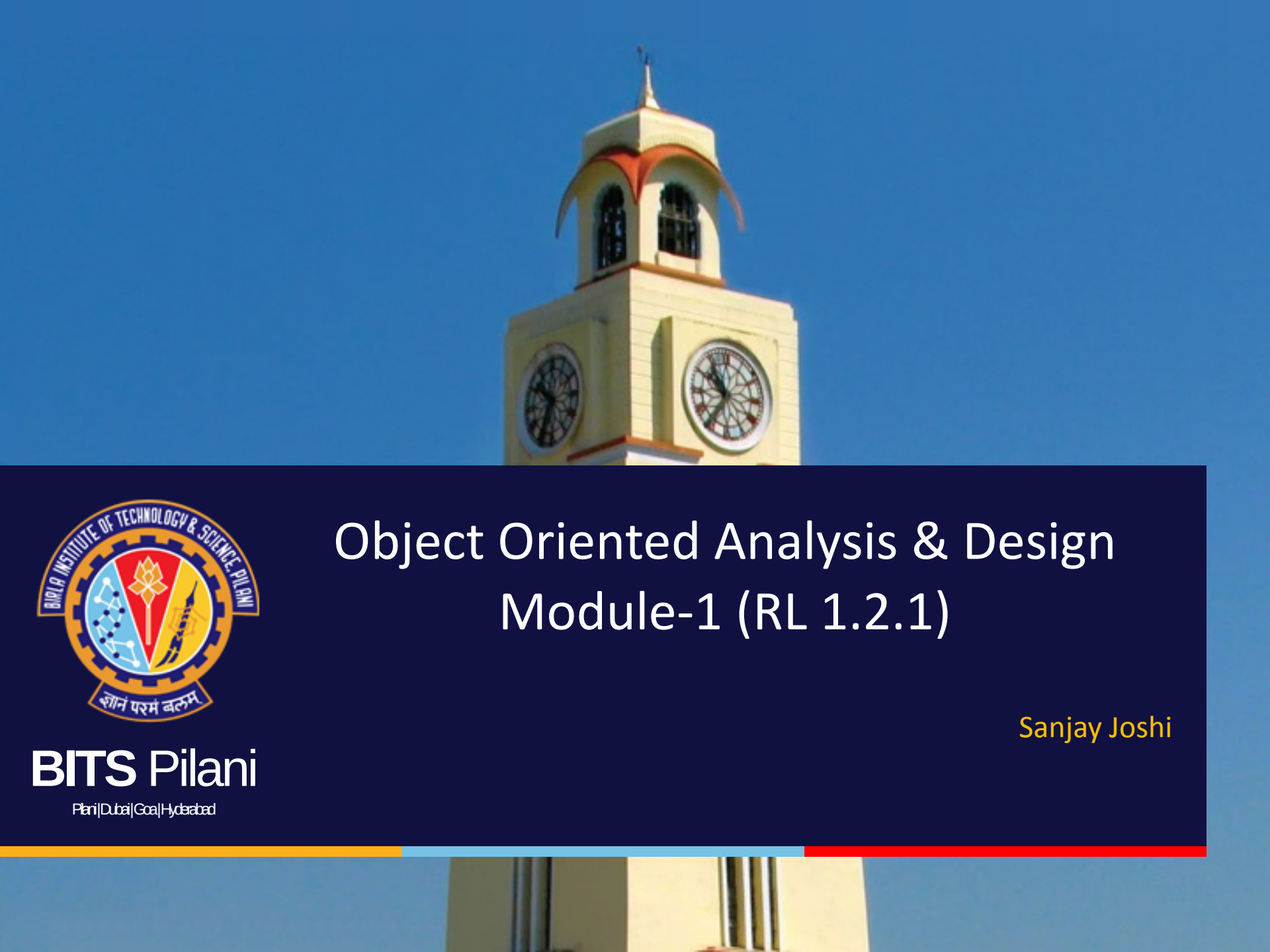


Object Oriented Analysis & Design
Module-1 (RL 1.2.1)

Sanjay Joshi

BITS Pilani

Pilani|Dubai|Goa|Hyderabad

innovate achieve lead

BITS Pilani

Pilani|Dubai|Goa|Hyderabad

Overview of Unified Process (UP)

The Unified Process (UP)

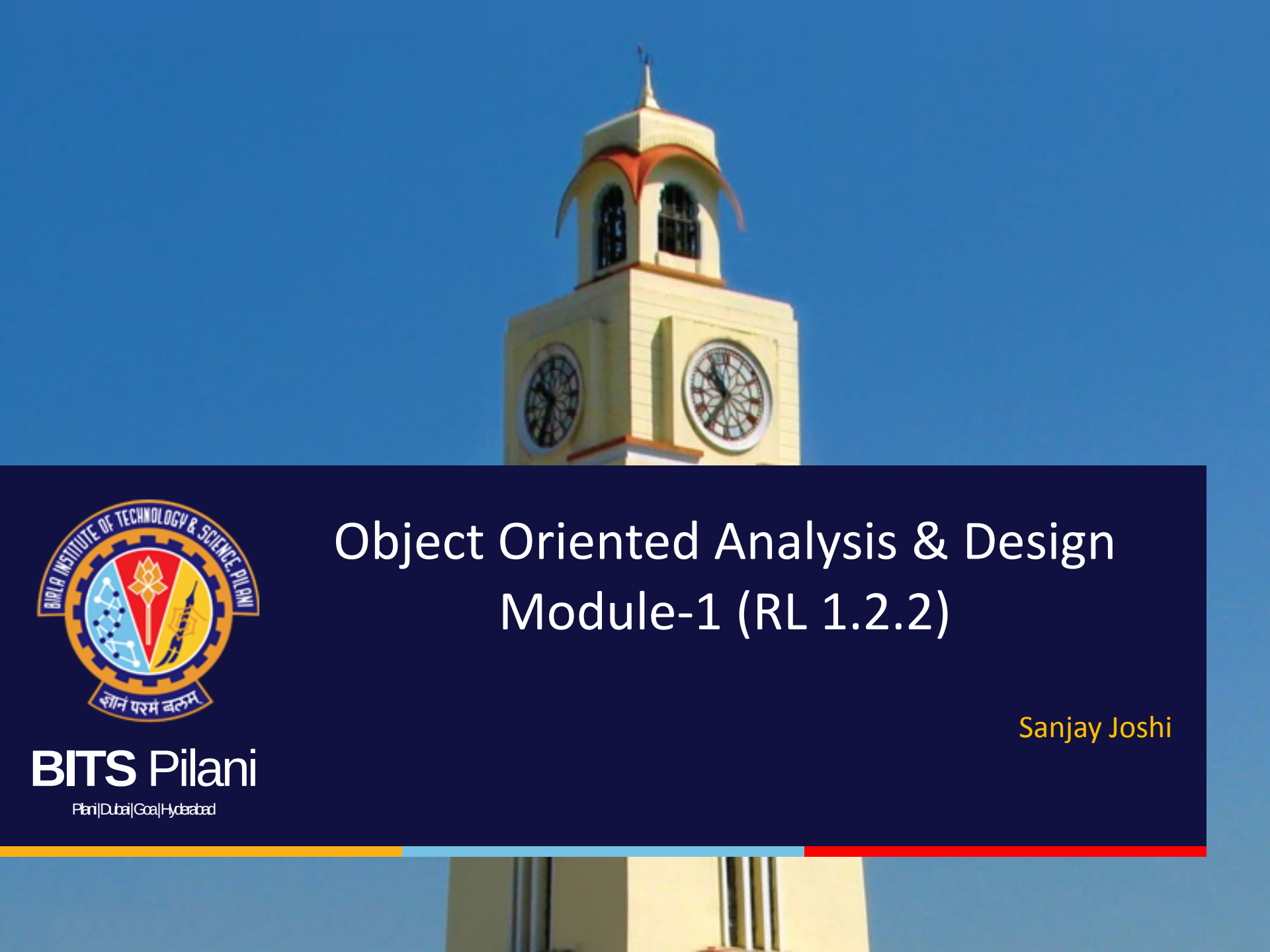
- For simple systems, it might be feasible to sequentially define the whole problem, design the entire solution, build the software, and then test the product.
- For complex and sophisticated systems, this linear approach is not realistic.
- The **Unified Process (UP)** is a process for building object-oriented systems.
- The goal of the UP is to enable the production of high quality software that meets users needs within predictable schedules and budgets.

The Unified Process (UP)

- Development is organized into a series of short fixed-length mini-projects called **iterations**.
- The outcome of each iteration is a tested, integrated and executable system.
- An iteration represents a complete development cycle: it includes its own treatment of requirements, analysis, design, implementation and testing activities.
- UML is compulsory to use for Modeling

Iteration Length and Timeboxing

- The UP recommends short iteration lengths to allow for rapid feedback and adaptation.
- Long iterations increase project risk.
- Iterations are fixed in length (**timeboxed**). If meeting deadline seems to be difficult, then remove tasks or requirements from the iteration and include them in a future iteration.
- The UP recommends that an iteration should be between two and six weeks in duration.



Object Oriented Analysis & Design
Module-1 (RL 1.2.2)

Sanjay Joshi

BITS Pilani

Pilani|Dubai|Goa|Hyderabad

innovate achieve lead

BITS Pilani

Pilani|Dubai|Goa|Hyderabad

UP : An Iterative & Evolutionary Development

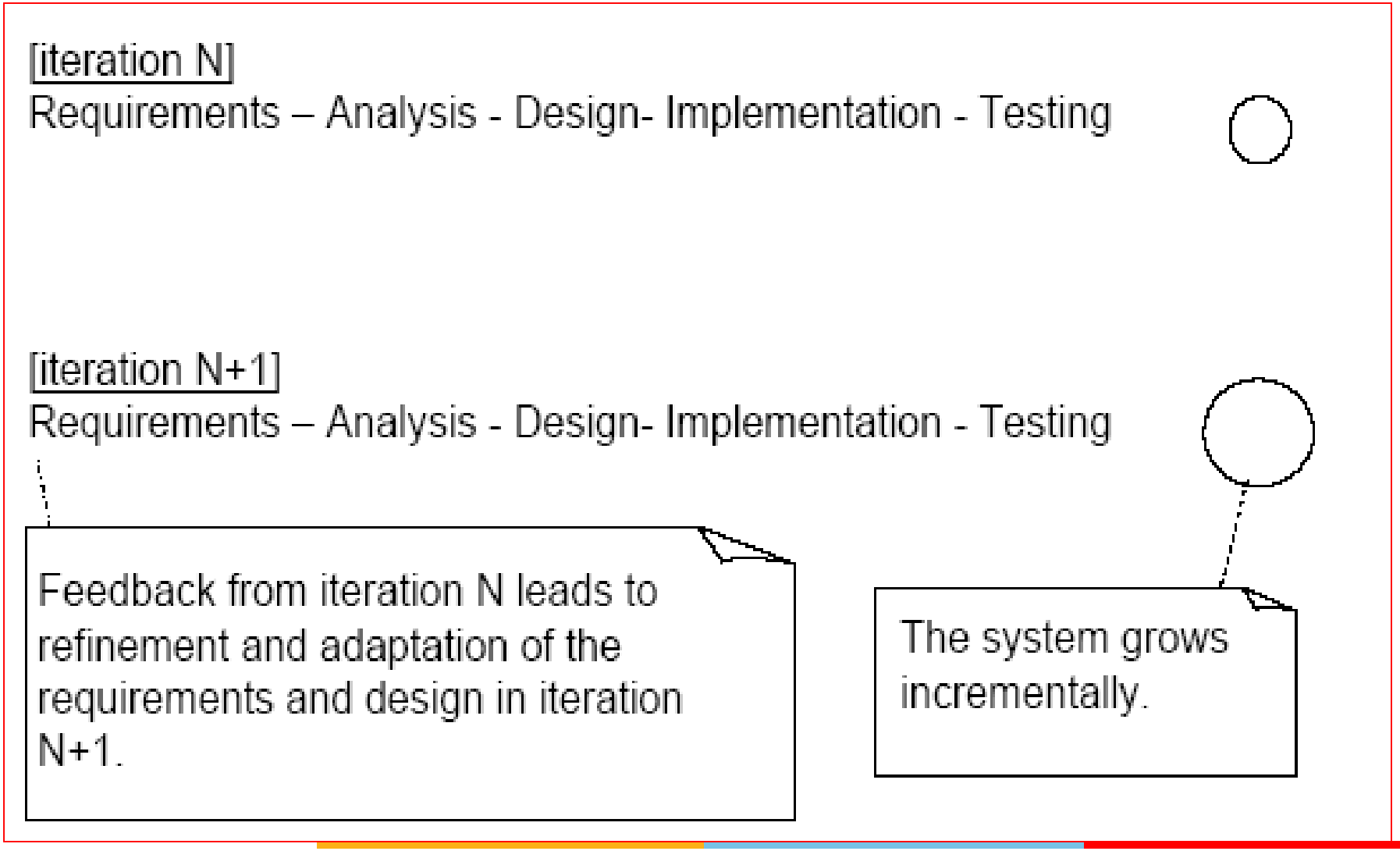
UP: An Iterative & Evolutionary Development

- The iterative lifecycle is based on the successive enlargement and refinement of a system though multiple iterations with feedback and adaptation.
- The system grows incrementally over time, iteration by iteration.
- The system may not be eligible for production deployment until after many iterations.

UP: An Iterative & Evolutionary Development

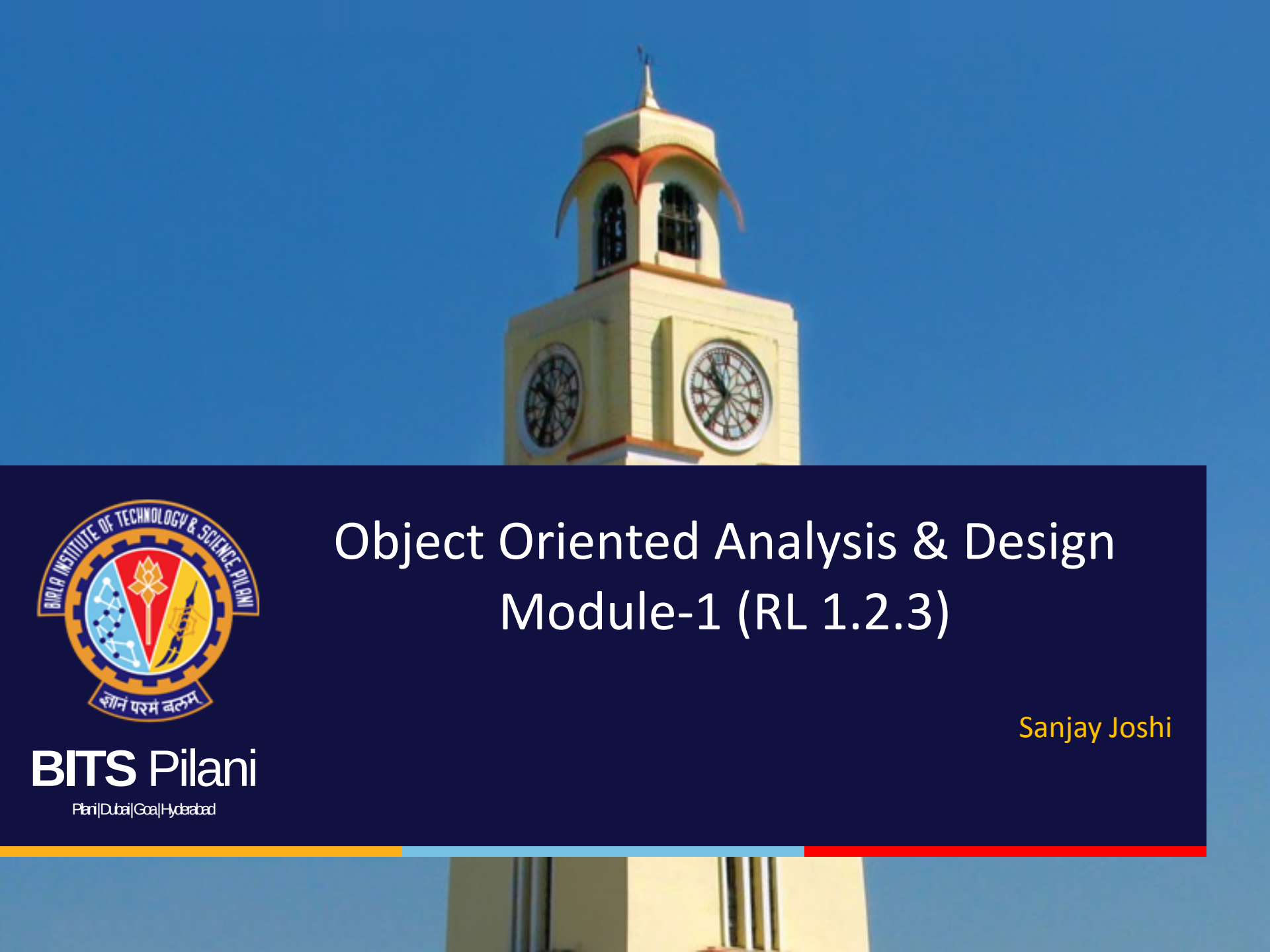
- The output of an iteration is not an experimental prototype but a production subset of the final system.
- Each iteration tackles new requirements and incrementally extends the system.
- An iteration may occasionally revisit existing software and improve it.

UP: An Iterative & Evolutionary Development



Embracing Change

- Stakeholders usually have changing requirements.
- Each iteration involves choosing a small subset of the requirements and quickly design, implement and testing them.
- This leads to rapid feedback, and an opportunity to modify or adapt understanding of the requirements or design.



Object Oriented Analysis & Design
Module-1 (RL 1.2.3)

Sanjay Joshi

BITS Pilani

Pilani|Dubai|Goa|Hyderabad

innovate achieve lead

BITS Pilani

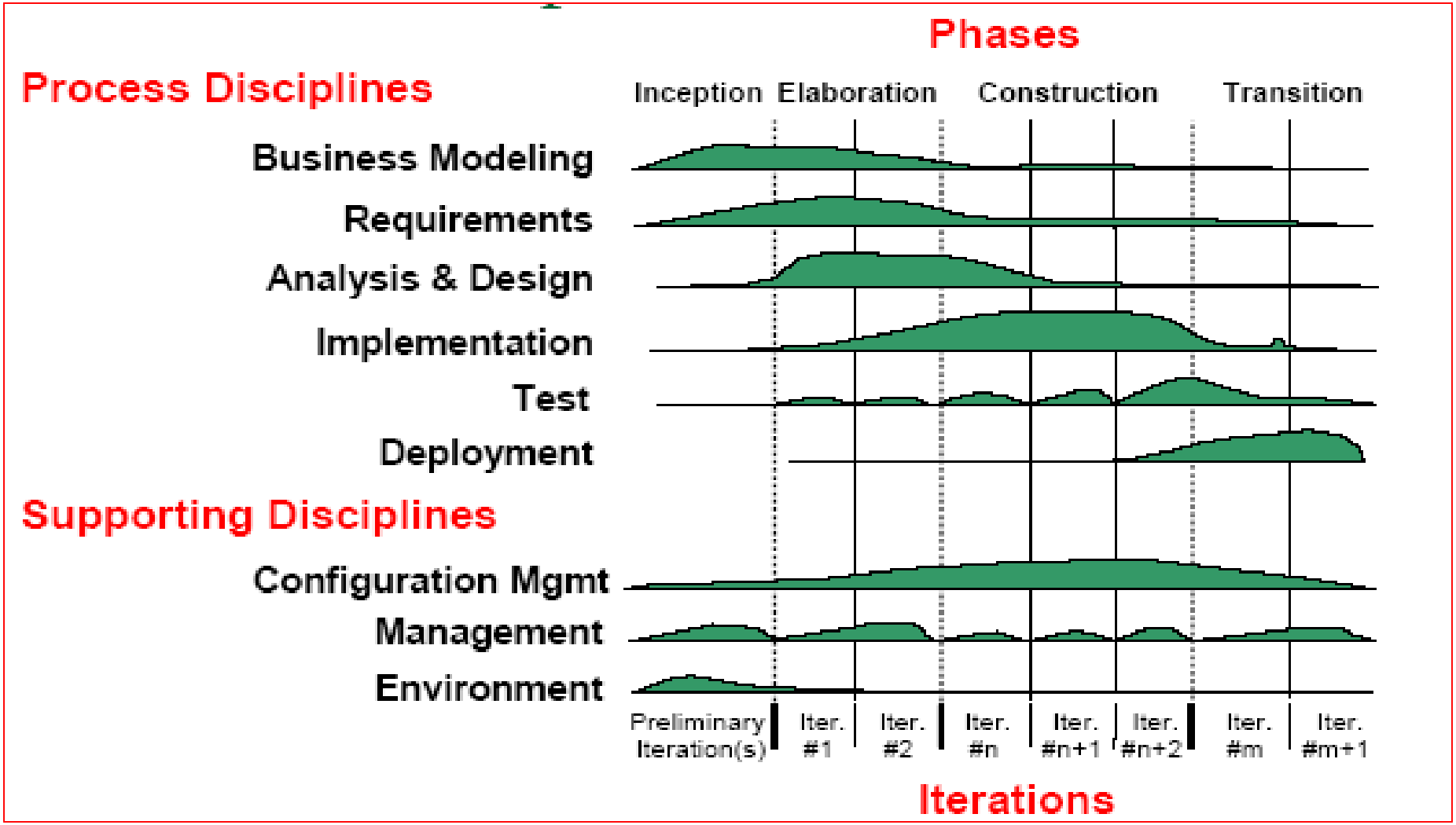
Pilani|Dubai|Goa|Hyderabad

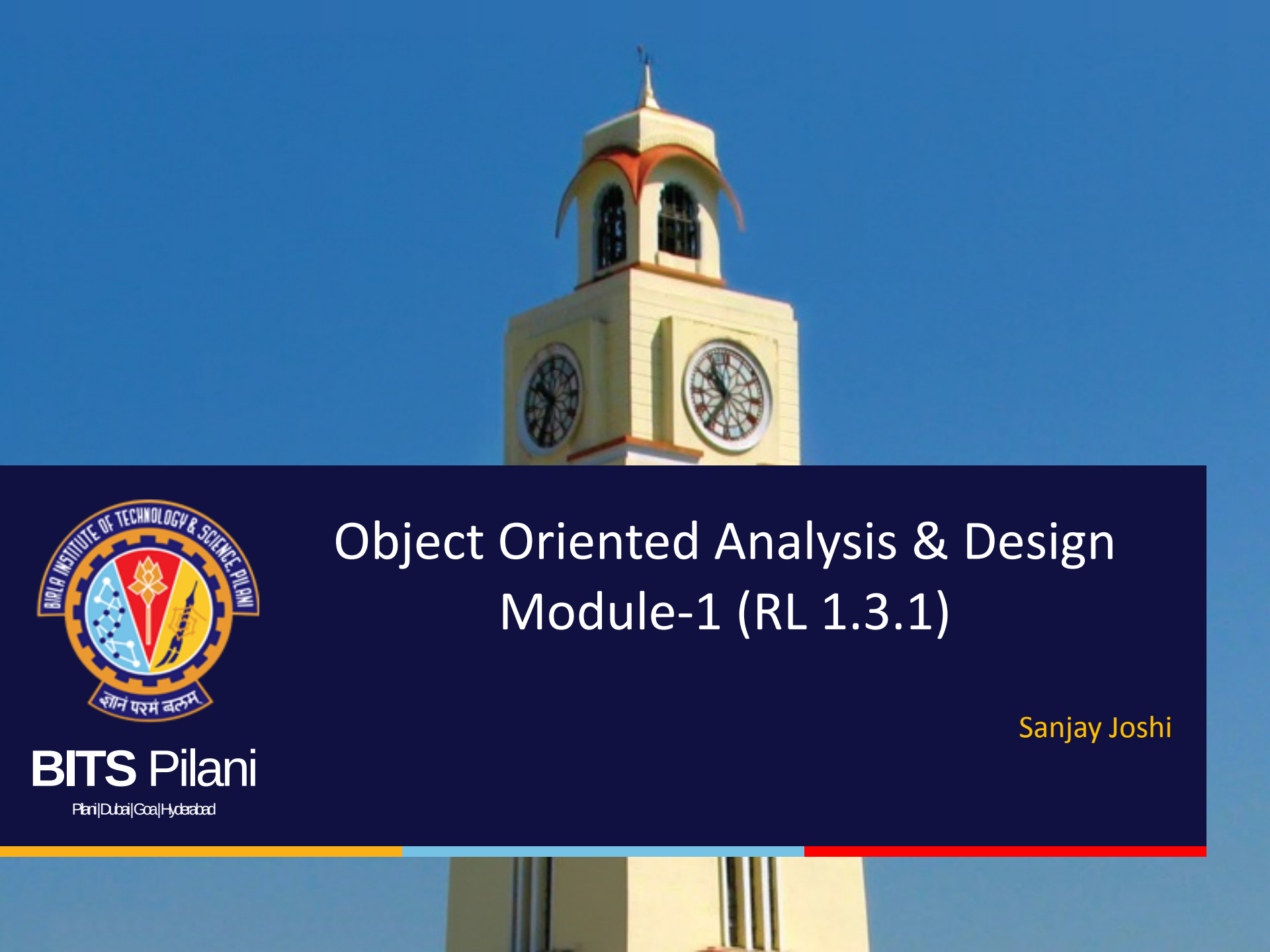
UP : Phases & Disciplines

Phases of the Unified Process

- A UP project organizes the work and iterations across four major phases:
  - Inception - Define the scope of project.
  - Elaboration - Plan project, specify features, baseline architecture.
  - Construction - Build the product
  - Transition - Transition the product into end user community

The UP Disciplines





Object Oriented Analysis & Design
Module-1 (RL 1.3.1)

Sanjay Joshi

BITS Pilani

Pilani|Dubai|Goa|Hyderabad

innovate achieve lead

BITS Pilani

Pilani|Dubai|Goa|Hyderabad

Agile Principles & Manifesto

Common Fears for Developers

- The project will produce the wrong product.
- The project will produce a product of inferior quality.
- The project will be late.
- We'll have to work 80 hour weeks.
- We'll have to break commitments.
- We won't be having fun.

