problem: How to support low dependency, low change impact, and increased reuse?

Coupling is a measure of how strongly one element is connected to, has knowledge of, or relies on other elements.

An element (class/subsystem/system) with low coupling is not dependent on too many other elements.

Low Coupling Pattern

Elements with high coupling rely on many other classes, and suffer from the following problems:

- Forced local changes because of changes in related classes
- Harder to understand in isolation
- Harder to reuse because its use requires the additional presence of the classes on which it is dependent.

Solution: Assign responsibility so that coupling remains low. Use this principle to evaluate alternatives.

Low Coupling : Example

In the SnakesAndLadders example,

- Who should be responsible for checking if the mouth of the Snake has a value larger than the tail?

High Cohesion Pattern

Problem: How to keep objects focused, understandable, and manageable, and as a side effect, support Low Coupling?

Cohesion (or functional cohesion) is a measure of how strongly related and focused the responsibilities of an element are. An element with highly related responsibilities that does not do a tremendous amount of unrelated work has high cohesion.

Solution: Assign responsibility so that cohesion remains high. Use this to evaluate alternatives.

High Cohesion

Classes with low cohesion- do unrelated things- and suffer from following problems:

- Hard to comprehend
- Hard to reuse
- Hard to maintain
- Delicate; constantly affected by change

High Cohesion: Example

In the SnakesAndLadders example,

- Who should be responsible for knowing the end points (tail or mouth) of Snake?
- Who should be responsible for knowing the color of a coin?
- Who should be responsible for knowing the number obtained when a dice is rolled?