What is “Agility”?

innovate achieve lead

- Effective (rapid and adaptive) response to change
- Effective communication among all stakeholders
- Drawing the customer onto the team
- Organizing a team so that it is in control of the work performed

Yielding …

- Rapid, incremental delivery of software

An Agile Process

innovate achieve lead

- Is driven by customer descriptions of what is required (scenarios)
- Recognizes that plans are short-lived
- Develops software iteratively with a heavy emphasis on construction activities
- Delivers multiple ‘software increments’
- Adapts as changes occur

Principles of Agility

- Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
- Welcome changing requirements, even late in development.
- Business people and developers must work together daily throughout the project.

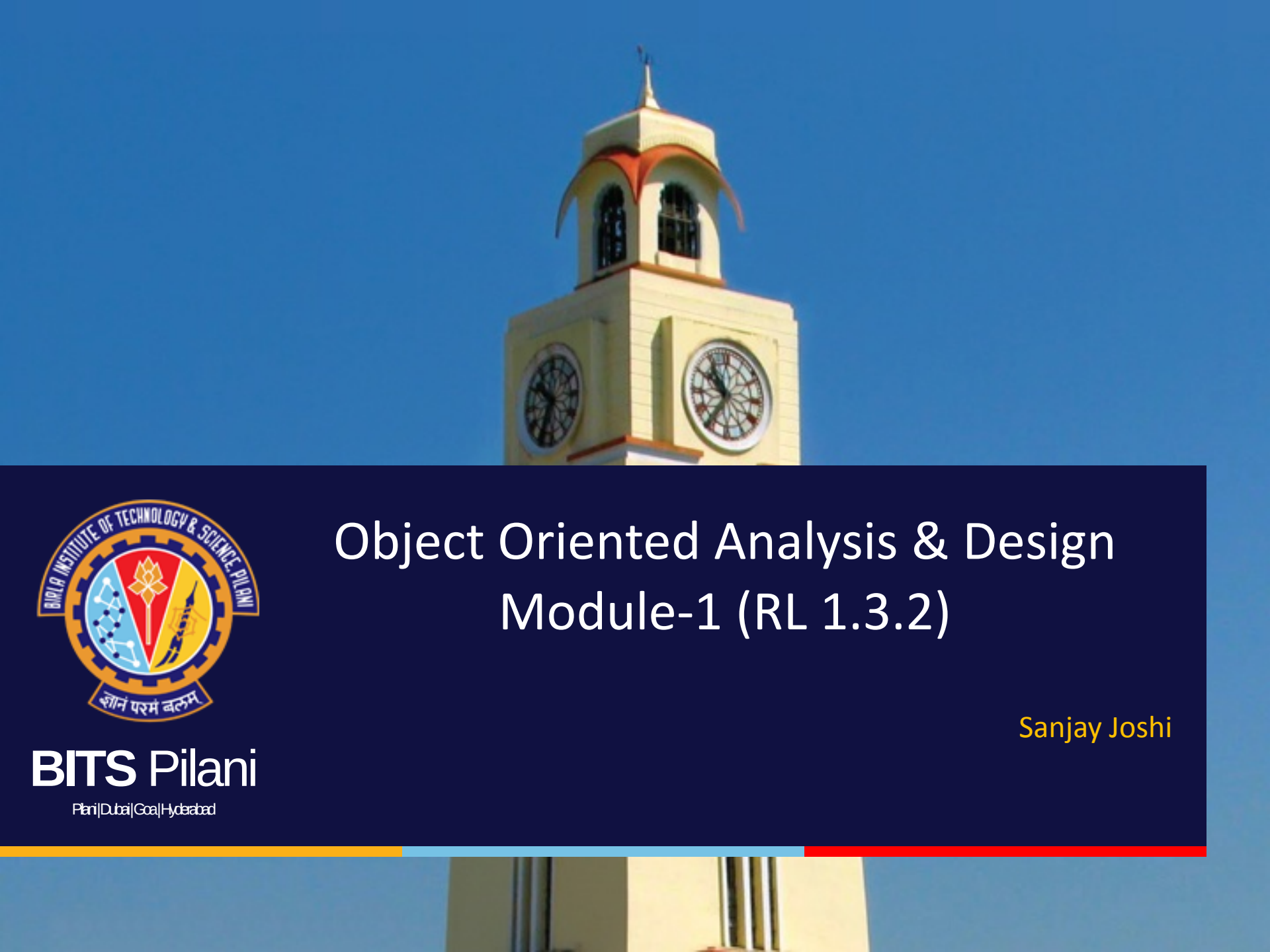
Principles of Agility

- Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
- The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
- Working software is the primary measure of progress.
- Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace continuously.

The Manifesto for Agile Software Development

"We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value"

- ***Individuals and interactions*** **over processes and tools**
- ***Working software*** **over comprehensive documentation**
- ***Customer collaboration*** **over contract negotiation**
- ***Responding to change*** **over following a plan**



Object Oriented Analysis & Design
Module-1 (RL 1.3.2)

Sanjay Joshi

BITS Pilani
Pilani|Dubai|Goa|Hyderabad

innovate achieve lead

BITS Pilani

Pilani|Dubai|Goa|Hyderabad

eXtreme Programming (XP)

eXtreme Programming (XP)

innovate achieve lead

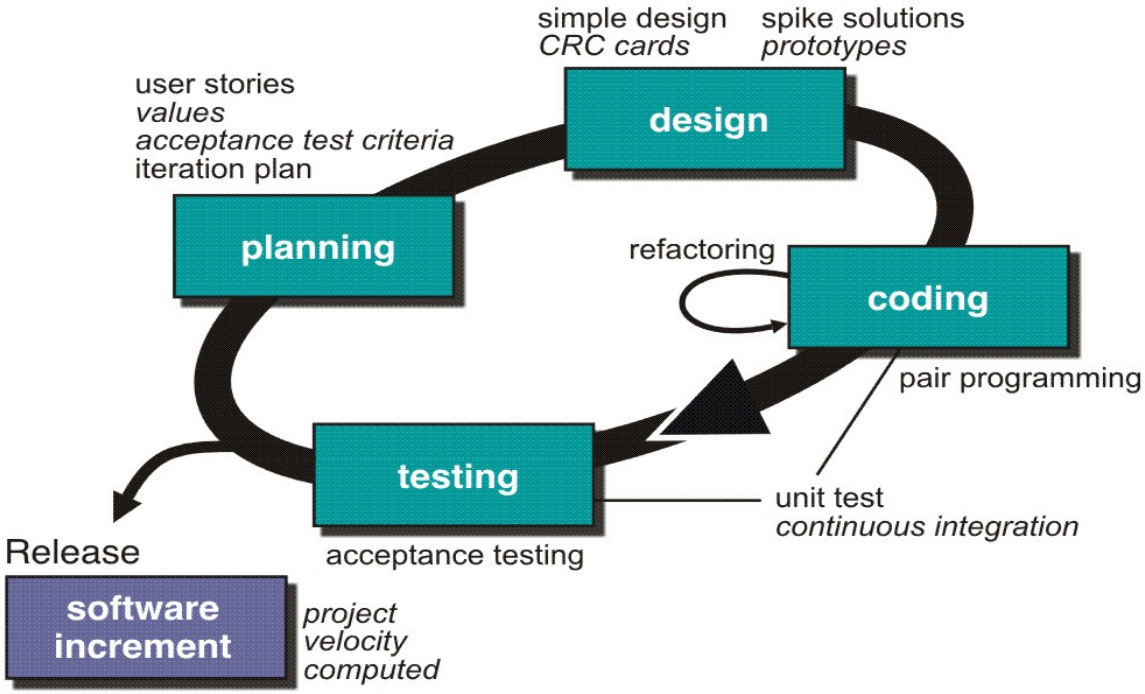
- The most widely used agile process, originally proposed by Kent Beck
- XP Planning
  - Begins with the creation of user stories
  - Agile team assesses each story and assigns a cost
  - Stories are grouped to for a deliverable increment
  - A commitment is made on delivery date
  - After the first increment project velocity (measure of how much work is getting done on your project) is used to help define subsequent delivery dates for other increments

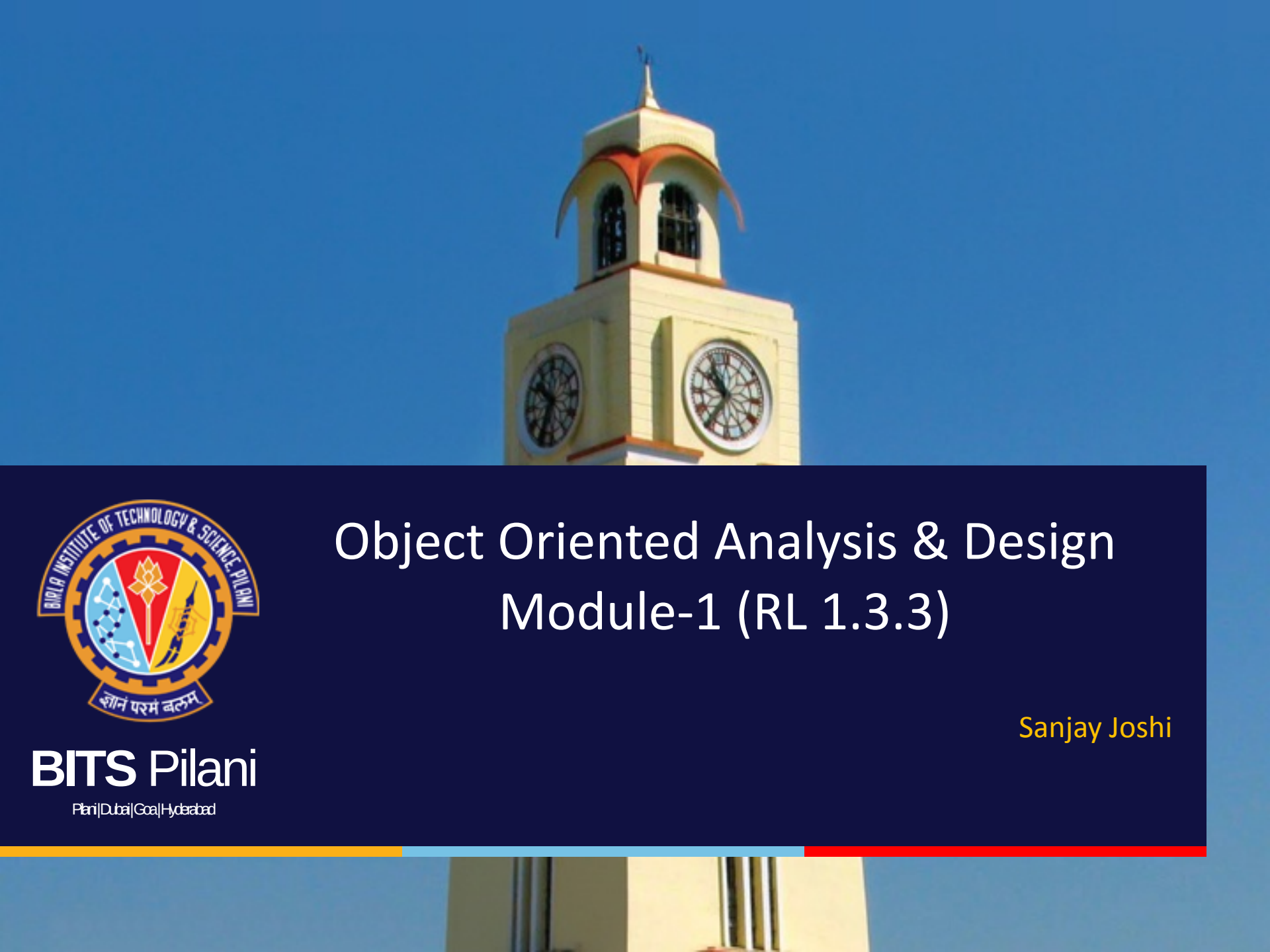
eXtreme Programming (XP)

innovate achieve lead

- XP Design
  - Follows the KIS principle
  - Encourage the use of CRC cards
  - For difficult design problems, suggests the creation of spike solutions — a design prototype
  - Encourages refactoring — an iterative refinement of the internal program design
- XP Coding
  - Recommends the construction of a unit test for a story *before* coding commences
  - Encourages pair programming
- XP Testing
  - All unit tests are executed daily
  - Acceptance tests are defined by the customer and executed to assess customer visible functionality

eXtreme Programming (XP)





Object Oriented Analysis & Design
Module-1 (RL 1.3.3)

Sanjay Joshi

BITS Pilani
Pilani|Dubai|Goa|Hyderabad

innovate achieve lead

BITS Pilani

Pilani|Dubai|Goa|Hyderabad

SCRUM

SCRUM

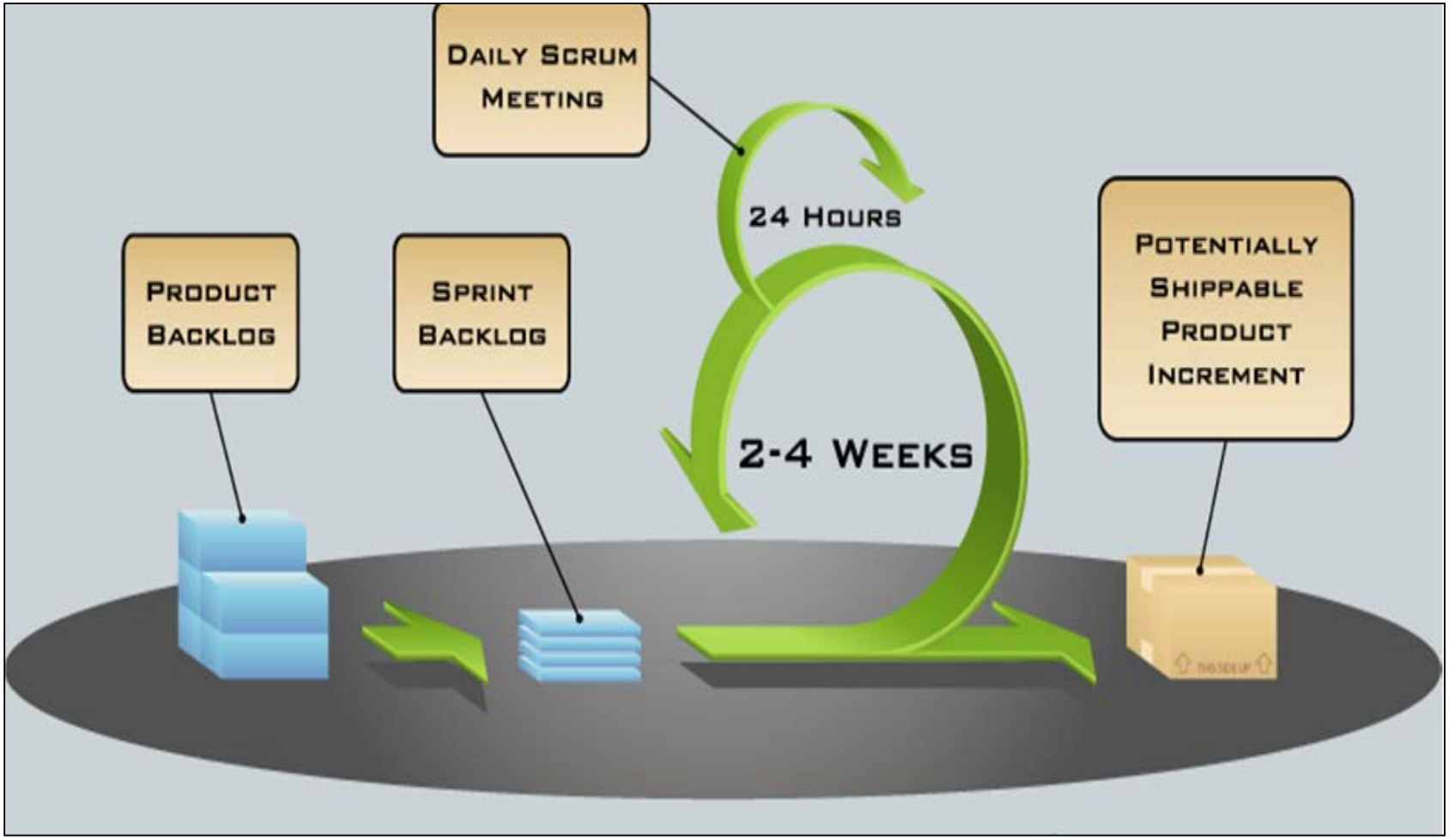
- Project Management Methodology
- Wrapper for existing engineering practices
- Advocates small team (7-9)
- Consists of three roles
  1. Product Owner
  2. Scrum Master
  3. Team
- Tracks progress regularly

Scrum Practices

innovate achieve lead

- Sprint Planning Meeting
- Sprint Daily Scrum
- Sprint Review Meeting
- Sprint Retrospective

SCRUM Process Flow



Object Oriented Analysis & Design
Module-1 (RL 1.3.4)

Sanjay Joshi

BITS Pilani
Pilani|Dubai|Goa|Hyderabad

innovate achieve lead

BITS Pilani

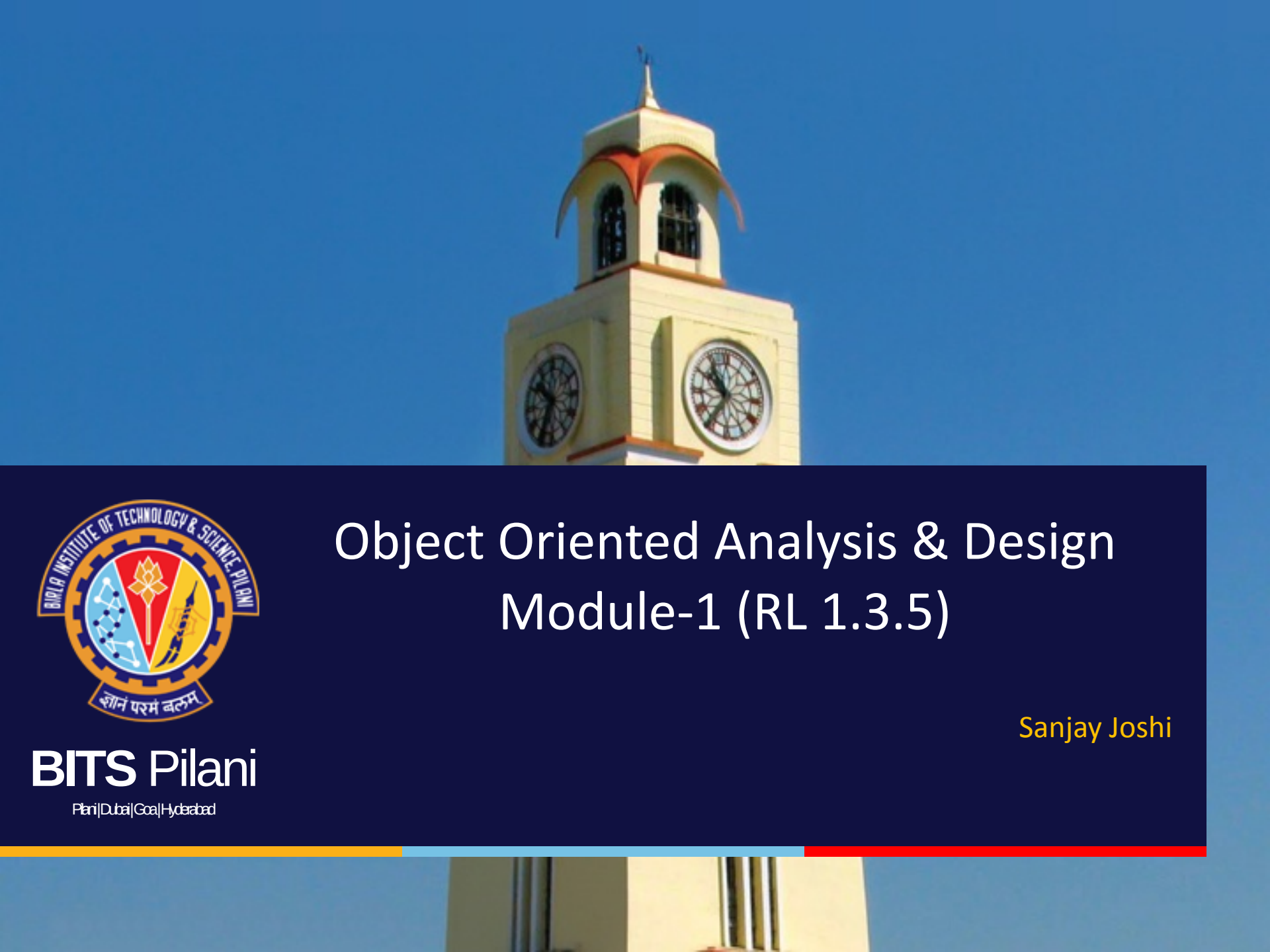
Pilani|Dubai|Goa|Hyderabad

Agile Modelling

Agile Modeling (AM)

innovate achieve lead

- Agile Modeling (AM) is a practice based methodology for effective modeling ans documentation of s/w based systems
- Following are modeling principles
  - Model with a purpose
  - Use multiple models
  - Travel light
  - Content is more important than representation
  - Know the models & tools you use
  - Adapt locally



Object Oriented Analysis & Design
Module-1 (RL 1.3.5)

Sanjay Joshi

BITS Pilani

Pilani|Dubai|Goa|Hyderabad

Test Driven Development

Test Driven Development

innovate achieve lead

- TDD is a technique whereby you write your test cases before you write any implementation code
- Tests drive or dictate the code that is developed
- An indication of “intent”

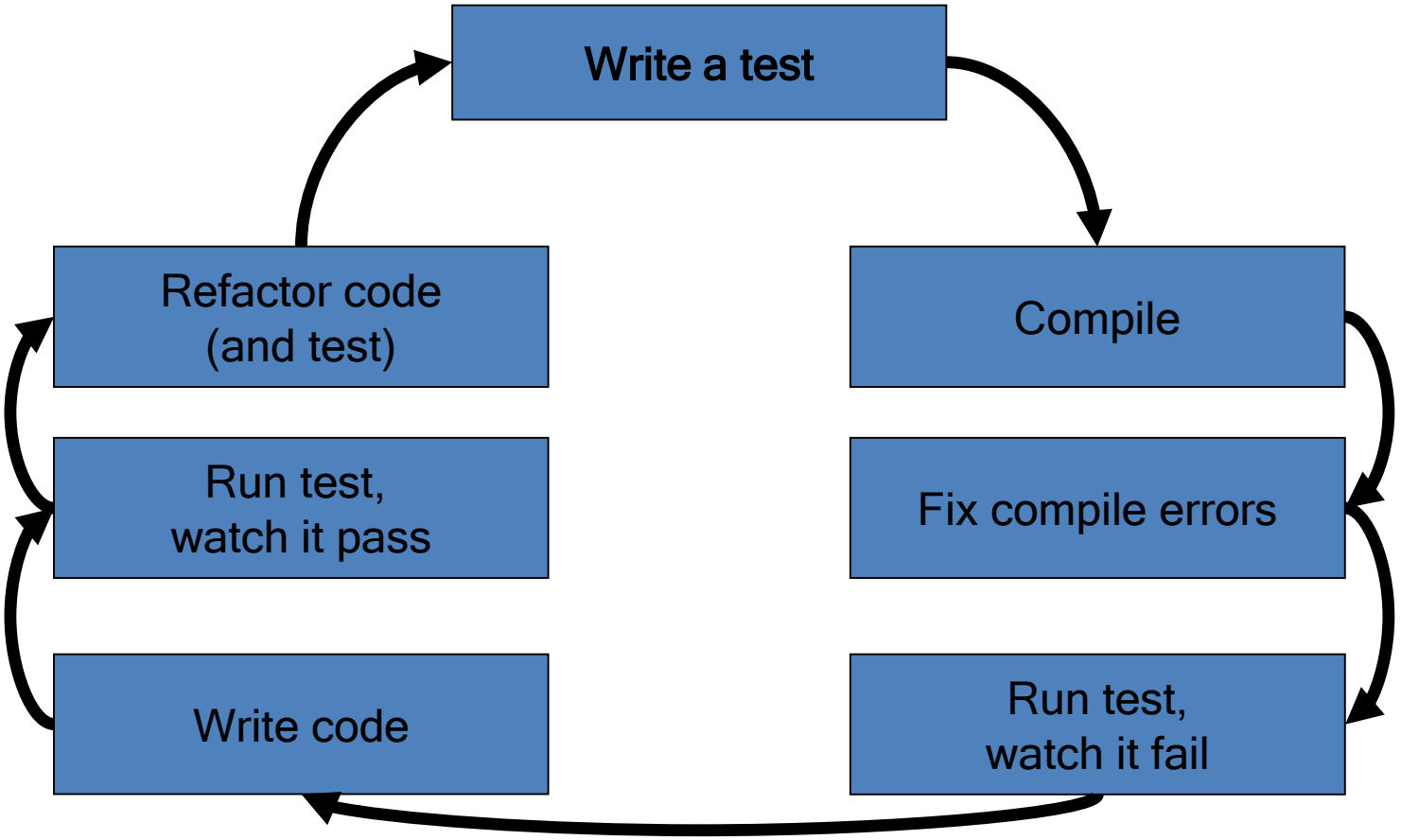
  Tests provide a specification of “what” a piece of code actually does

  – Some might argue that “tests are part of the documentation”

Why is TDD

- TDD can lead to more modularized, flexible, and extensible code
- Clean code
- Leads to better design
- Better code documentation
- More productive
- Good design

Introduction to TDD



Introduction to TDD

- In Extreme Programming Explored (The Green Book), Bill Wake describes the test / code cycle:

1. Write a single test
2. Compile it. It shouldn't compile because you've not written the implementation code
3. Implement just enough code to get the test to compile
4. Run the test and see it **fail**
5. Implement just enough code to get the test to pass
6. Run the test and see it **pass**
7. **Refactor** for clarity
8. Repeat

Introduction to TDD

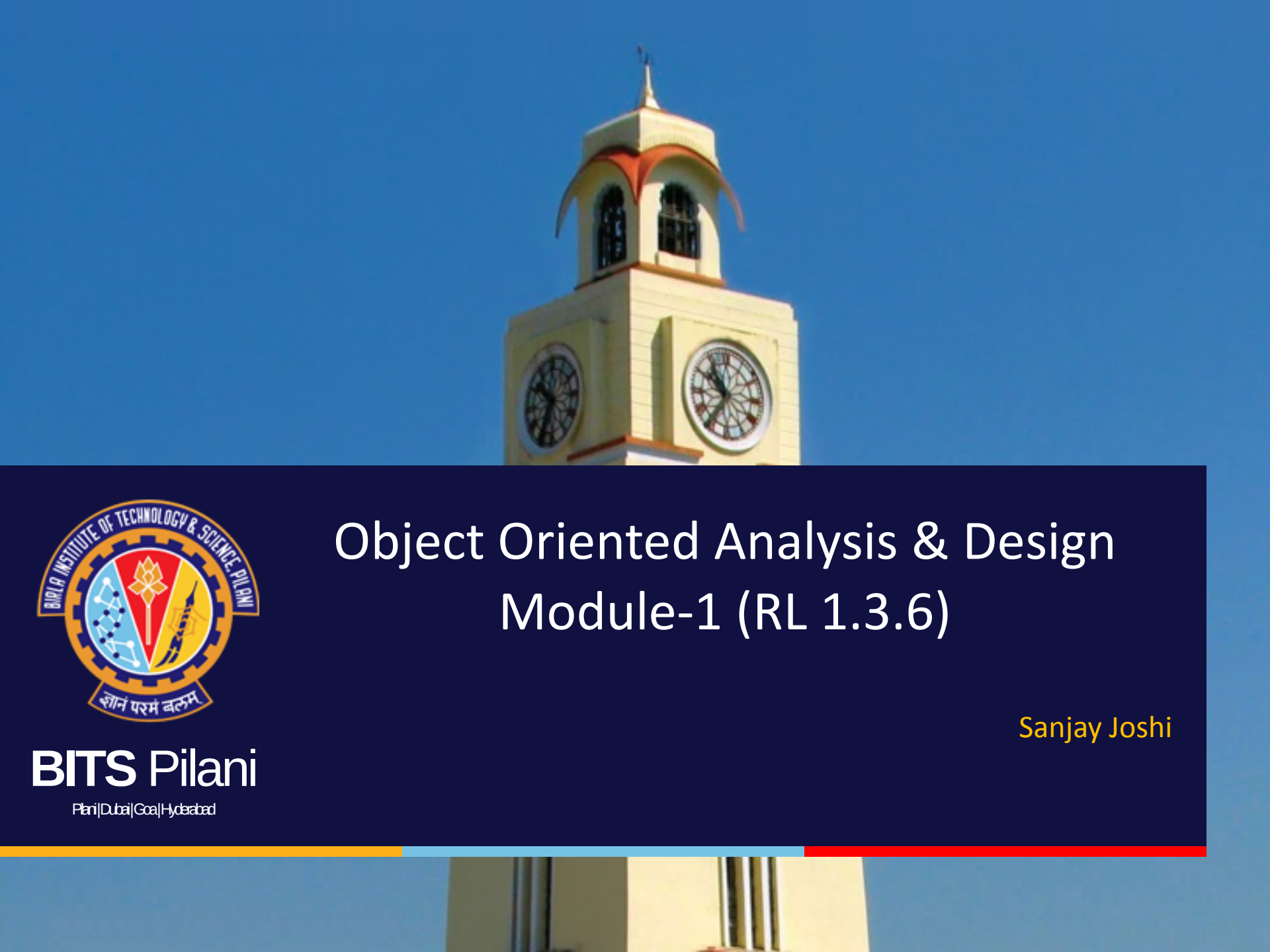
- Example
- Given a string swap the last two characters
- Sample input & output set is

Blank String => Blank String

A => A

AB => BA

ABCD => ABDC



Object Oriented Analysis & Design
Module-1 (RL 1.3.6)

Sanjay Joshi

BITS Pilani

Pilani|Dubai|Goa|Hyderabad

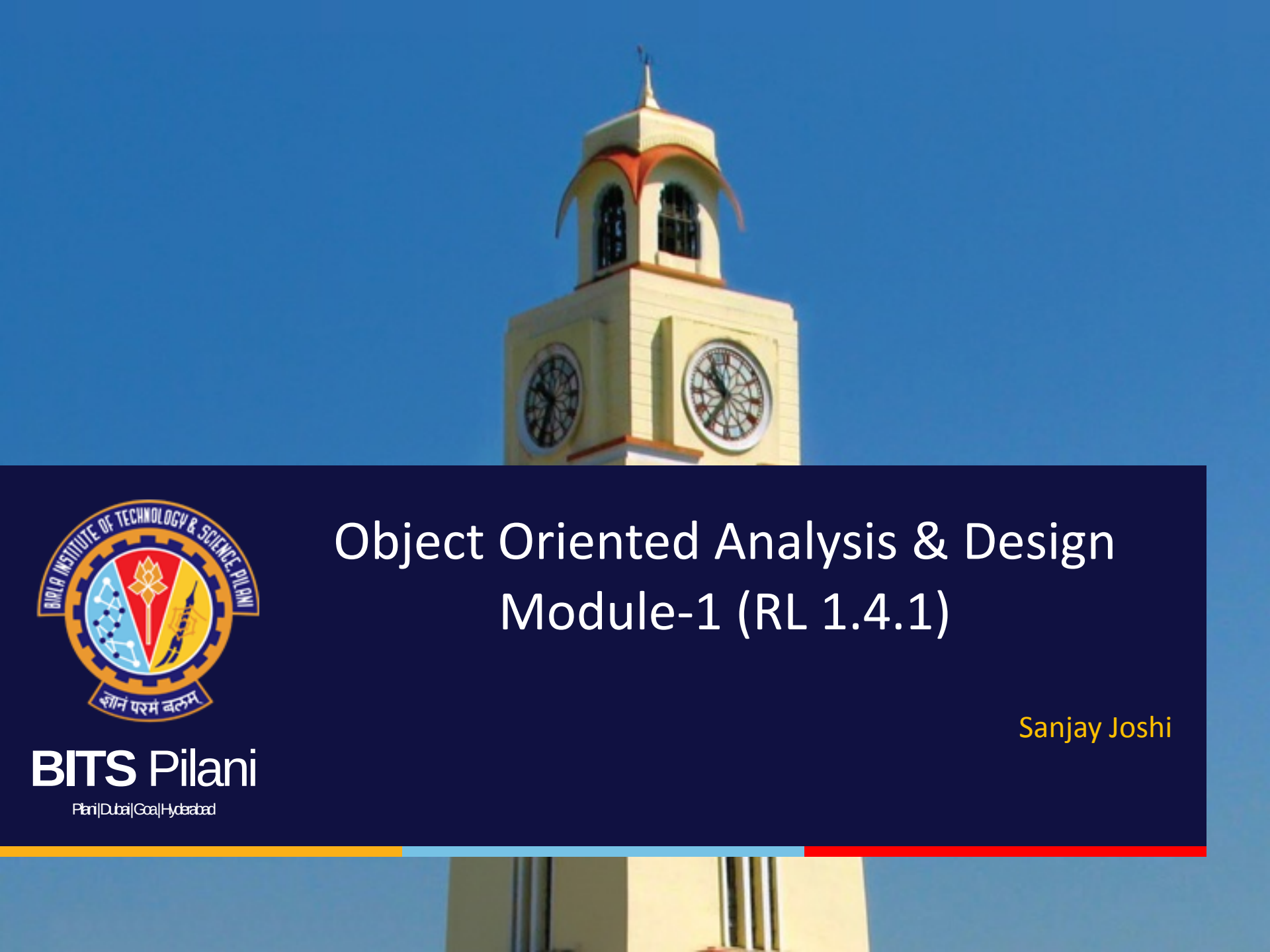
Refactoring & Continuous Integration

Refactoring

- What is Refactoring?
- Revisit, Optimize
- Refactoring in
  - Programming
  - Design

Continuous Integration (CI)

- 2 Ways of testing
  - Big-bang fashion
  - Continious Integration (CI)
- You decide which one is better
- Need to take a call with many parameters



Object Oriented Analysis & Design
Module-1 (RL 1.4.1)

Sanjay Joshi

BITS Pilani

Pilani|Dubai|Goa|Hyderabad

innovate achieve lead

BITS Pilani

Pilani|Dubai|Goa|Hyderabad

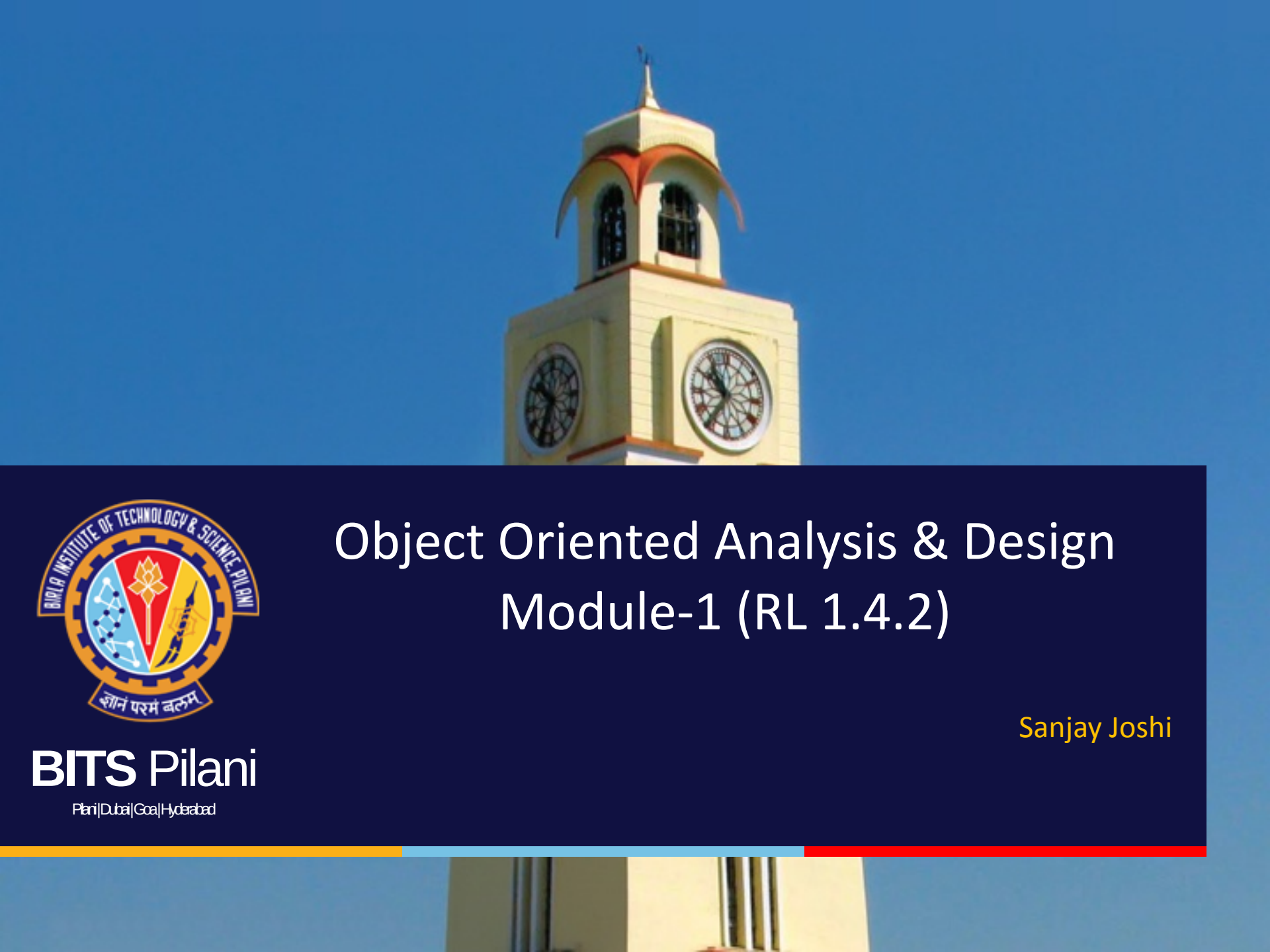
Introduction to OOA & OOD

Object-Oriented Analysis

- An investigation of the problem (rather than how a solution is defined)
- During OO analysis, there is an emphasis on finding and describing the objects (or concepts) in the problem domain.
  - For example, concepts in a Library Information System include *Book* and *Library*.

Object-Oriented Design

- Emphasizes a conceptual solution that fulfils the requirements.
- Need to define software objects and how they collaborate to fulfill the requirements.
  - For example, in the Library Information System, a *Book* software object may have a *title* attribute and a *getChapter* method.
- Designs are implemented in a programming language.
  - In the example, we will have a *Book* class in Java.



Object Oriented Analysis & Design
Module-1 (RL 1.4.2)

Sanjay Joshi

BITS Pilani

Pilani|Dubai|Goa|Hyderabad

innovate achieve lead

BITS Pilani

Pilani|Dubai|Goa|Hyderabad

Overview of UML

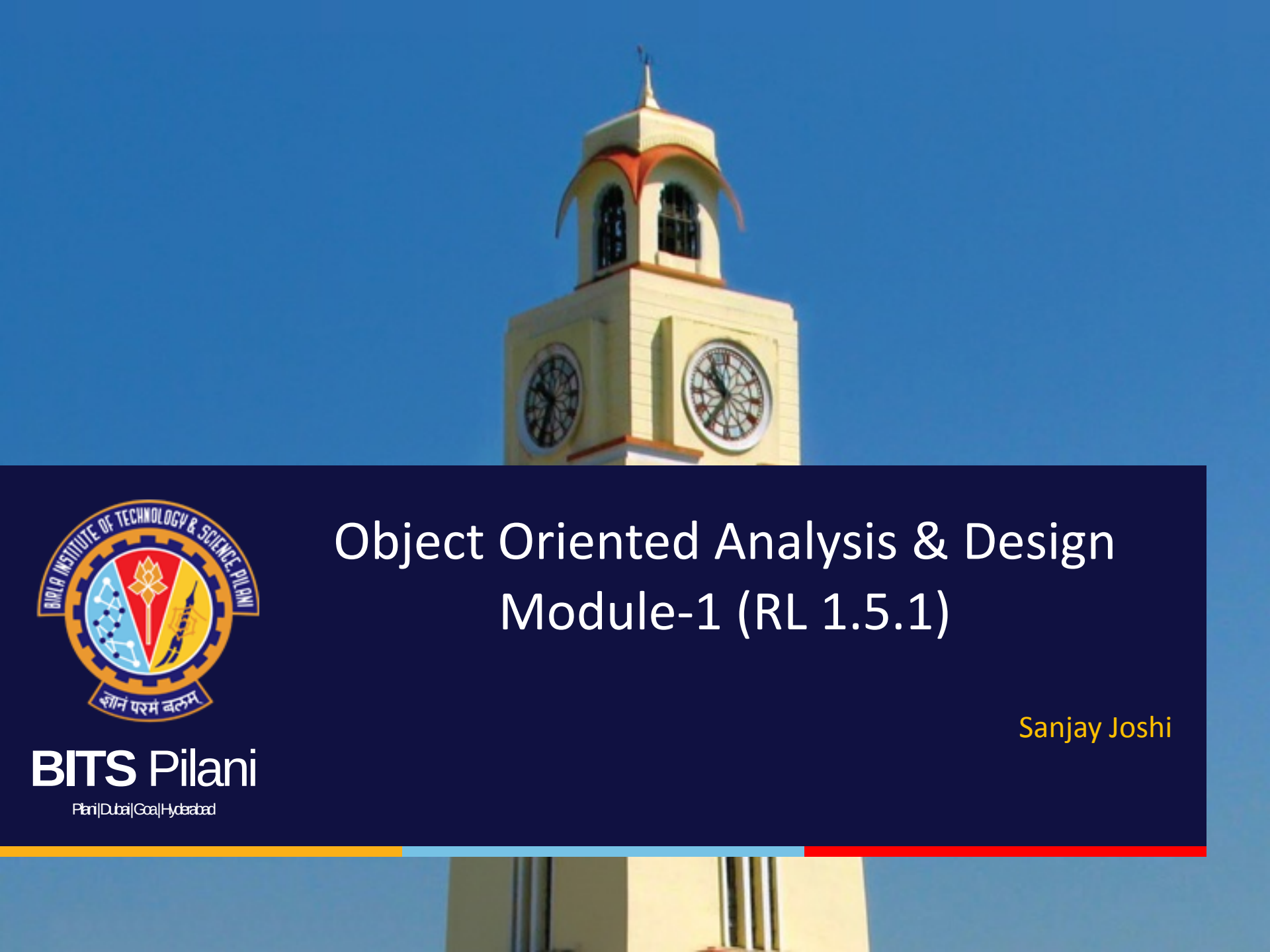
What is UML?

innovate achieve lead

- Unified Modeling Language

- modeling language to draw diagrams in Uniform way Object Oriented Analysis & Design

- Need of UML
- Who uses UML?



Object Oriented Analysis & Design
Module-1 (RL 1.5.1)

Sanjay Joshi

BITS Pilani

Pilani|Dubai|Goa|Hyderabad

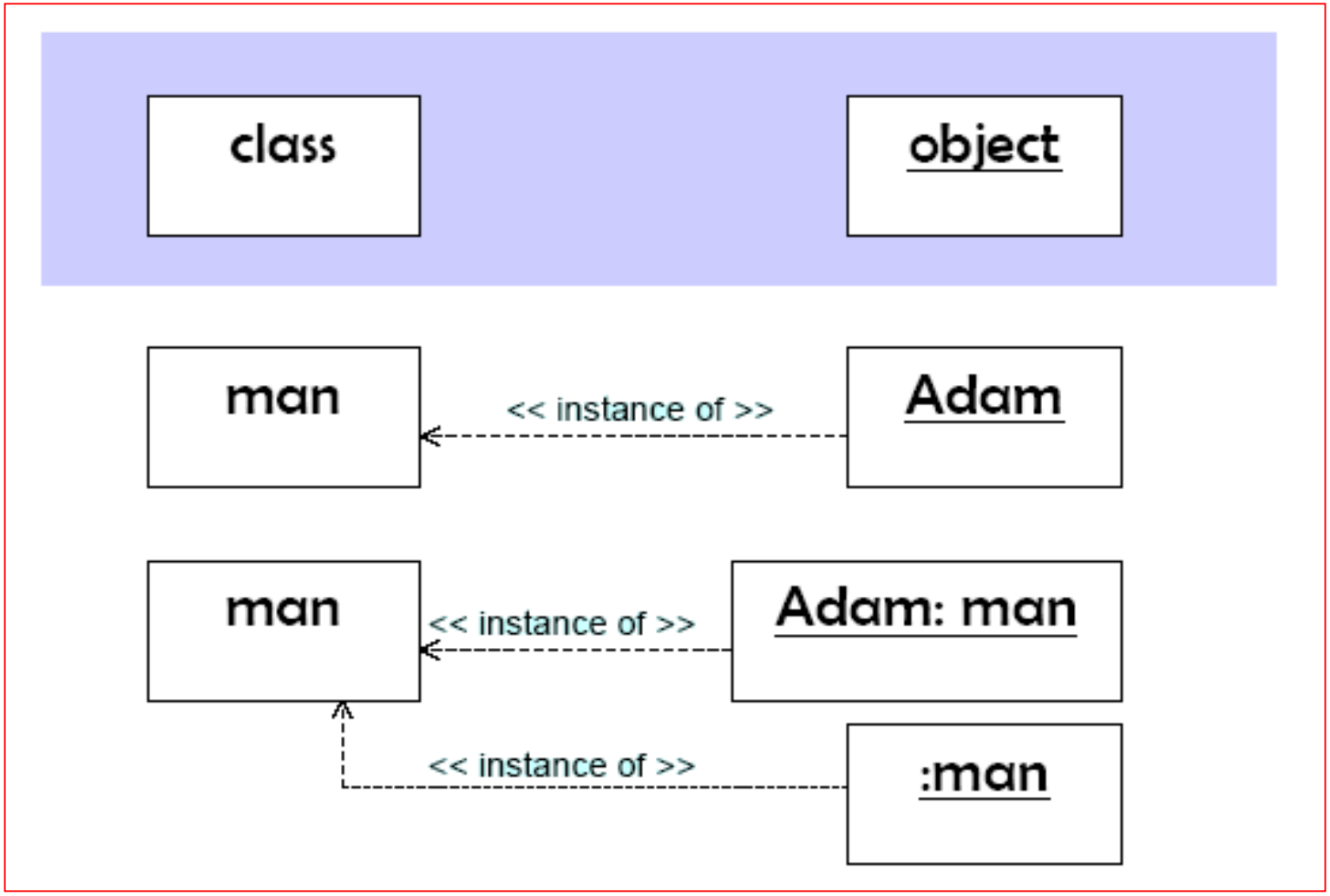
innovate achieve lead

BITS Pilani

Pilani|Dubai|Goa|Hyderabad

Concept of Class & Object

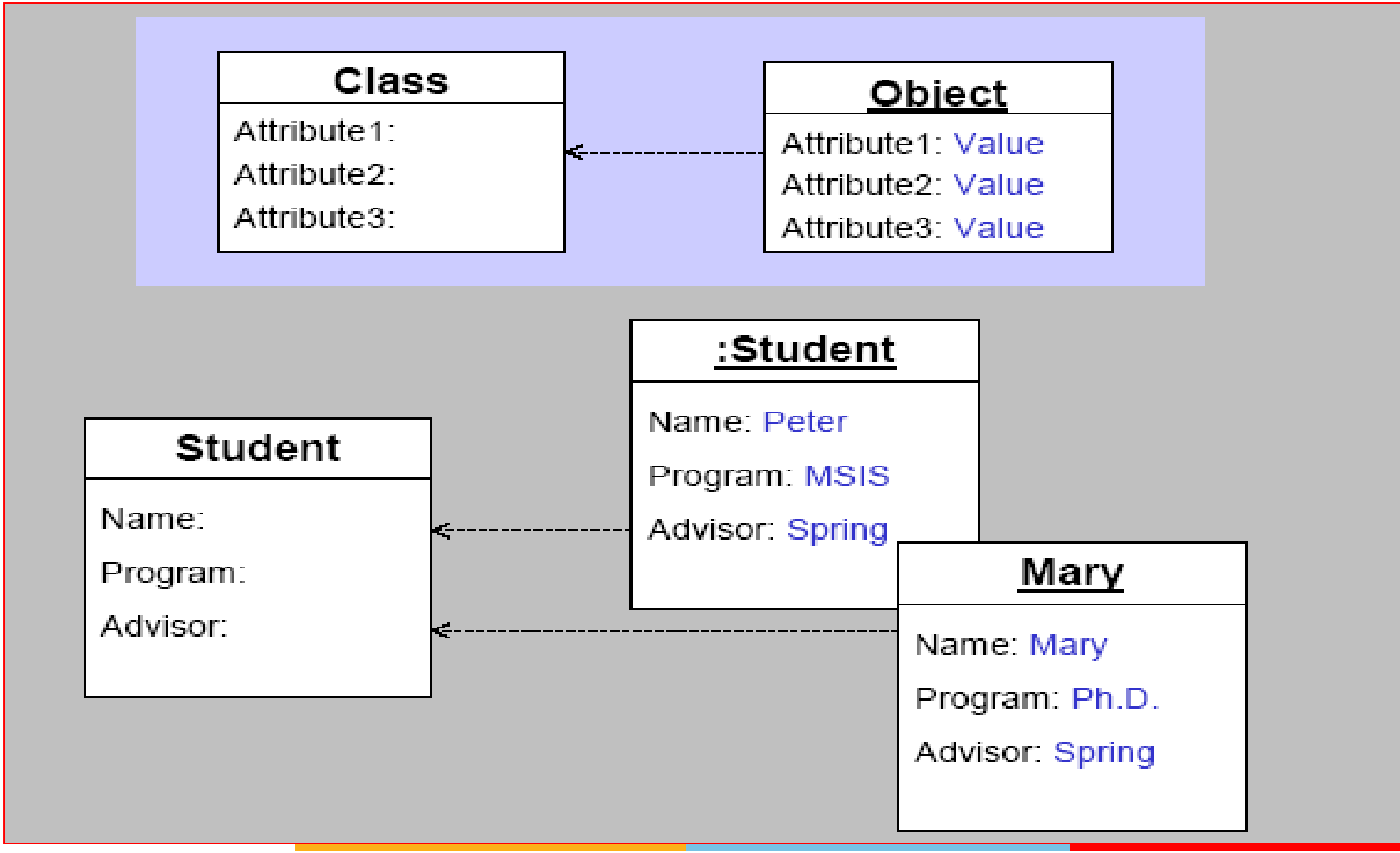
Class and Object: in UML



Attributes

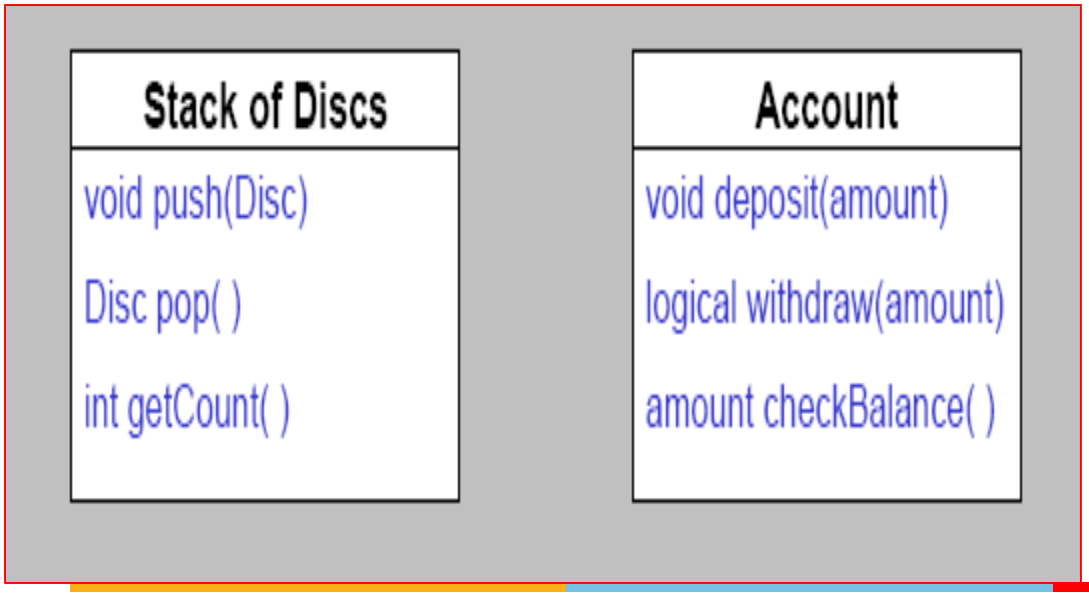
- The class captures the abstraction of properties in the set of objects.
- **An attribute** of a class is identified by **name**, and it identifies a property of the objects of the class, for which each object takes a **value**.

Attributes: in UML

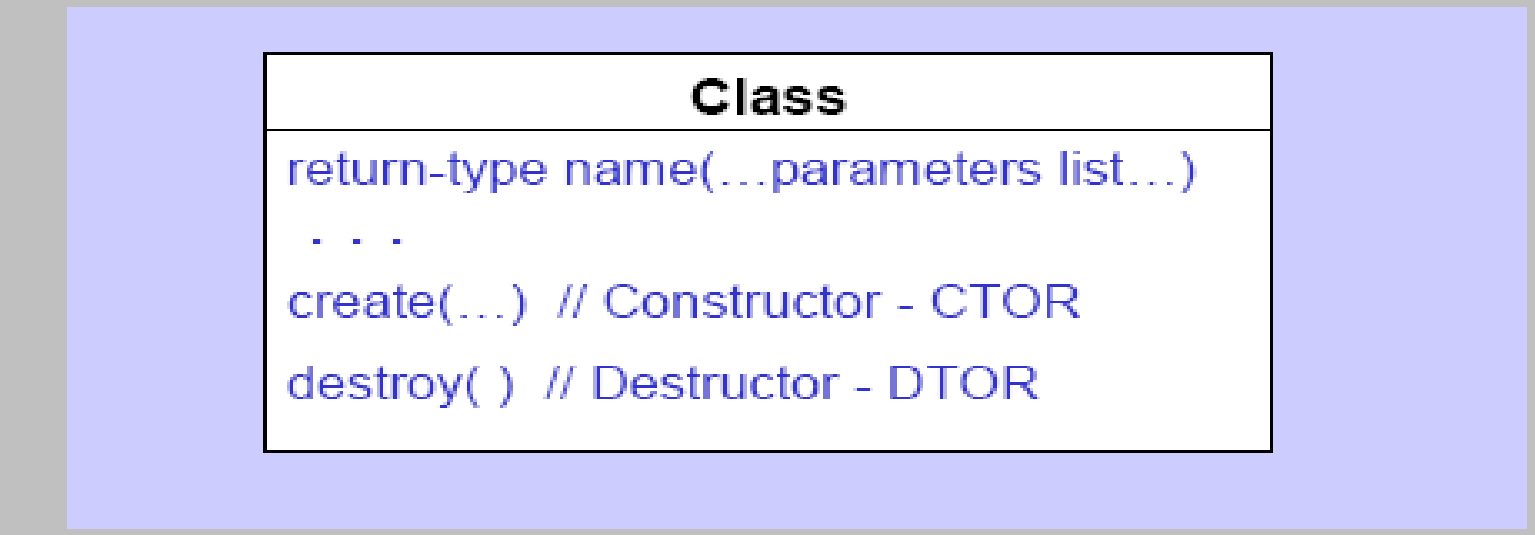


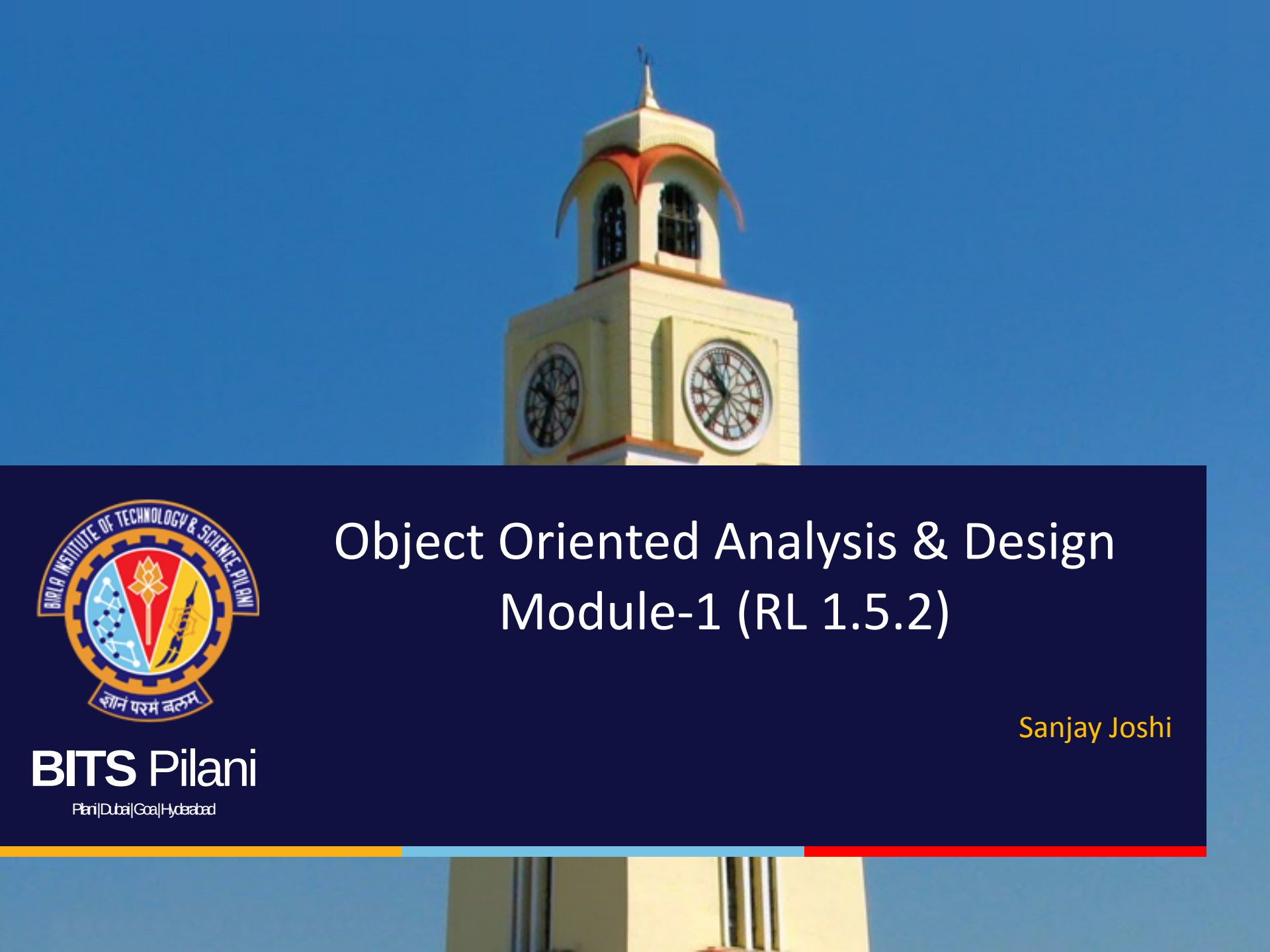
Operations

- The operations are the “responsibilities” – the things we can ask an object to do.
- Note that these are classes
- Should we have open( ) and close( ) with Account?



Operations: in UML





Object Oriented Analysis & Design
Module-1 (RL 1.5.2)

Sanjay Joshi

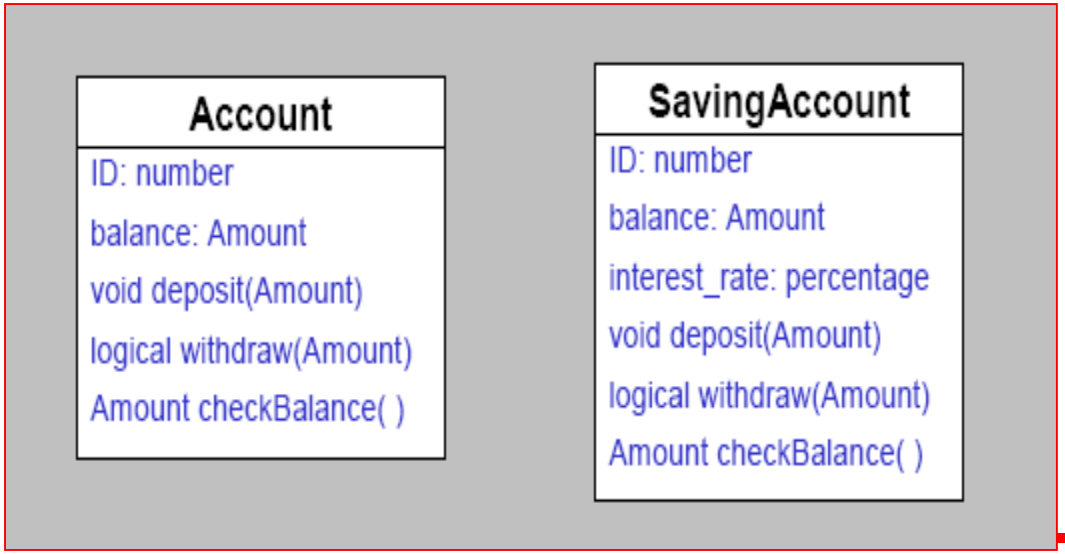
BITS Pilani

Pilani|Dubai|Goa|Hyderabad

Class Relationships in UML

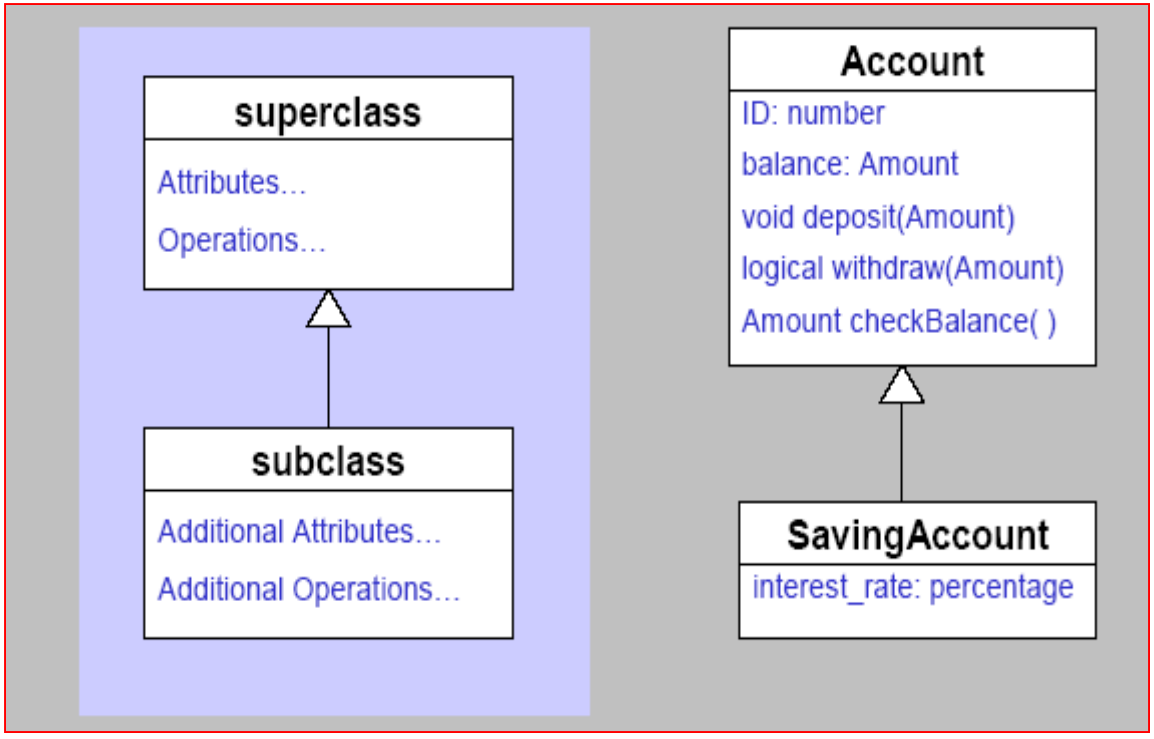
Inheritance: superclass & subclass

- A class may be the subset of another class…
- A saving account is also an account.
- Not all accounts are saving accounts.



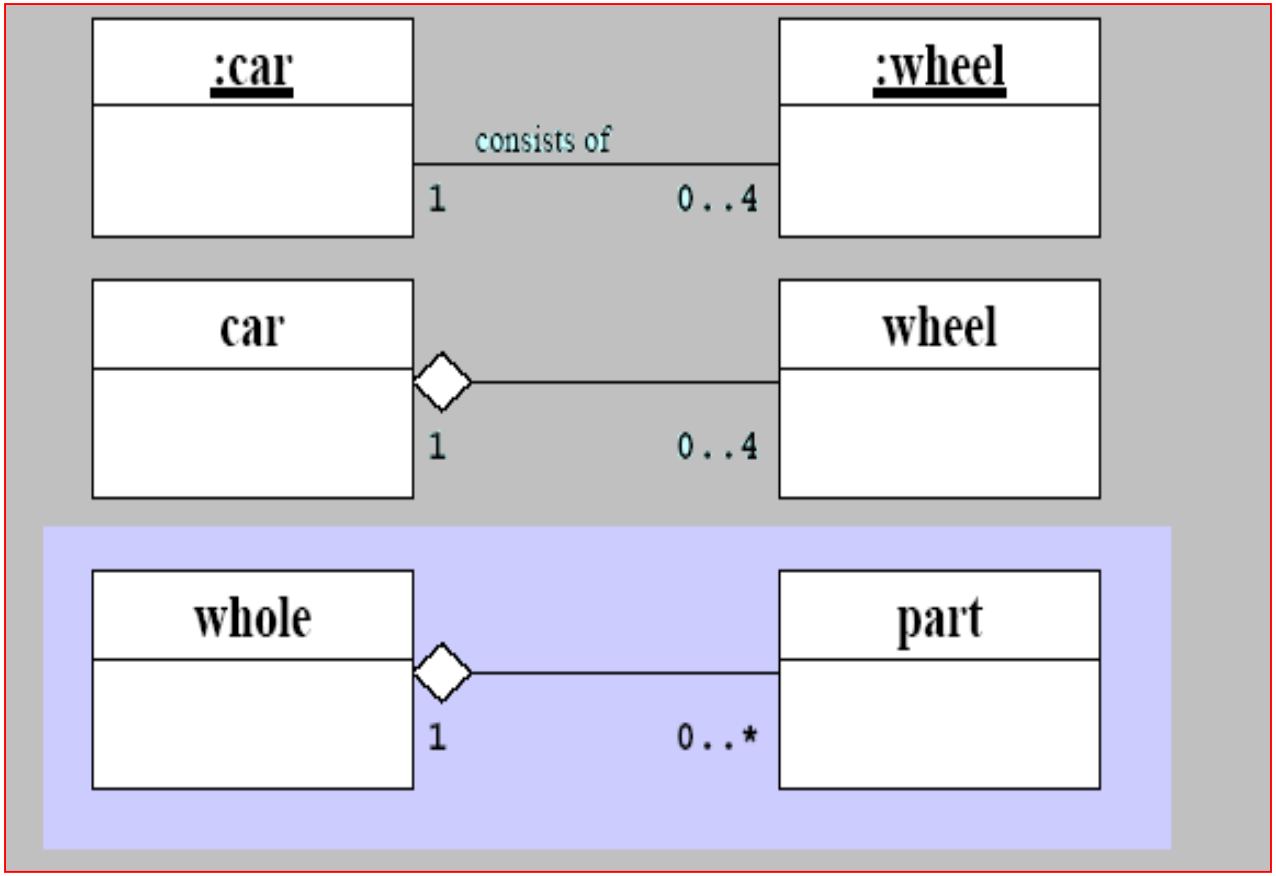
Inheritance in UML

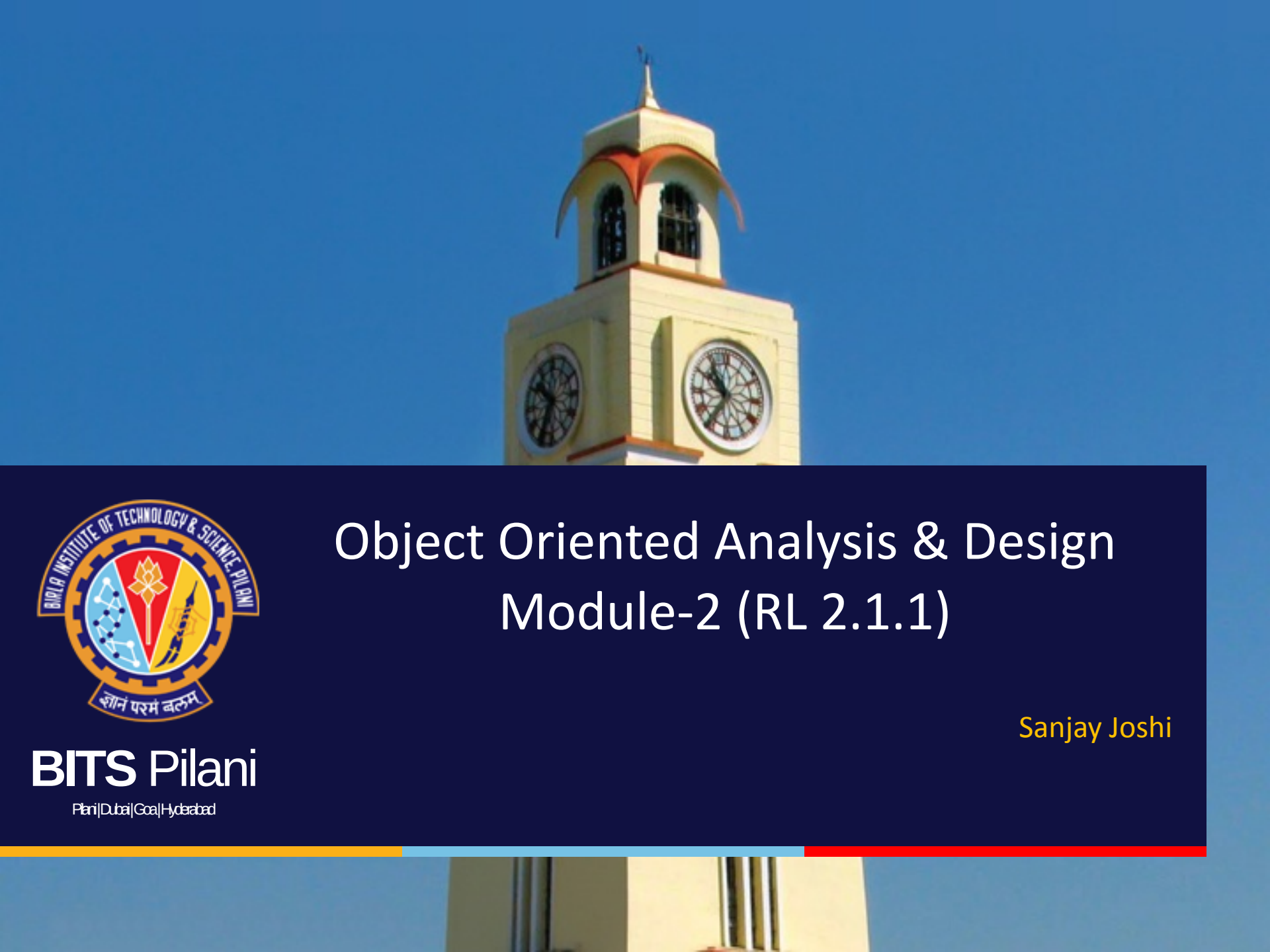
- The subclass inherits from the superclass the attributes as well as the operations.



Aggregation: in UML

- An object may be part of another object…





Object Oriented Analysis & Design
Module-2 (RL 2.1.1)

Sanjay Joshi

BITS Pilani

Pilani|Dubai|Goa|Hyderabad

innovate achieve lead

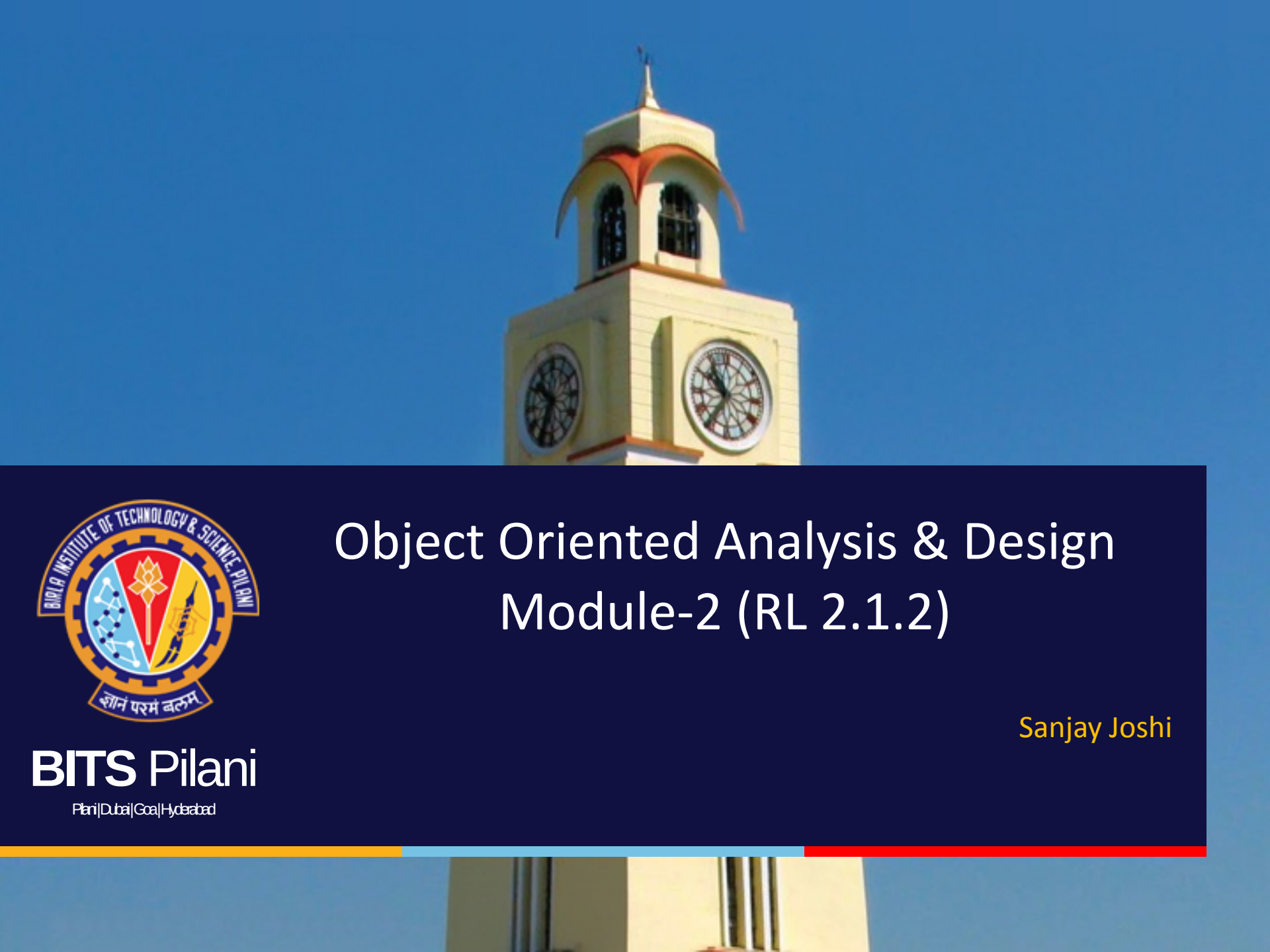
BITS Pilani

Pilani|Dubai|Goa|Hyderabad

Point of Sale (PoS) Case Study

PoS (Point Of Sale) System

- Mall System
- Customer purchasing goods
- Cashier making entry in the system
- Salesperson is having commission
- Customer can return goods as well.



Object Oriented Analysis & Design
Module-2 (RL 2.1.2)

Sanjay Joshi

BITS Pilani

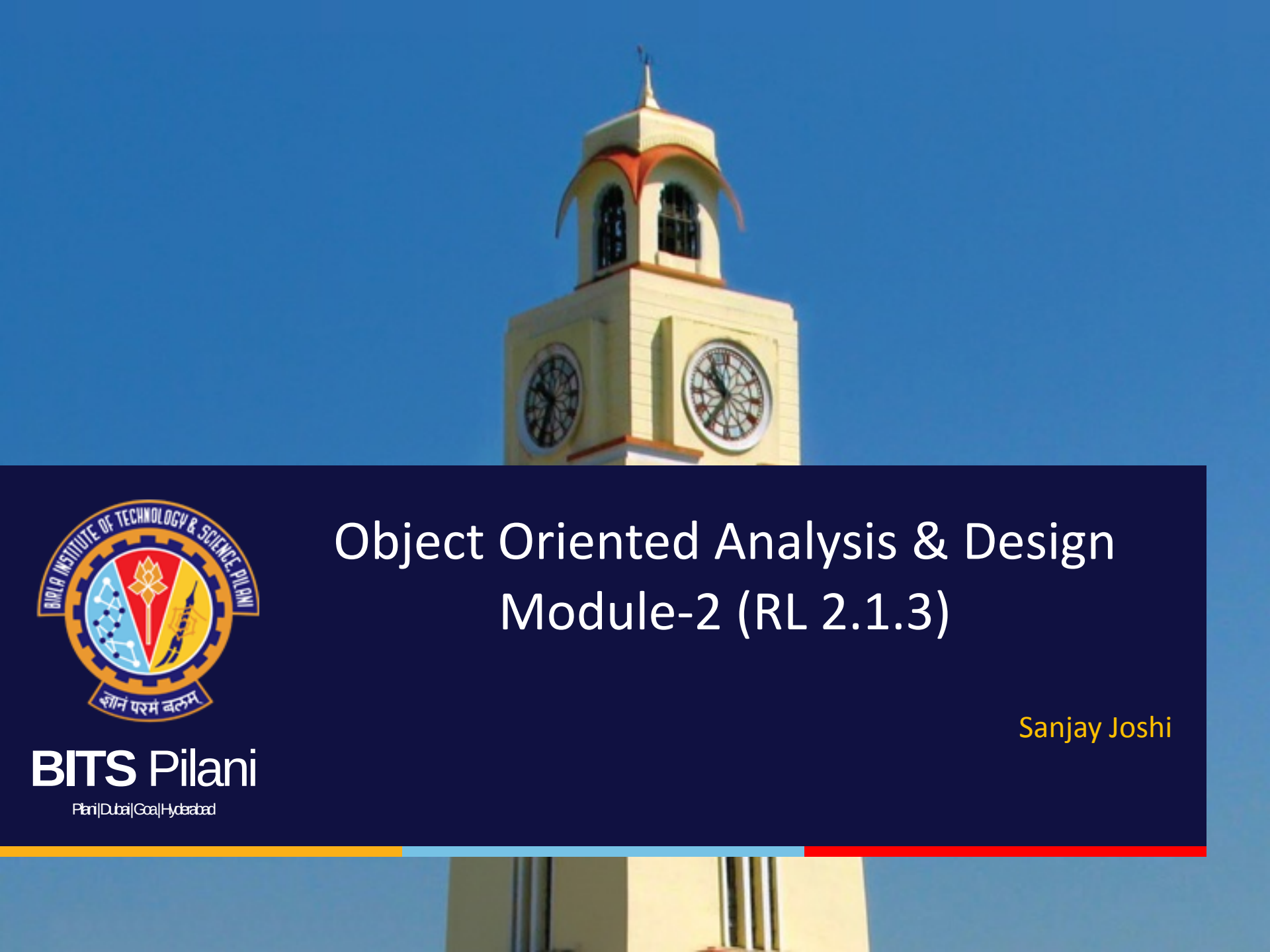
Pilani|Dubai|Goa|Hyderabad

Requirement Categories – Functional & Non Functional Requirements

Requirement Categorization

innovate achieve lead

- Functional requirements
  - Features and capabilities.
  - Recorded in the Use Case model (see next), and in the systems features list of the Vision artifact.
- Non-functional (or quality requirements)
  - Usability (Help, documentation, …), Reliability (Frequency of failure, recoverability, …), Performance (Response times, availability, …), Supportability (Adaptability, maintainability, …)
  - Recorded in the Use Case model or in the Supplementary Specifications artifact.



Object Oriented Analysis & Design
Module-2 (RL 2.1.3)

Sanjay Joshi

BITS Pilani
Pilani|Dubai|Goa|Hyderabad

innovate achieve lead

BITS Pilani

Pilani|Dubai|Goa|Hyderabad

What is Use Case Diagram & Use Cases?

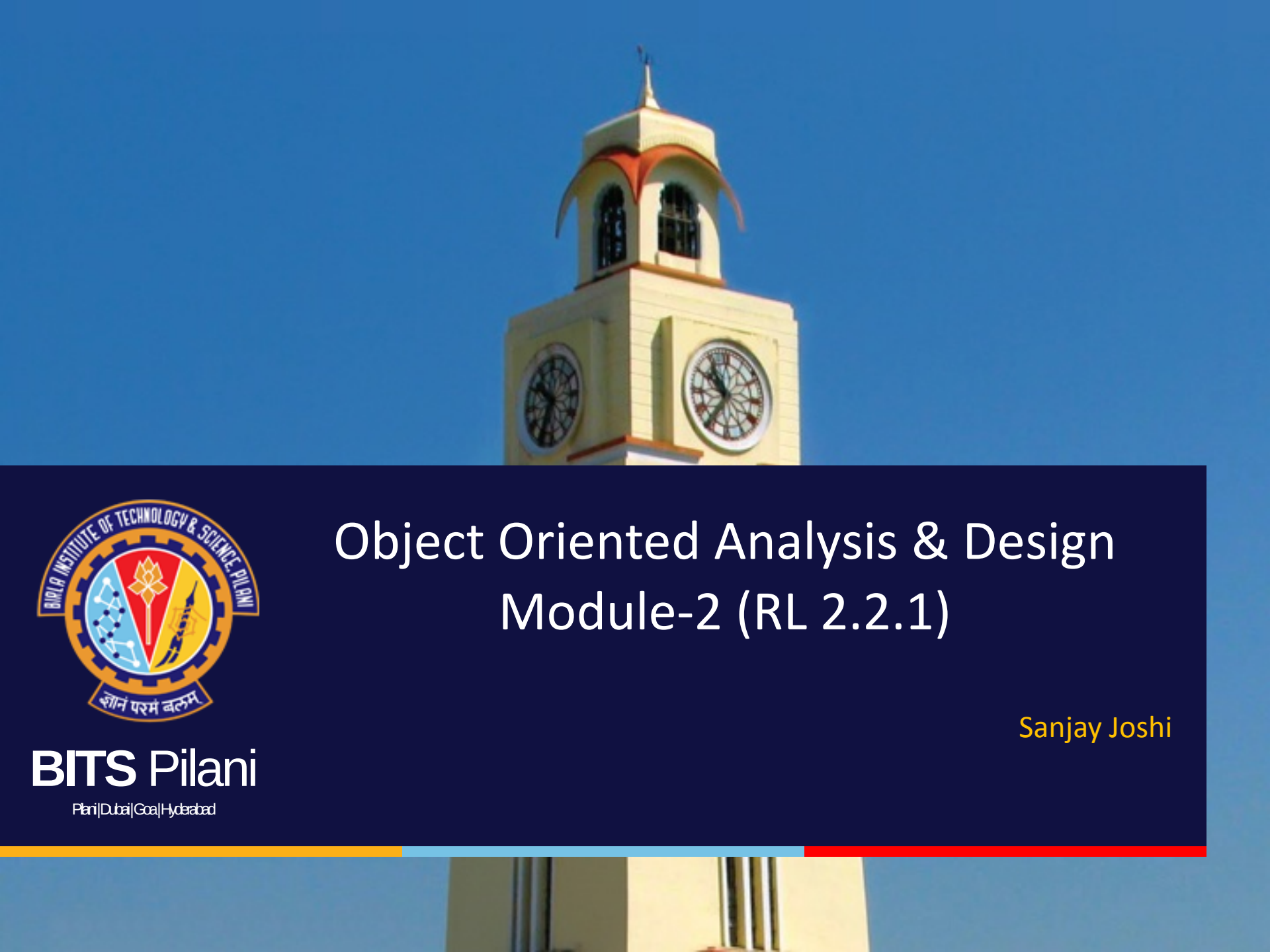
What is Use Case Diagram (UCD)

- It shows interaction among actors (external parties) and Use Cases (Modules) of the System
- It is not Data Flow Diagram !!
- Significance of Use Case Diagram

What is Use Case?

innovate achieve lead

- Use Cases is detailing of the Use Case Diagram (UCD)
- Use Cases are textual artifacts
- Use Case depicts functional requirements primarily.
- Significance of Use Case



Object Oriented Analysis & Design
Module-2 (RL 2.2.1)

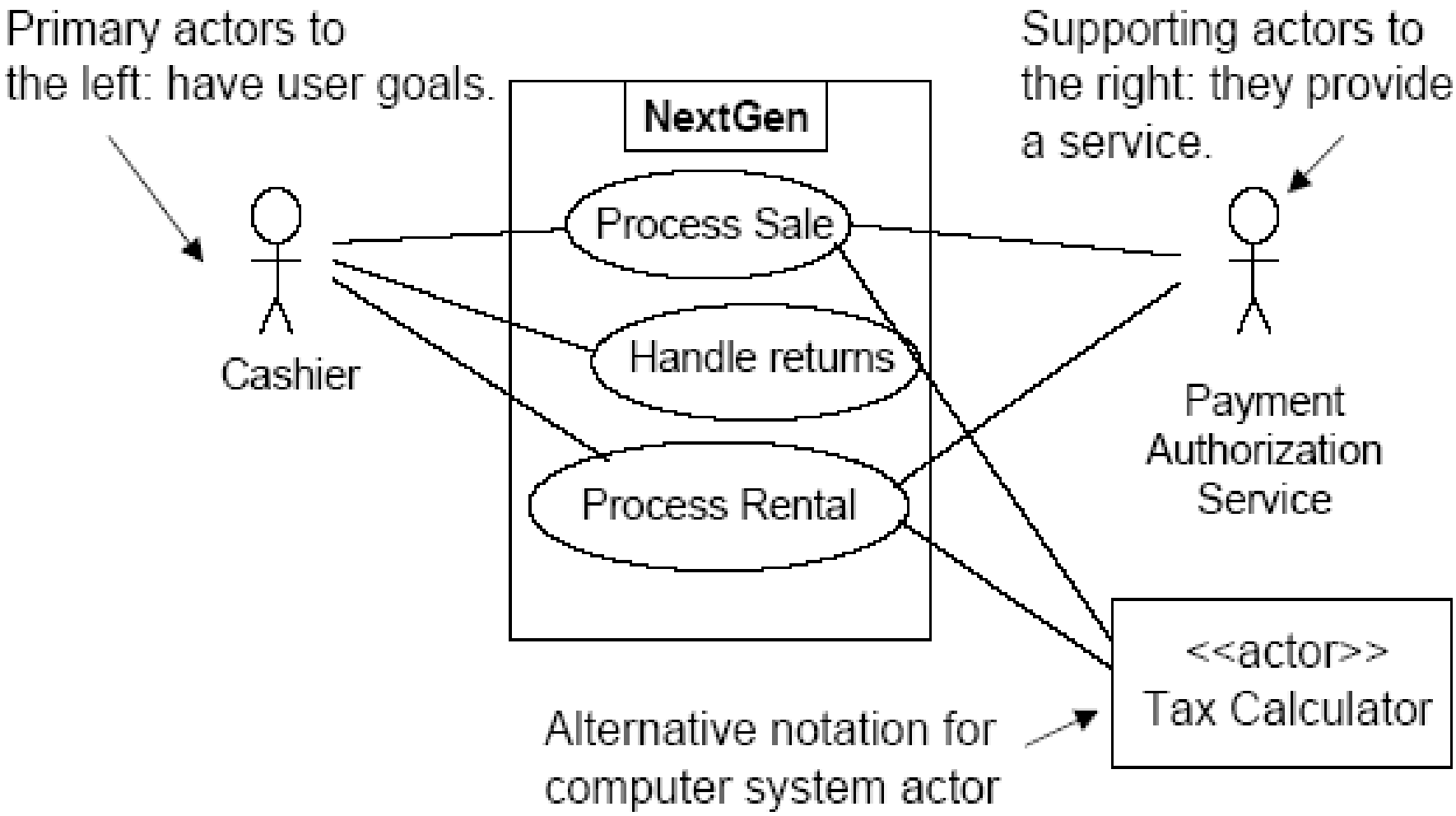
Sanjay Joshi

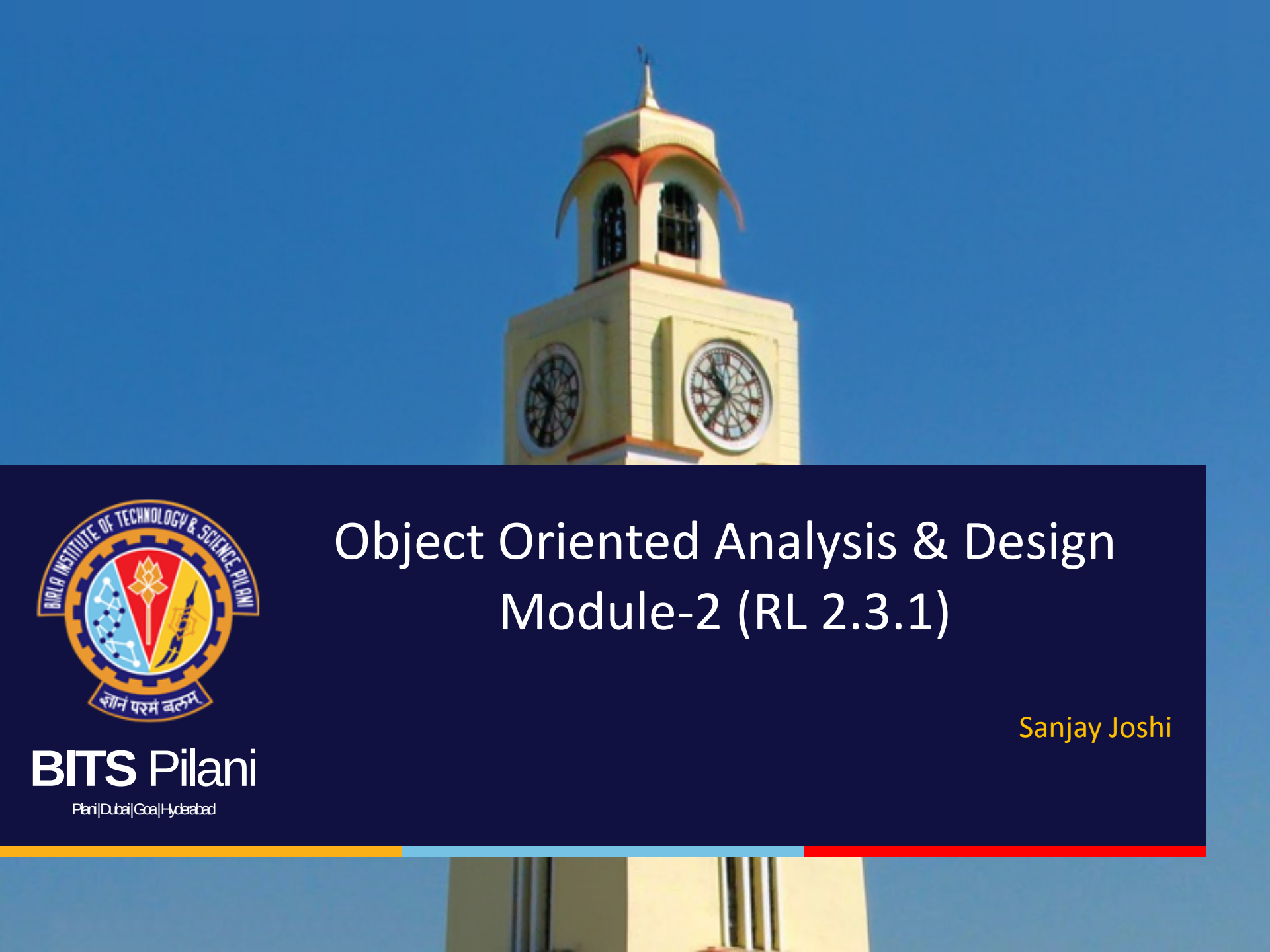
BITS Pilani

Pilani | Dubai | Goa | Hyderabad

Drawing Use Case Diagram for PoS

Use Case Diagram for PoS





Object Oriented Analysis & Design
Module-2 (RL 2.3.1)

Sanjay Joshi

BITS Pilani

Pilani|Dubai|Goa|Hyderabad

innovate achieve lead

BITS Pilani

Pilani|Dubai|Goa|Hyderabad

Use Cases Types

Use case types and formats

innovate achieve lead

- Black-box use cases describe system responsibilities, i.e. define what the system must do.
- Uses cases may be written in three formality types
  - Brief: one-paragraph summary, usually of the main success scenario.
  - Casual: Informal paragraph format (e.g. Handle returns)
  - Fully dressed: elaborate. All steps and variations are written in detail.

Use case types and formats

innovate achieve lead

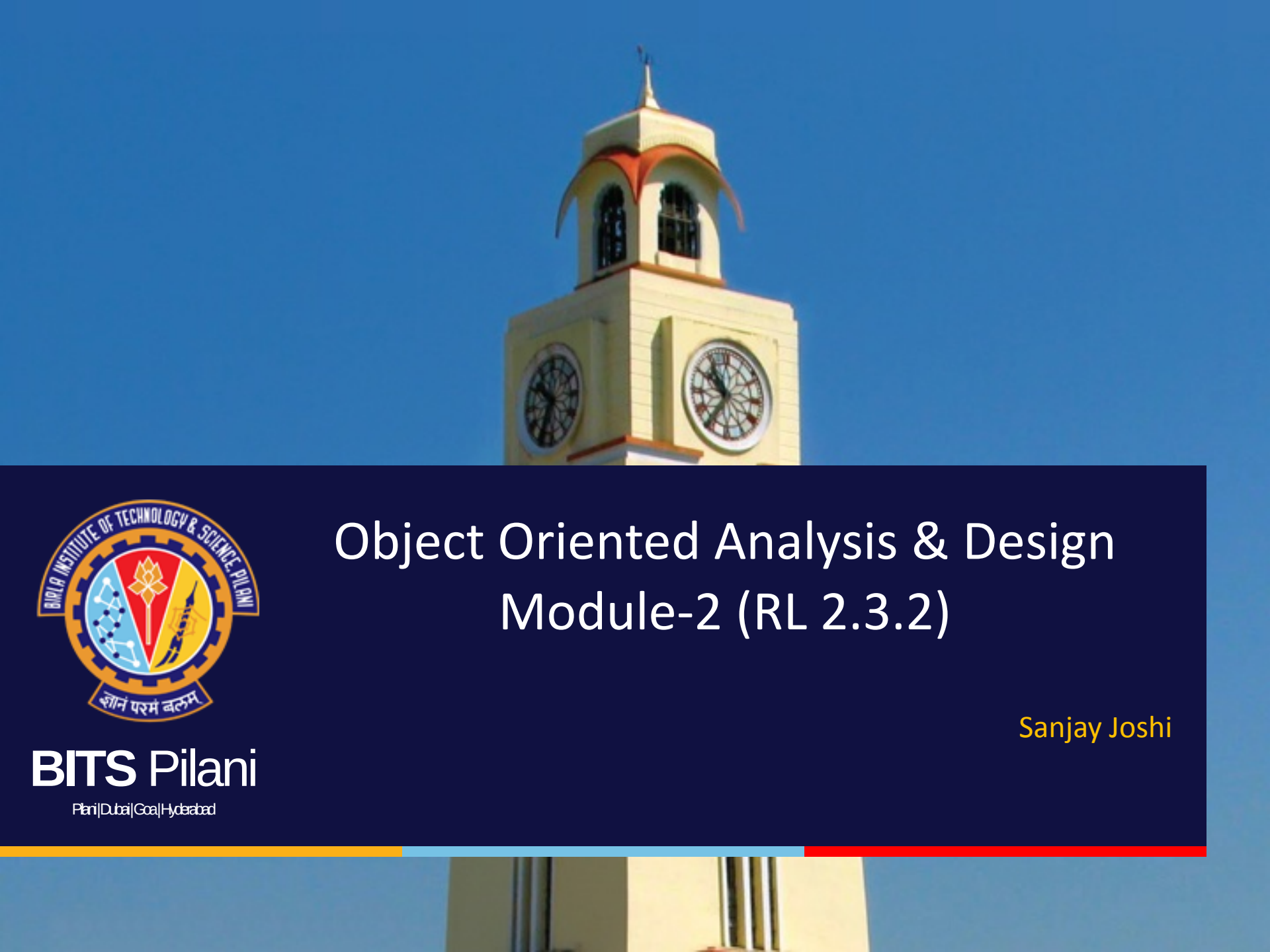
- **Handle returns**

  Main success scenario: A customer arrives at a checkout with items to return. The cashier uses the POS system to record each returned item…

  Alternate scenarios:

  If the credit authorization is reject, inform customer and ask for an alternative payment method.

  If item identifier not found in the system, notify the Cashier and suggest manual entry of the identifier code.



Object Oriented Analysis & Design
Module-2 (RL 2.3.2)

Sanjay Joshi

BITS Pilani
Pilani|Dubai|Goa|Hyderabad

Fully Dressed Use Case for PoS

Fully-dressed example: Process Sale

Use case UC1: Process Sale
Primary Actor: Cashier
Stakeholders and Interests:
-Cashier: Wants accurate and fast entry, no payment errors, …
-Salesperson: Wants sales commissions updated.
…
Preconditions: Cashier is identified and authenticated.
Success Guarantee (Postconditions):
-Sale is saved. Tax correctly calculated.
…
Main success scenario (or basic flow): [see next slide]
Extensions (or alternative flows): [see next slide]
Special requirements: Touch screen UI, …
Open issues: What are the tax law variations? …

Fully dressed example: Process Sale (cont.)

Main success scenario (or basic flow):

1.The Customer arrives at a POS checkout with items to purchase.

2. The cashier records the identifier for each item. If there is more than one of the same item, the Cashier can enter the quantity as well.

3. The system determines the item price and adds the item information to the running sales transaction. The description and the price of the current item are presented.

4. On completion of item entry, the Cashier indicates to the POS system that item entry is complete.

5. The System calculates and presents the sale total.

6.The Cashier tells the customer the total.

7.The Customer gives a cash payment ("cash tendered") possibly greater than the sale total.

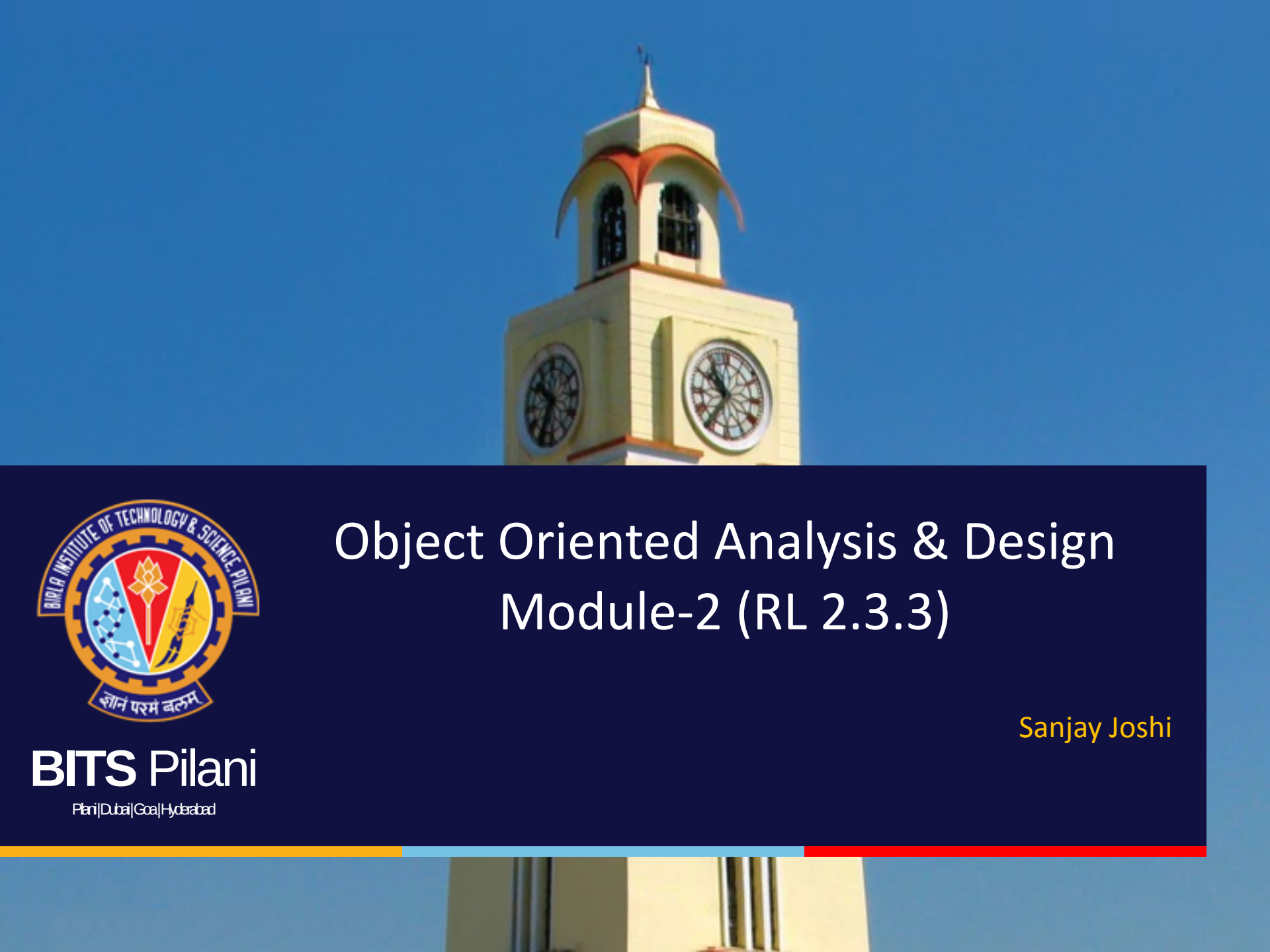
Extensions (or alternative flows):

2a. If sticker is tampered. Enter item id mannually

If invalid identifier entered. Indicate error.

If customer didn't have enough cash, cancel sales transaction.

*If Power failure. Restart the transaction.



Object Oriented Analysis & Design
Module-2 (RL 2.3.3)

Sanjay Joshi

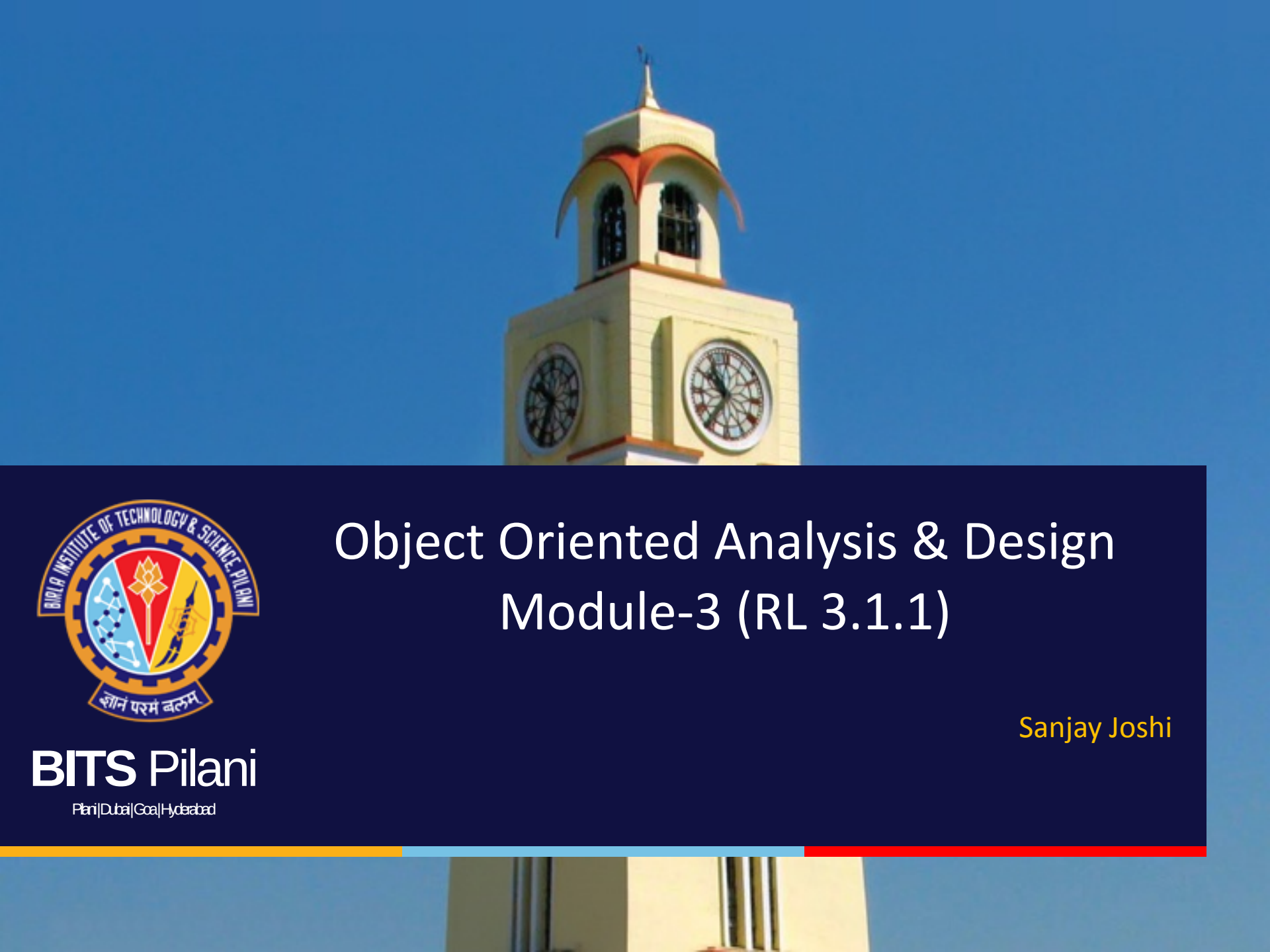
BITS Pilani

Pilani|Dubai|Goa|Hyderabad

Styles of Use Cases

Essential vs. Concrete style

- Essential: Focus is on intend.
  - Avoid making UI decisions
- Concrete: UI decisions are embedded in the use case text.
  - e.g. “Admin enters ID and password in the dialog box (see picture X)”
  - Concrete style not suitable during early requirements analysis work.



Object Oriented Analysis & Design
Module-3 (RL 3.1.1)

Sanjay Joshi

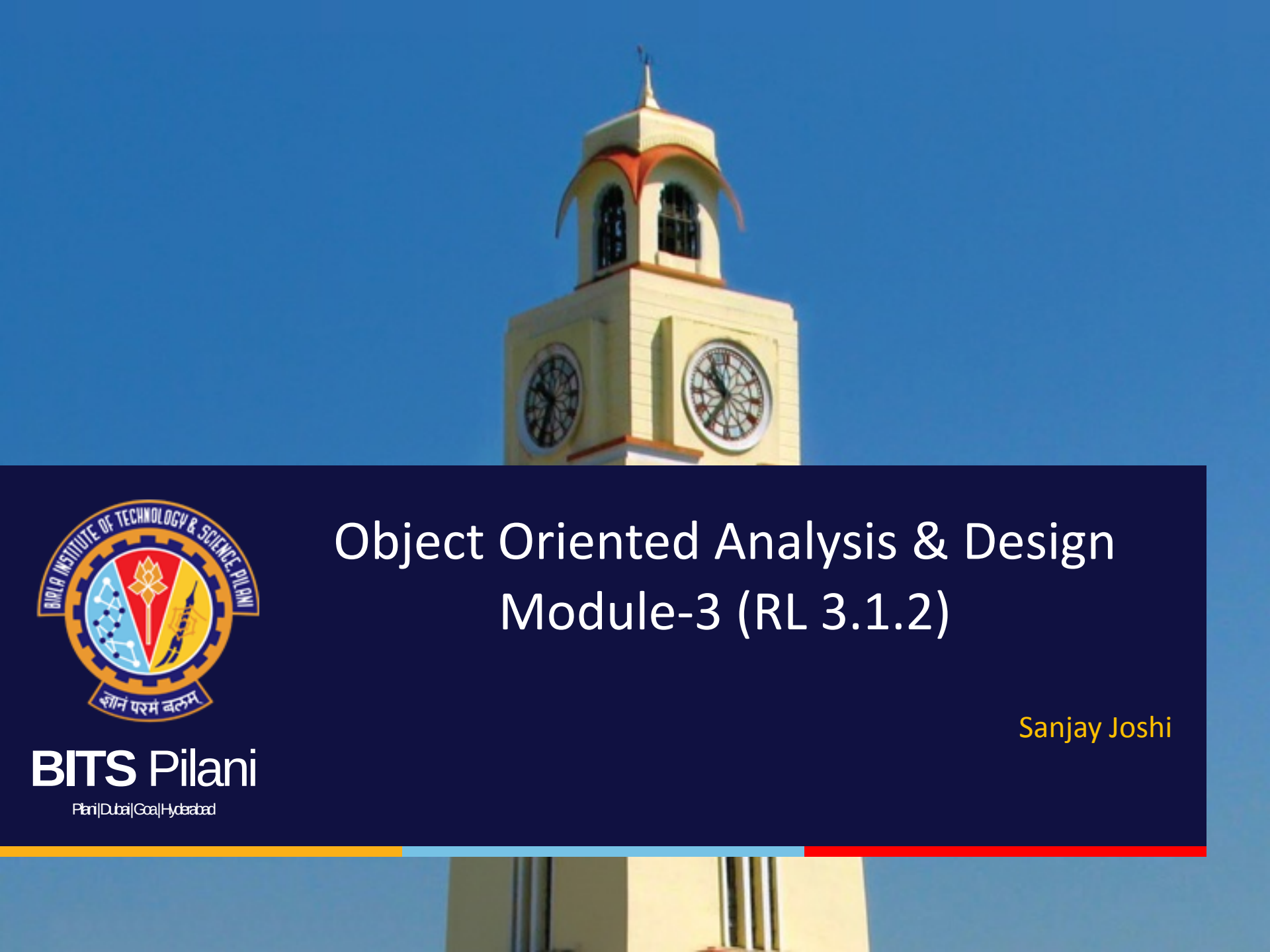
BITS Pilani

Pilani|Dubai|Goa|Hyderabad

What is Domain Model?

Domain Model

- A Domain Model illustrates meaningful concepts in a problem domain.
- It is a representation of real-world things, not software components.
- It is a set of static structure diagrams; no operations are defined.
- It may show:
  - concepts
  - associations between concepts
  - attributes of concepts



Object Oriented Analysis & Design
Module-3 (RL 3.1.2)

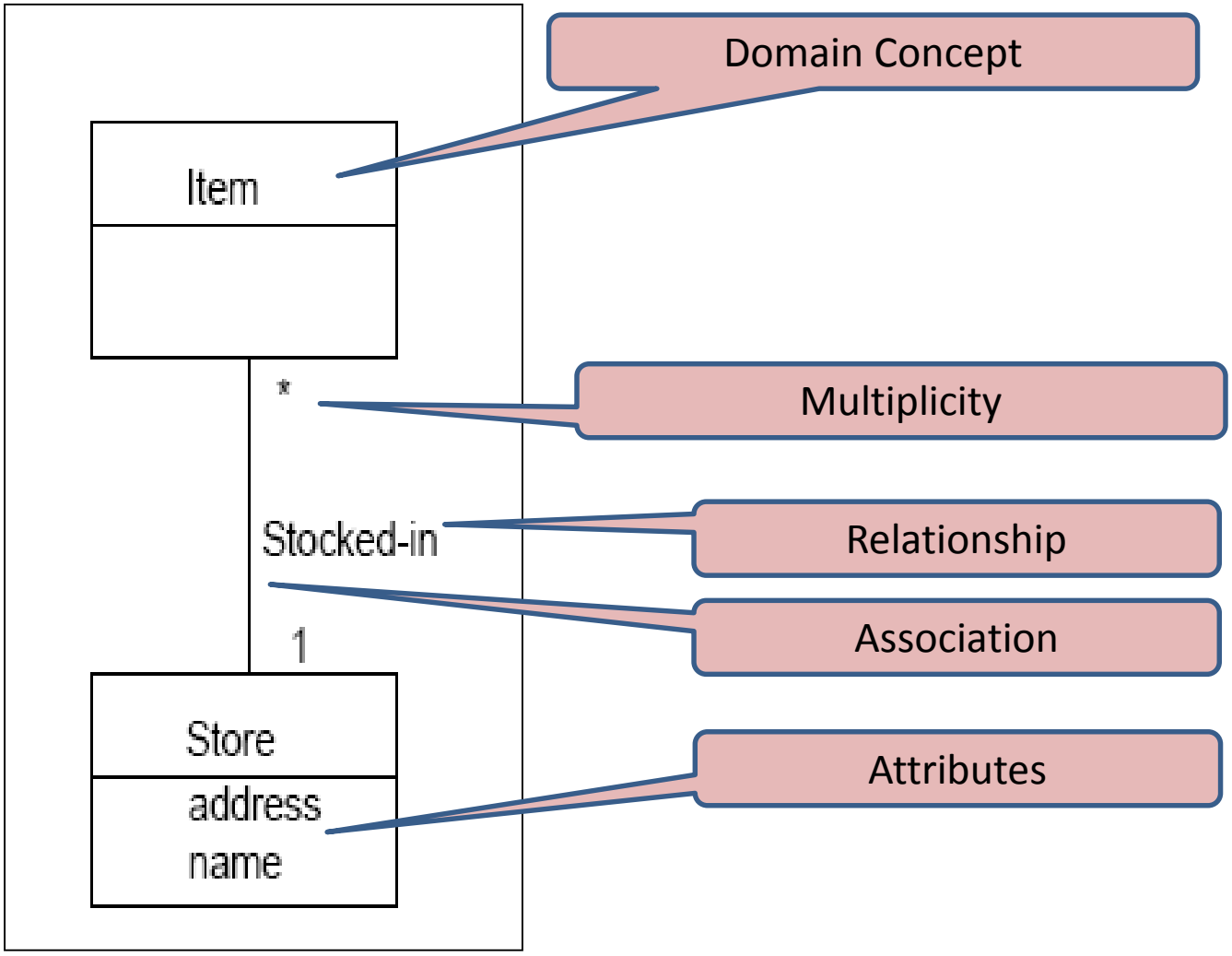
Sanjay Joshi

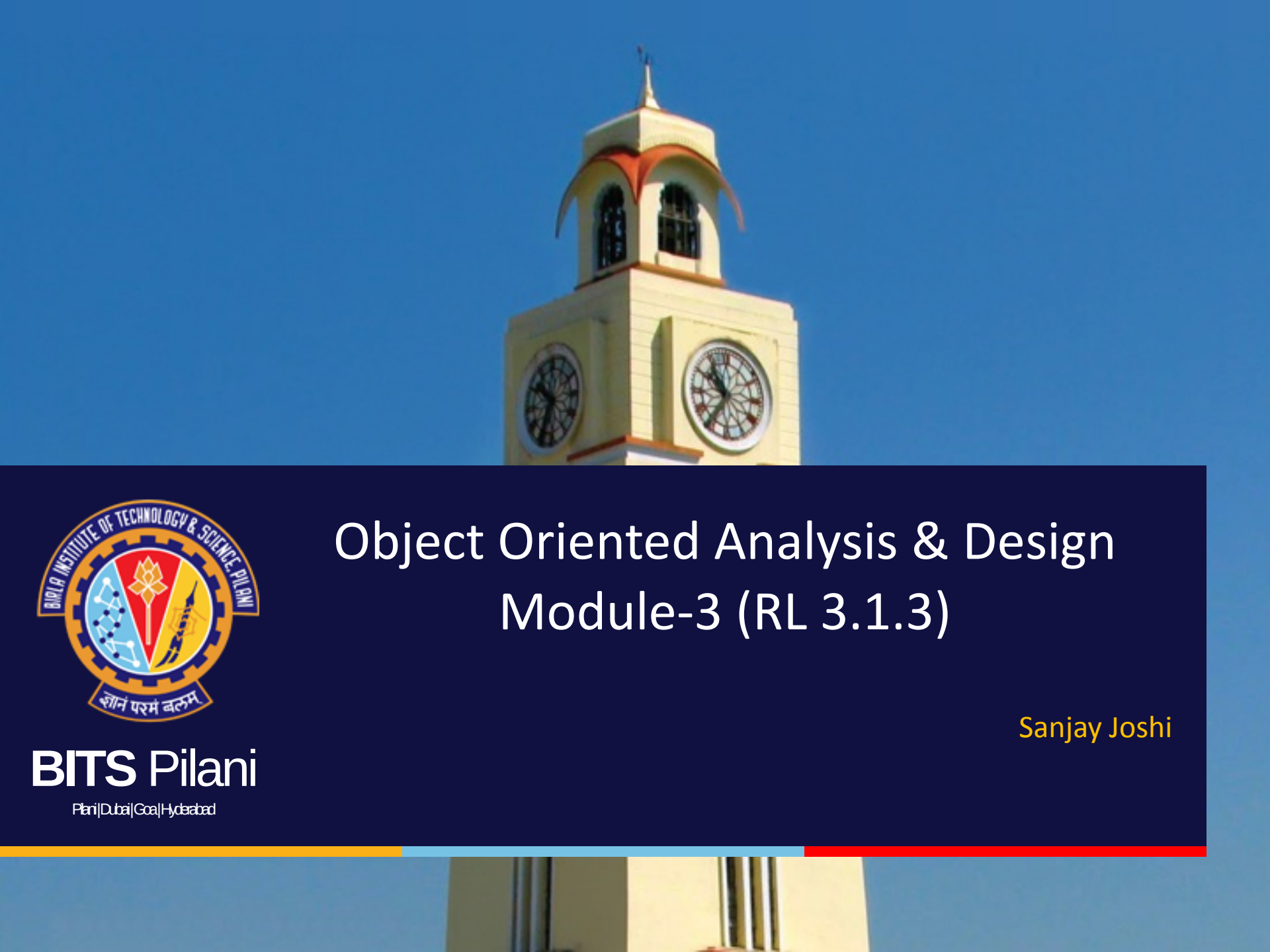
BITS Pilani

Pilani|Dubai|Goa|Hyderabad

How Domain Model is represented in UML?

Domain Model in UML





Object Oriented Analysis & Design
Module-3 (RL 3.1.3)

Sanjay Joshi

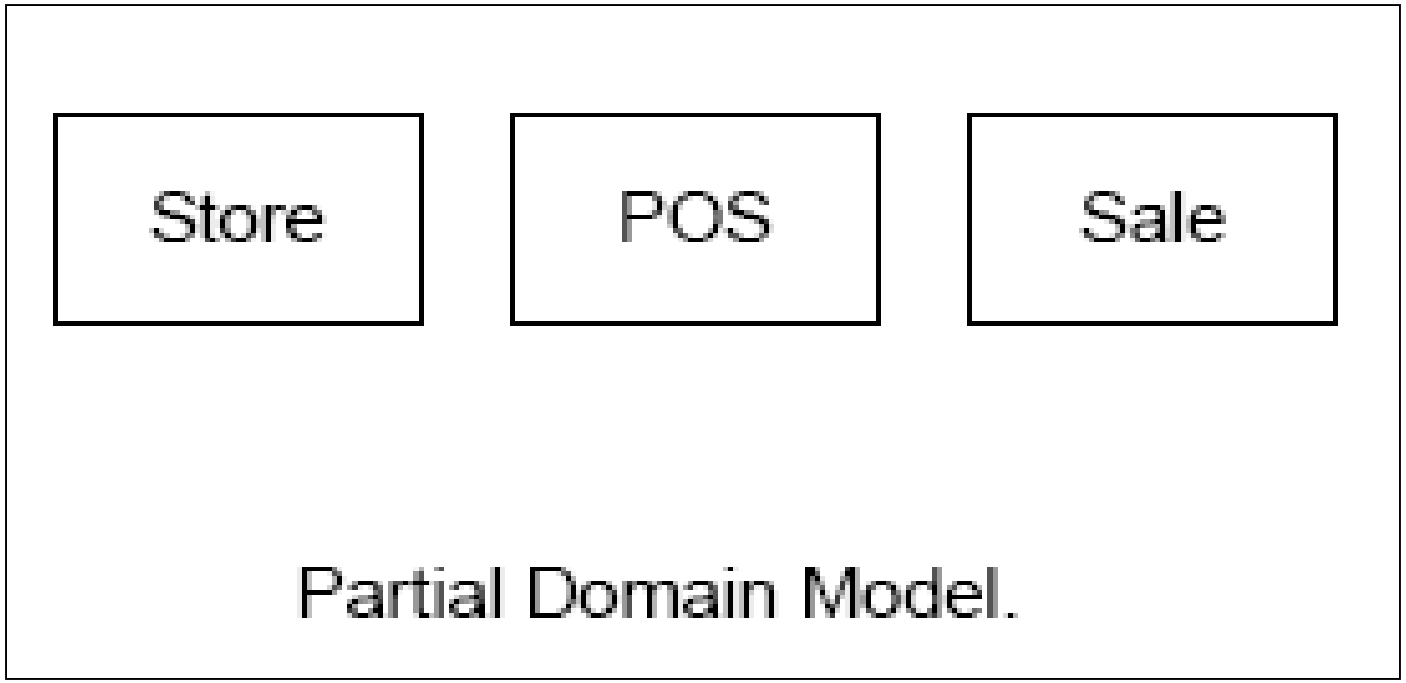
BITS Pilani

Pilani|Dubai|Goa|Hyderabad

Identification of Domain Concepts from Use Case

Domains Concepts in PoS

- A central distinction between Object oriented and Procedure Oriented: division by concepts (objects) rather than division by functions.



Strategy to Identify Conceptual Classes

- Use noun phrase identification.
  - Identify noun (and noun phrases) in textual descriptions of the problem domain, and consider them as concepts or attributes.
  - Use Cases are excellent description to draw for this analysis.

Finding Conceptual Classes with Noun Phrase Identification

1. This use case begins when a **Customer** arrives at a **POS checkout** with items to purchase.
2. The **Cashier** starts a new sale.
3. **Cashier** enters **item identifier**.

...

- The fully addressed Use Cases are an excellent description to draw for this analysis.
- Some of these noun phrases are candidate concepts; some may be attributes of concepts.
- A mechanical noun-to-concept mapping is not possible, as words in a natural language are (sometimes) ambiguous.

Fully-dressed Use Case: Process Sale

innovate achieve lead

Use case UC1: Process Sale
Primary Actor: Cashier
Stakeholders and Interests:
-Cashier: Wants accurate and fast entry, no payment errors, …
-Salesperson: Wants sales commissions updated.
…
Preconditions: Cashier is identified and authenticated.
Success Guarantee (Postconditions):
-Sale is saved. Tax correctly calculated.
…
Main success scenario (or basic flow): [see next slide]
Extensions (or alternative flows): [see next slide]
Special requirements: Touch screen UI, …
Open issues: What are the tax law variations? …

Fully dressed example: Process Sale

innovate achieve lead

Main success scenario (or basic flow):

1.The Customer arrives at a POS checkout with items to purchase.
2. The cashier records the identifier for each item. If there is more than one of the same item, the Cashier can enter the quantity as well.
3. The system determines the item price and adds the item information to the running sales transaction. The description and the price of the current item are presented.
4. On completion of item entry, the Cashier indicates to the POS system that item entry is complete.
5. The System calculates and presents the sale total.
6.The Cashier tells the customer the total.
7.The Customer gives a cash payment ("cash tendered") possibly greater than the sale total.

Extensions (or alternative flows):

2a. If sticker is tampered. Enter item id mannually
If invalid identifier entered. Indicate error.
If customer didn't have enough cash, cancel sales transaction.
*If Power failure. Restart the transaction.

The NextGen POS (partial) Domain Model

Object Oriented Analysis & Design
Module-3 (RL 3.1.4)

Sanjay Joshi

BITS Pilani

Pilani|Dubai|Goa|Hyderabad

innovate achieve lead

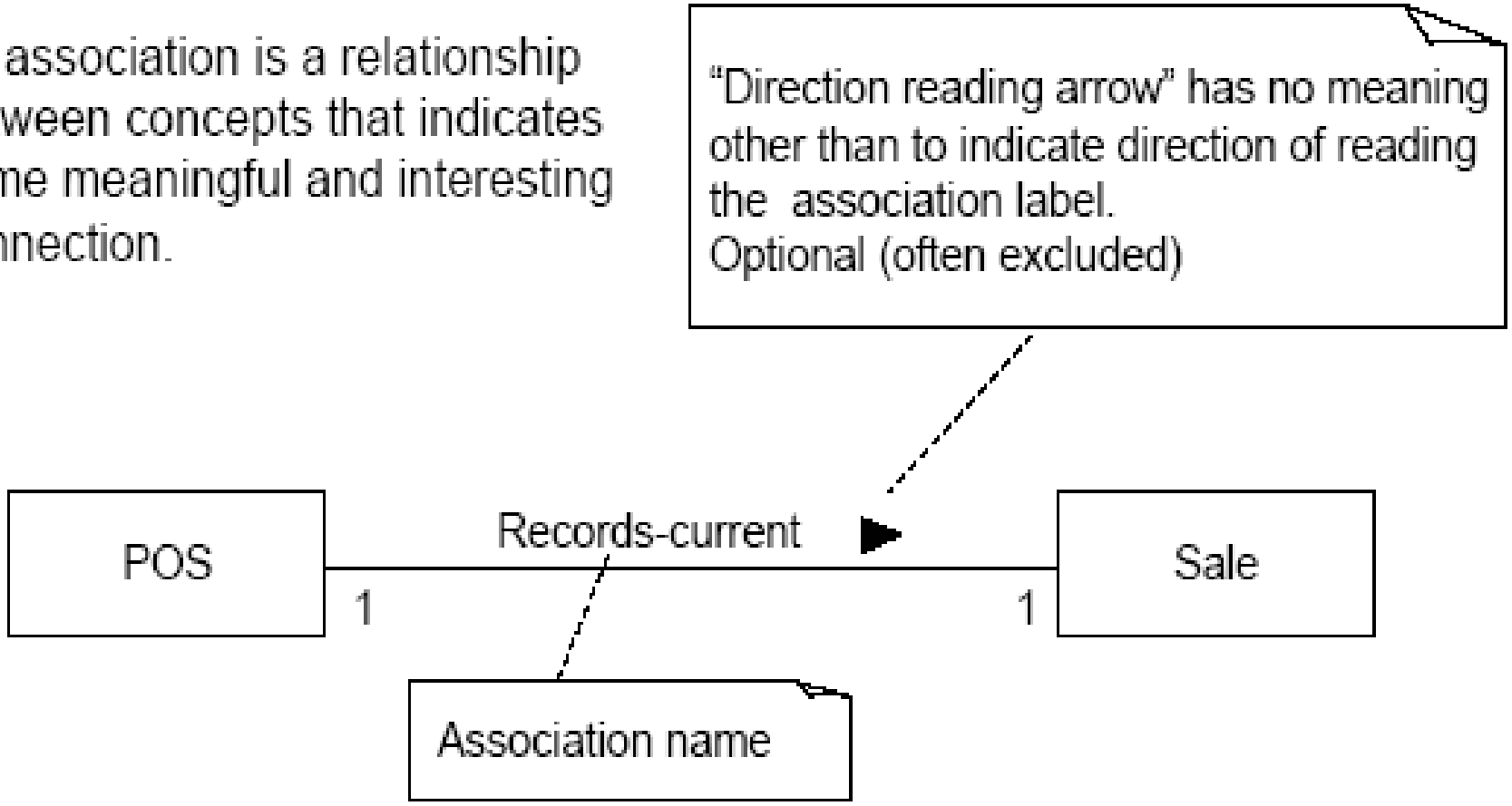
BITS Pilani

Pilani|Dubai|Goa|Hyderabad

Identification of relationship among domain concepts

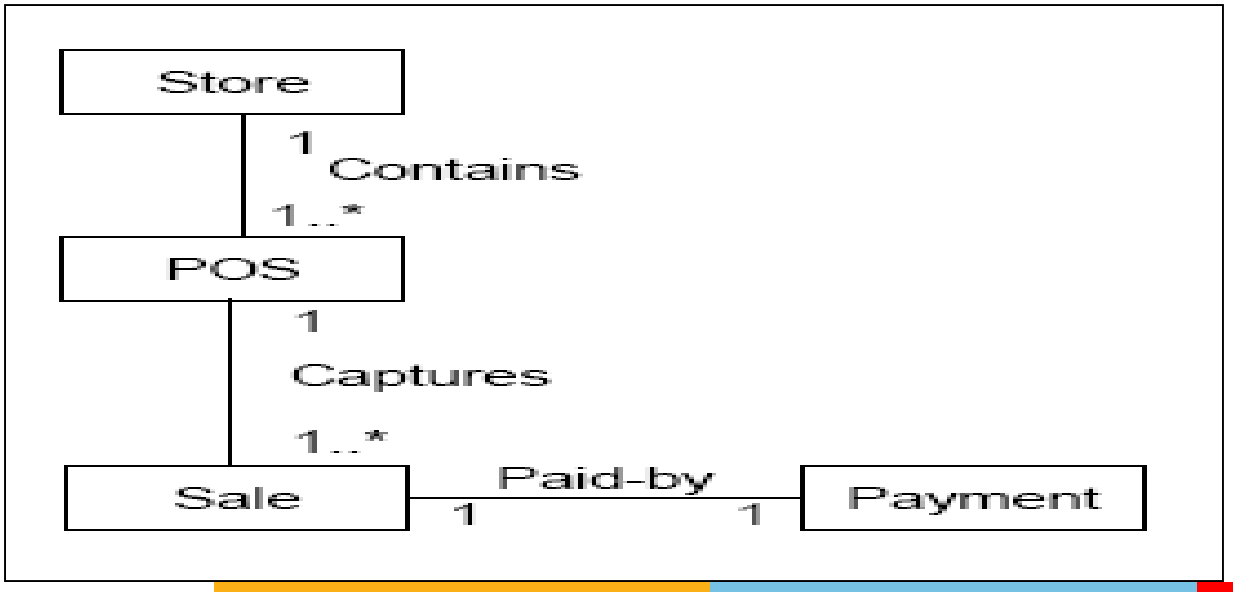
Adding Associations

An association is a relationship between concepts that indicates some meaningful and interesting connection.



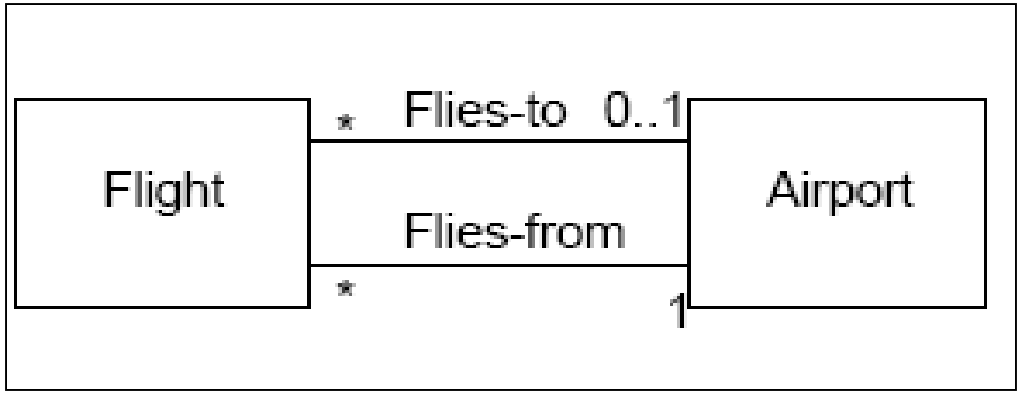
Naming Associations

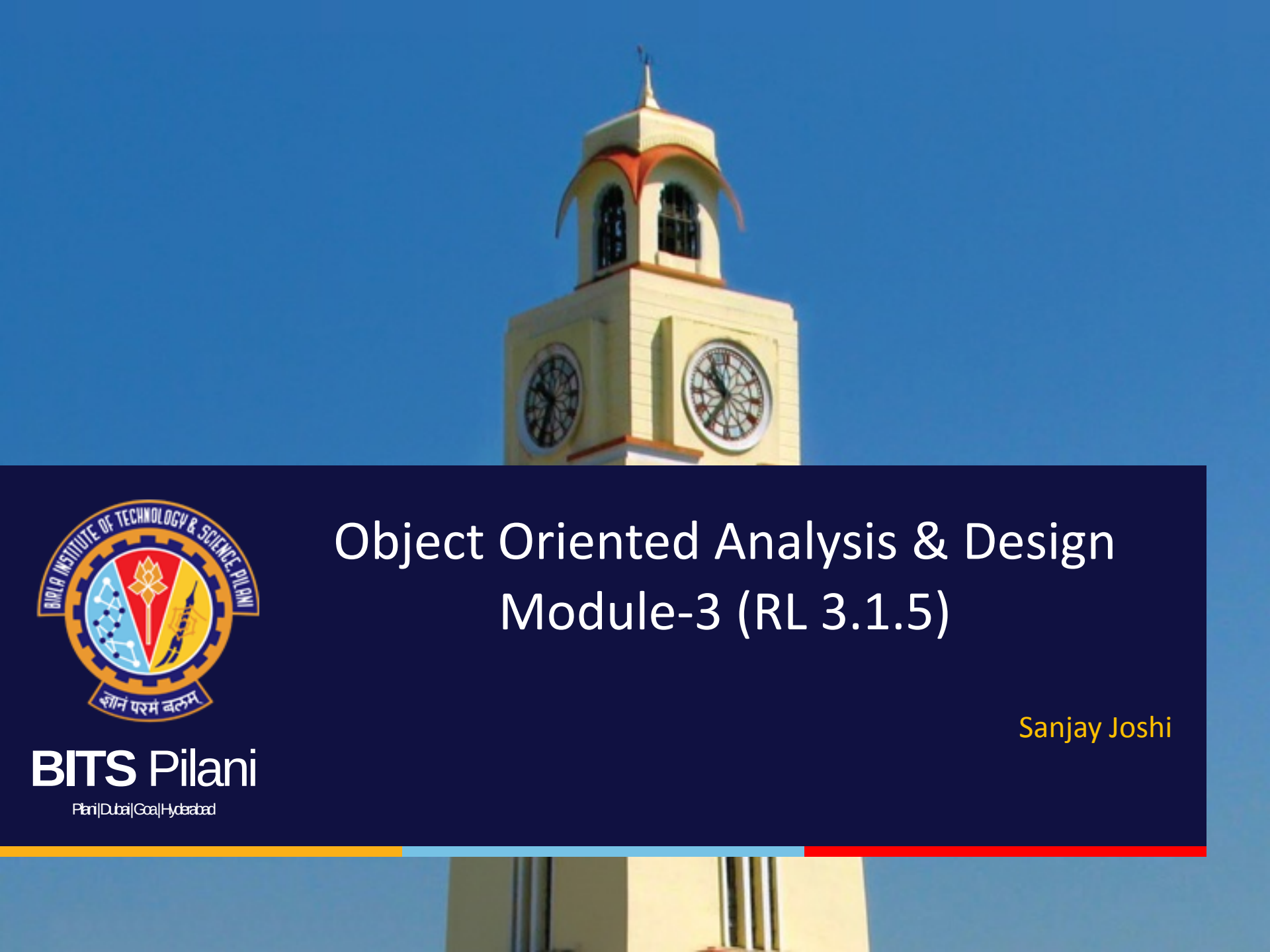
- Name an association based on a TypeName-VerbPhrase-TypeName format.
- Association names should start with a capital letter.
- A verb phrase should be constructed with hyphens.
- The default direction to read an association name is left to right, or top to bottom.



Multiple Associations Between Two Types

- It is not uncommon to have multiple associations between two types.
- In the example, not every flight is guaranteed to land at an airport.





Object Oriented Analysis & Design
Module-3 (RL 3.1.5)

Sanjay Joshi

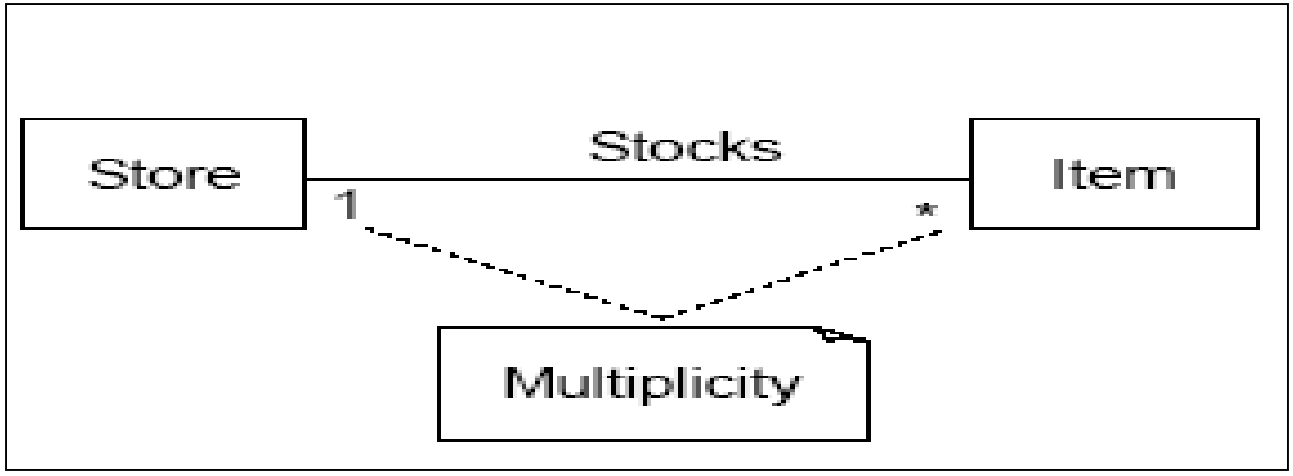
BITS Pilani

Pilani|Dubai|Goa|Hyderabad

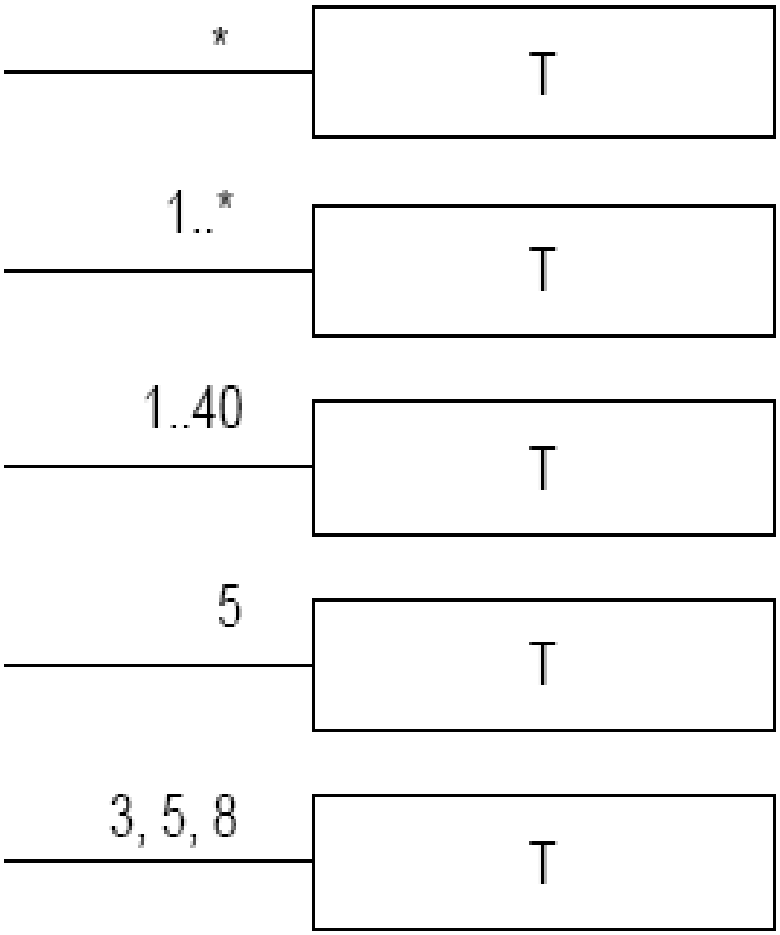
Finding multiplicity among Domain Concepts

Multiplicity

- Multiplicity defines how many instances of a type A can be associated with one instance of a type B, at a particular moment in time.
- For example, a single instance of a Store can be associated with “many” (zero or more) Item instances.



Multiplicity



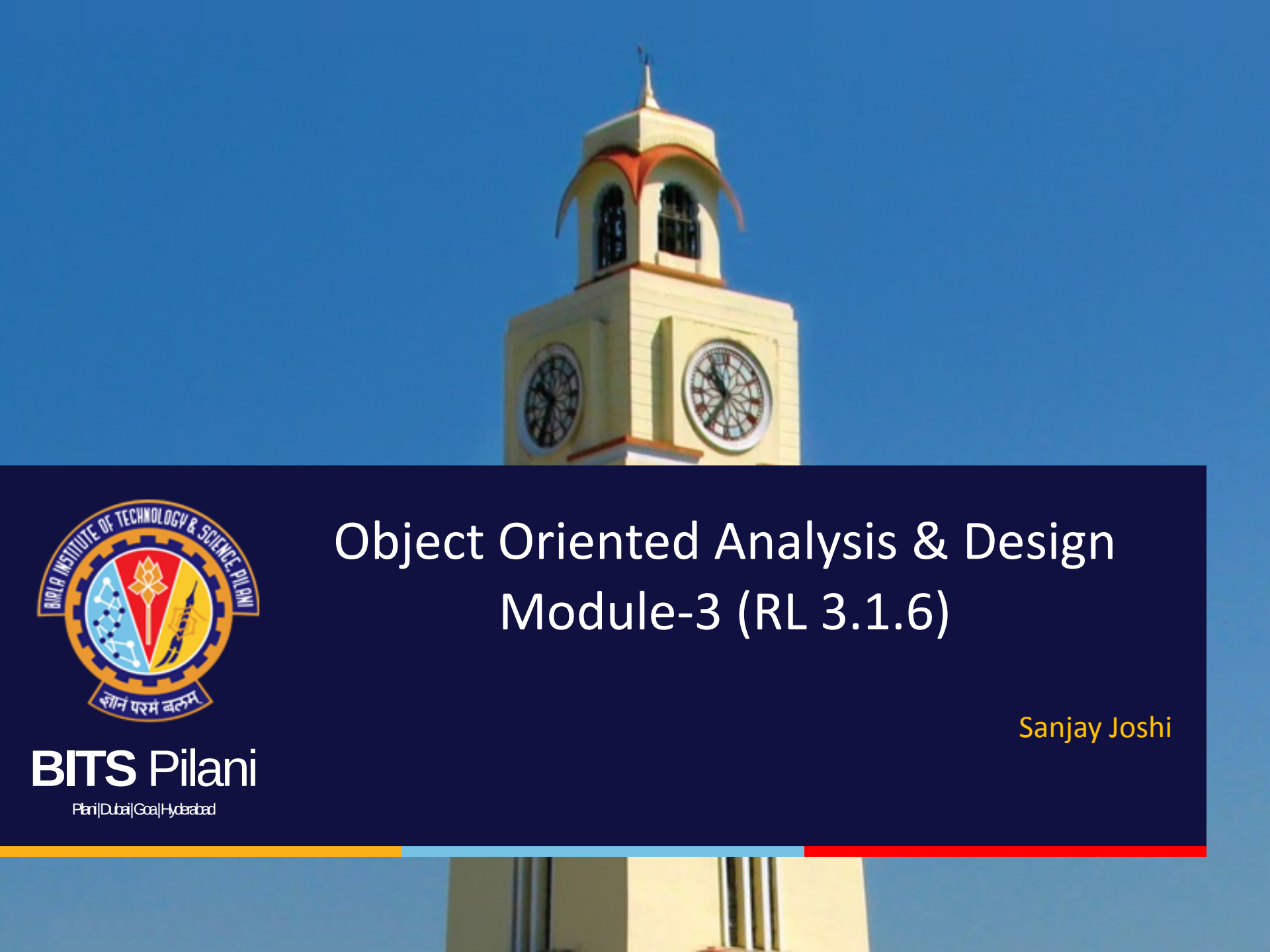
Zero or more; "many"

One or more

One to forty

Exactly five

Exactly three, five or eight.



Object Oriented Analysis & Design
Module-3 (RL 3.1.6)

Sanjay Joshi

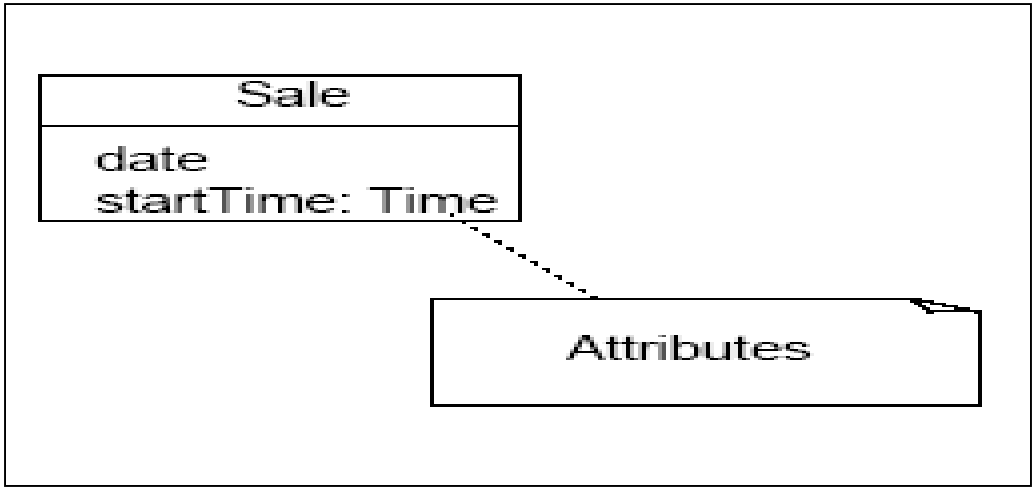
BITS Pilani

Pilani|Dubai|Goa|Hyderabad

Adding attributes to Domain Model

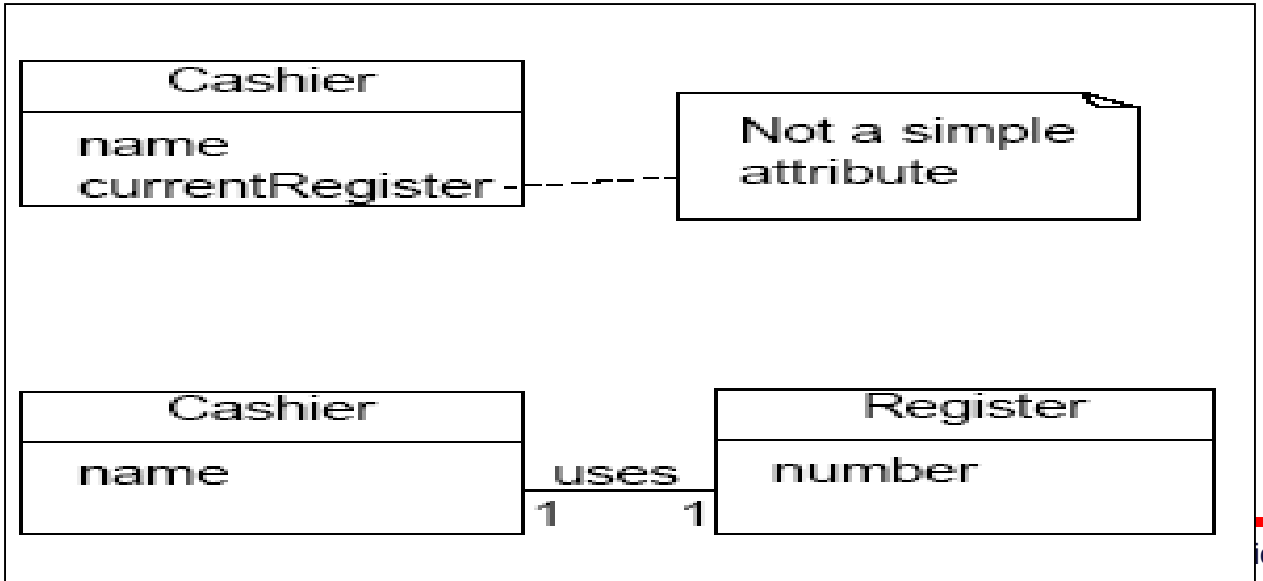
Adding Attributes

- An attribute is a logical data value of an object.
- Include the following attributes: those for which the requirements suggest or imply a need to remember information.
- For example, a Sales receipt normally includes a date and time.
- The Sale concept would need a date and time attribute.



Valid Attribute Types

- Keep attributes simple.
- The type of an attribute should not normally be a complex domain concept, such as Sale or Airport.
- Attributes in a Domain Model should preferably be
  - Pure data values: Boolean, Date, Number, String, …
  - Simple attributes: color, phone number, zip code, universal product code (UPC), ...



Object Oriented Analysis & Design
Module-3 (RL 3.1.7)

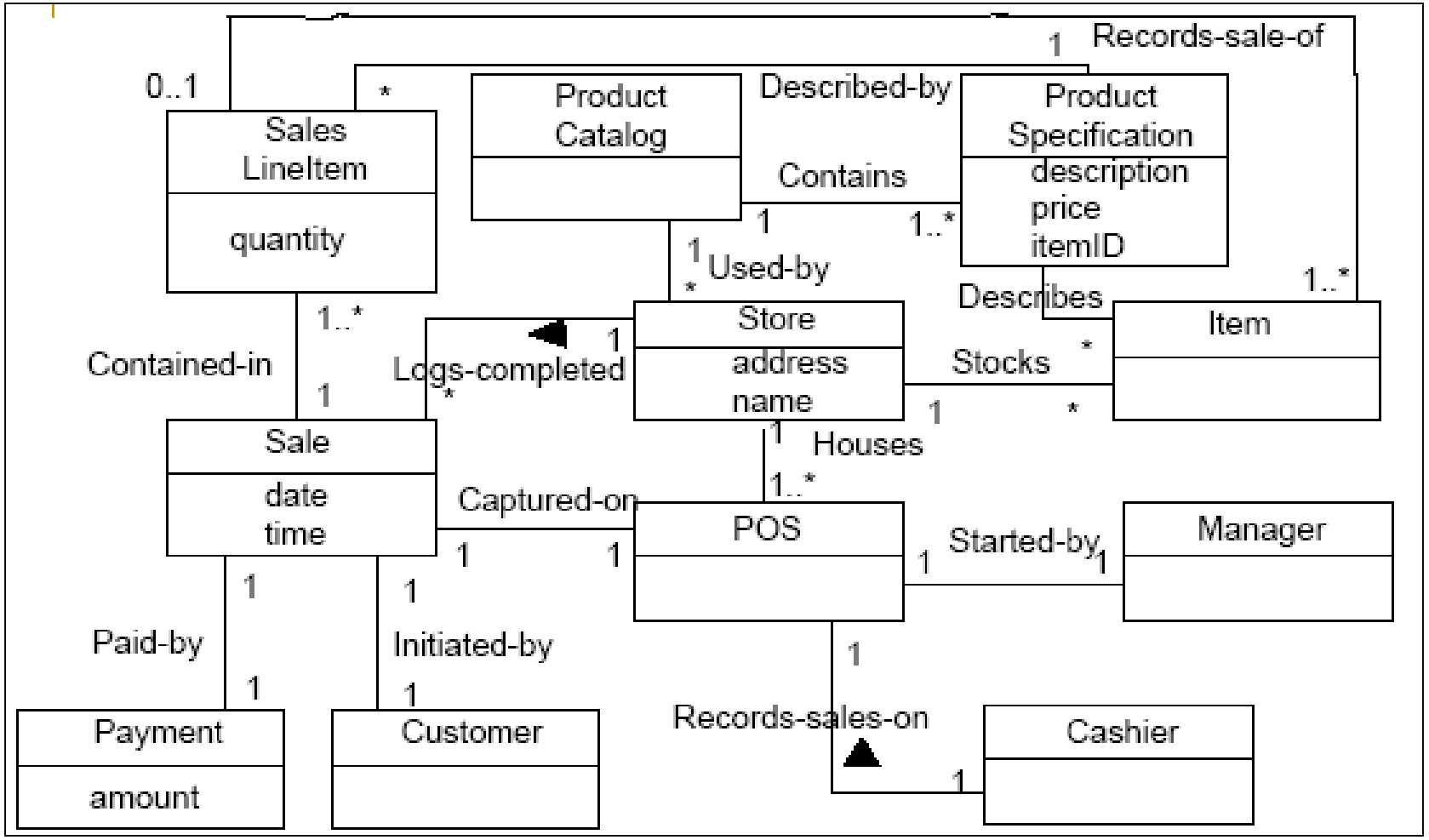
Sanjay Joshi

BITS Pilani

Pilani | Dubai | Goa | Hyderabad

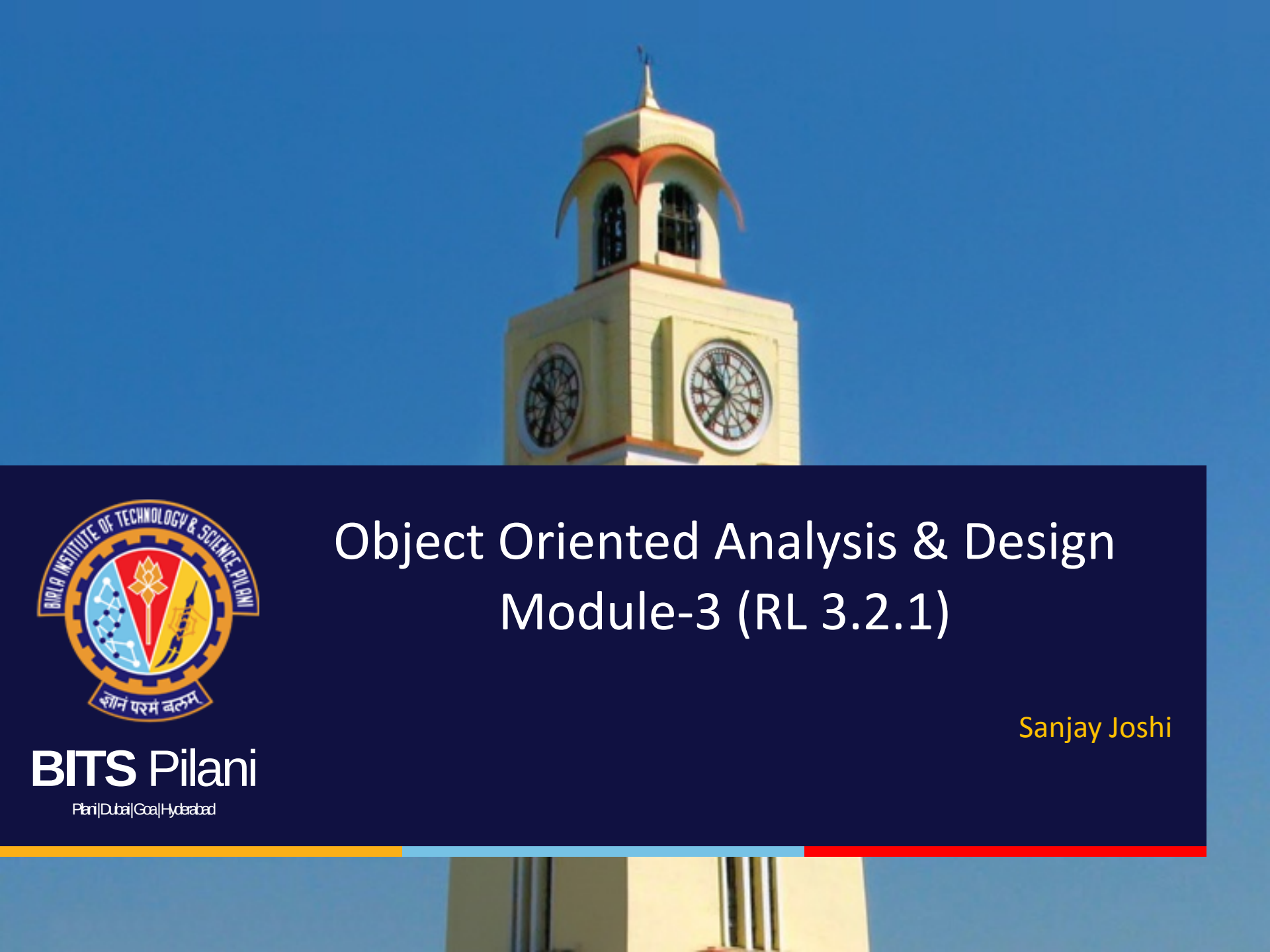
Significance of Domain Model

Domain Model : Partial?



Significance of Domain Model

- Domain Model is base for Designer to draw Class Diagram
- Not necessary that all Domain Concepts will be carried forward.
- More implementation classes can be added by Designed in Class Diagram



Object Oriented Analysis & Design
Module-3 (RL 3.2.1)

Sanjay Joshi

BITS Pilani
Pilani|Dubai|Goa|Hyderabad

What is SSD (System Sequence Diagram)

What is SSD (System Sequence Diagram) ?

- It is useful to investigate and define the behavior of the software as a “black box”.
- System behavior is a description of *what the system does* (without an explanation of how it does it).
- Use cases describe how external actors interact with the software system. During this interaction, an actor generates events.
- A request event initiates an operation upon the system.

Object Oriented Analysis & Design
Module-3 (RL 3.2.2)

Sanjay Joshi

BITS Pilani

Pilani|Dubai|Goa|Hyderabad

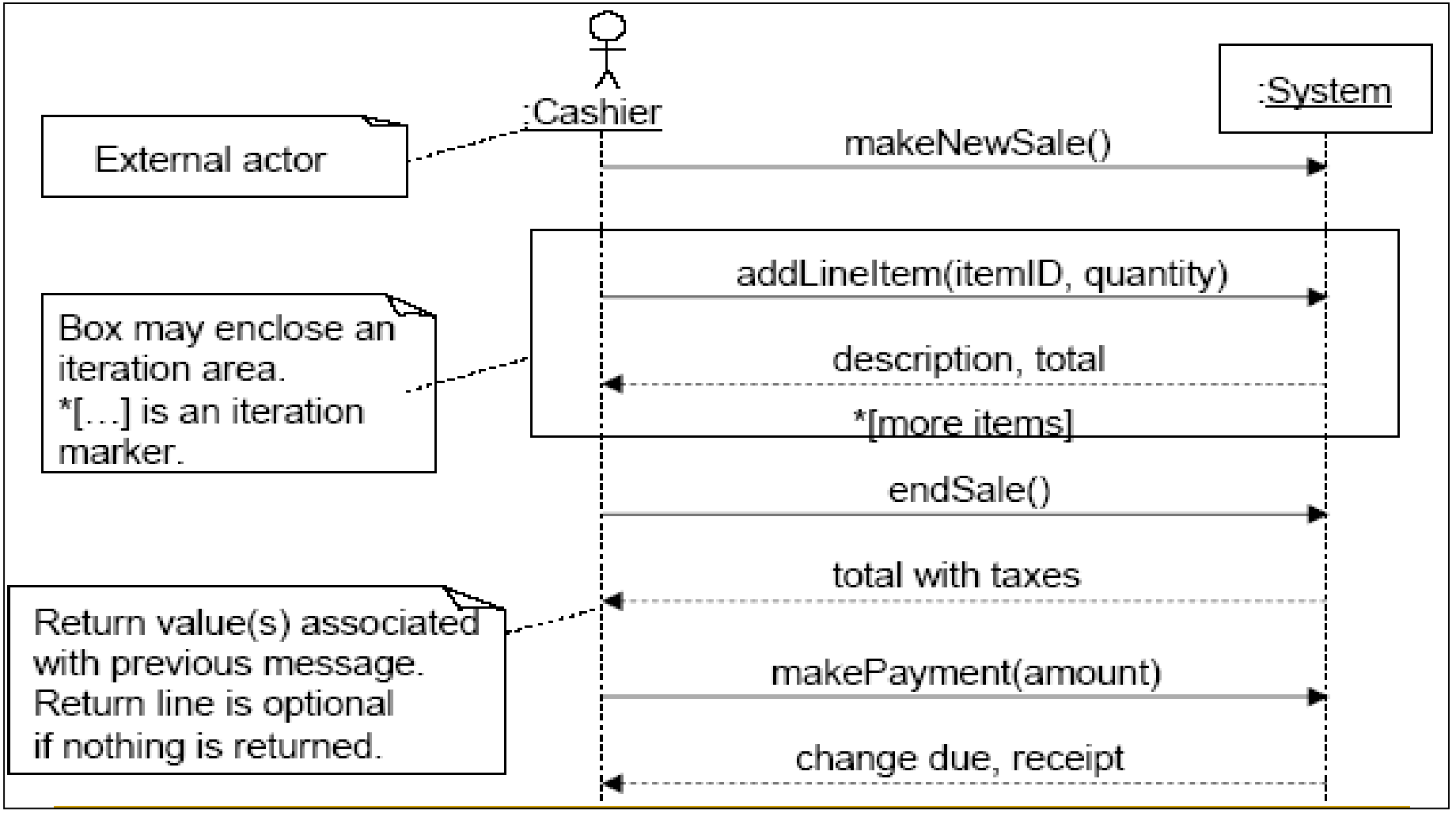
innovate achieve lead

BITS Pilani

Pilani|Dubai|Goa|Hyderabad

Drawing SSD for PoS

SSDs for Process Sale Scenario

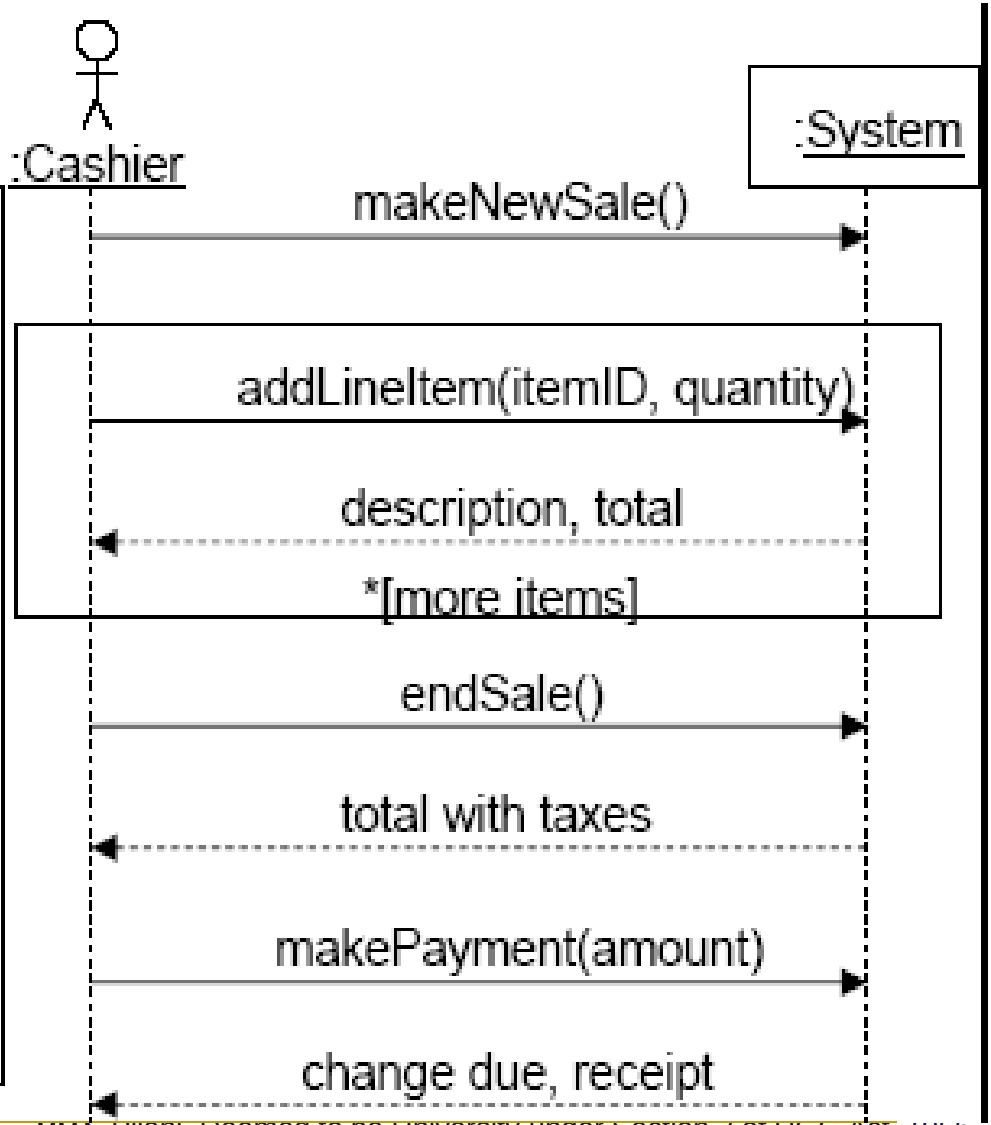


SSD and Use Cases

Simple cash-only Process Sale Scenario

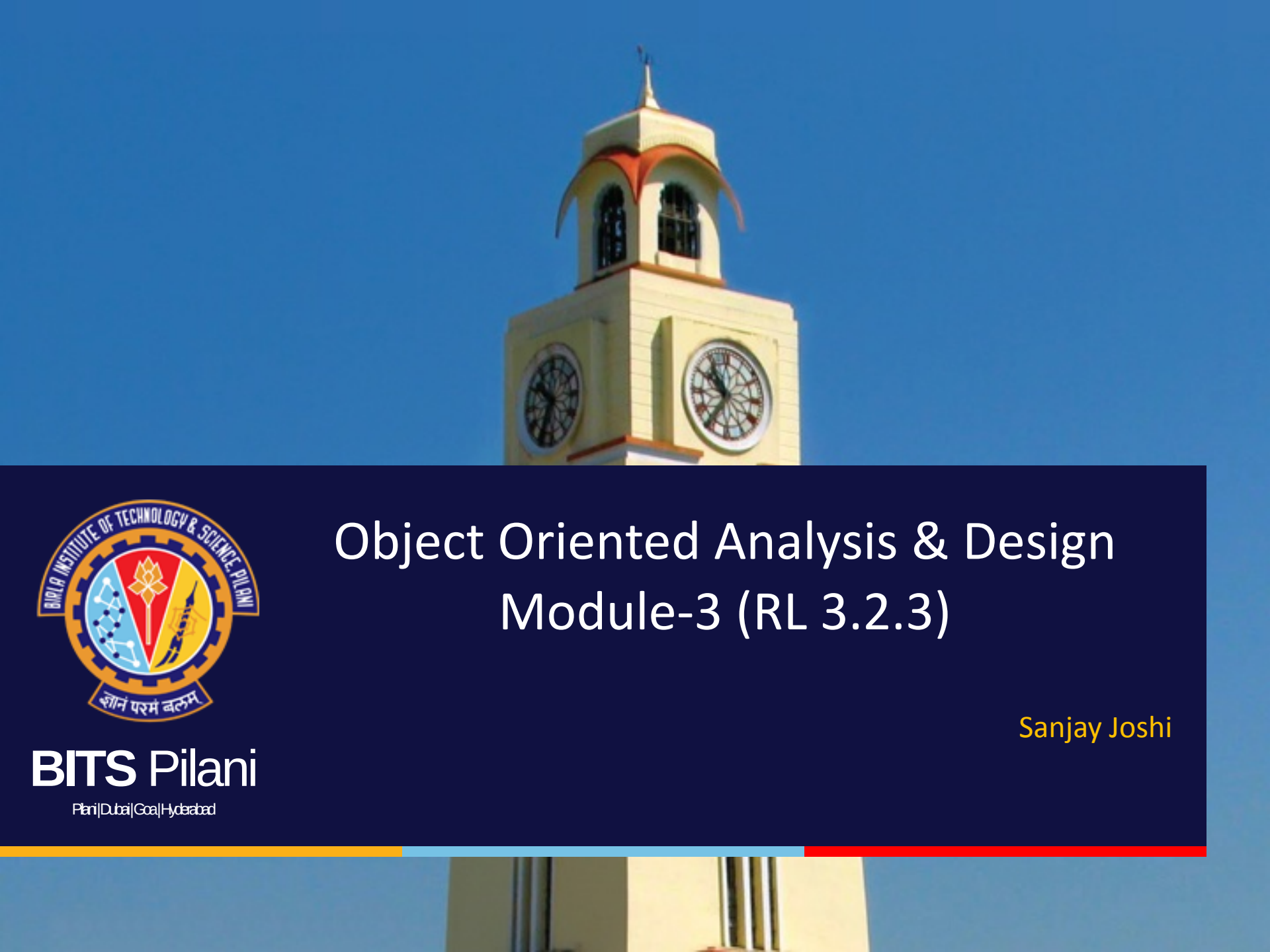
1. Customer arrives at a POS checkout with goods to purchase.
2. Cashier starts a new sale.
3. Cashier enters item identifier.
4. System records sale line item, and presents item description, price and running total.
   cashier repeats steps 3-4 until indicates done.
5. System presents total with taxes calculated.

…



Naming System Events and Operations

- The set of all required system operations is determined by identifying the system events.
  - makeNewSale()
  - addLineItem(itemID, quantity)
  - endSale()
  - makePayment(amount)



Object Oriented Analysis & Design
Module-3 (RL 3.2.3)

Sanjay Joshi

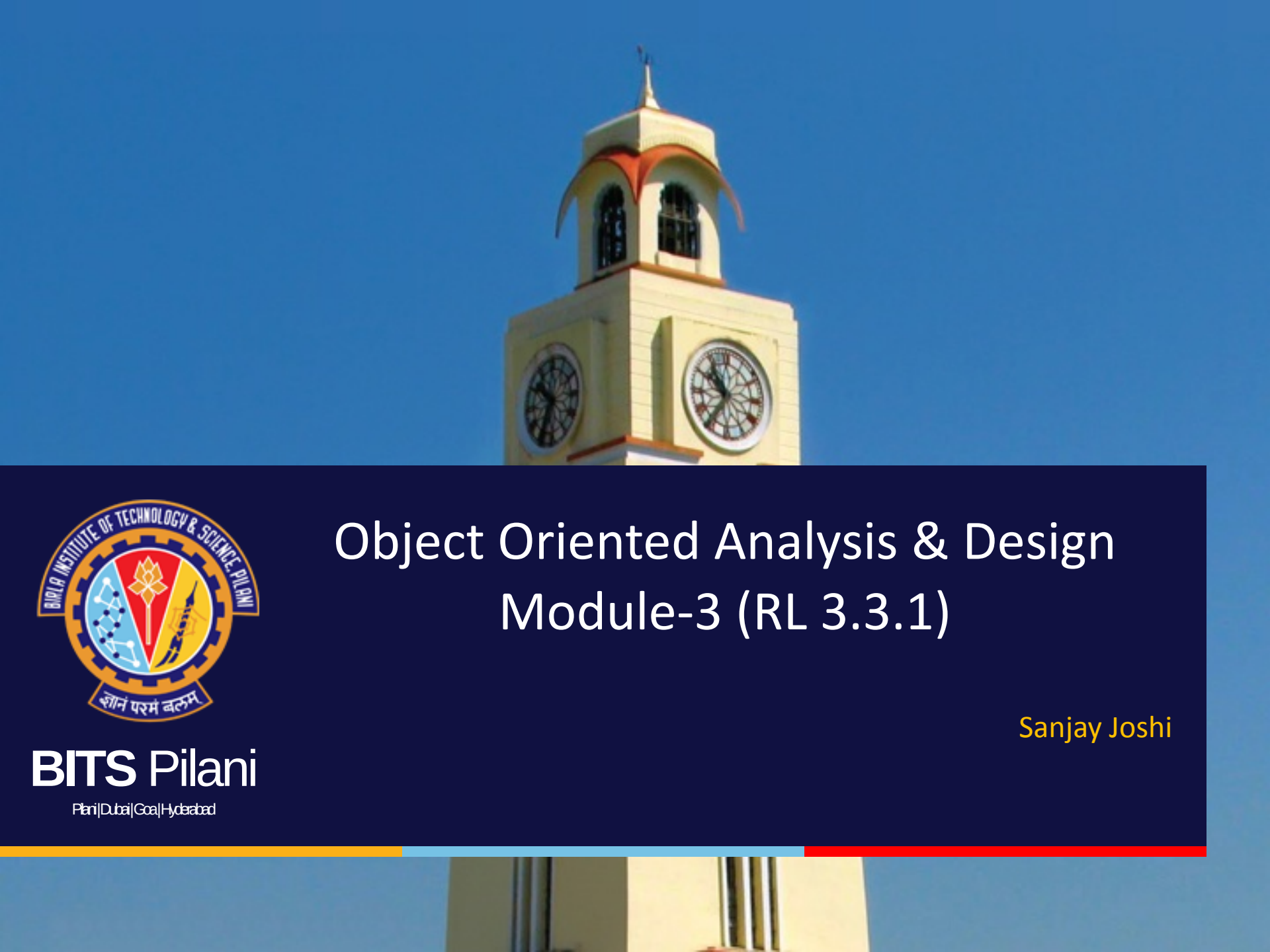
BITS Pilani

Pilani | Dubai | Goa | Hyderabad

Significance of SSD

Significance of SSD

- Interaction of the system with the outside world
- It is used to depict how System responses to external events
- It plays key role in GUI design of the system



Object Oriented Analysis & Design
Module-3 (RL 3.3.1)

Sanjay Joshi

BITS Pilani

Pilani|Dubai|Goa|Hyderabad

innovate achieve lead

BITS Pilani

Pilani|Dubai|Goa|Hyderabad

What is Operation Contracts

Why Operation Contracts?

- Details missing from System Sequence Diagram
- What an Analyst Reflect in Operation Contract?

What is Operation Contract?

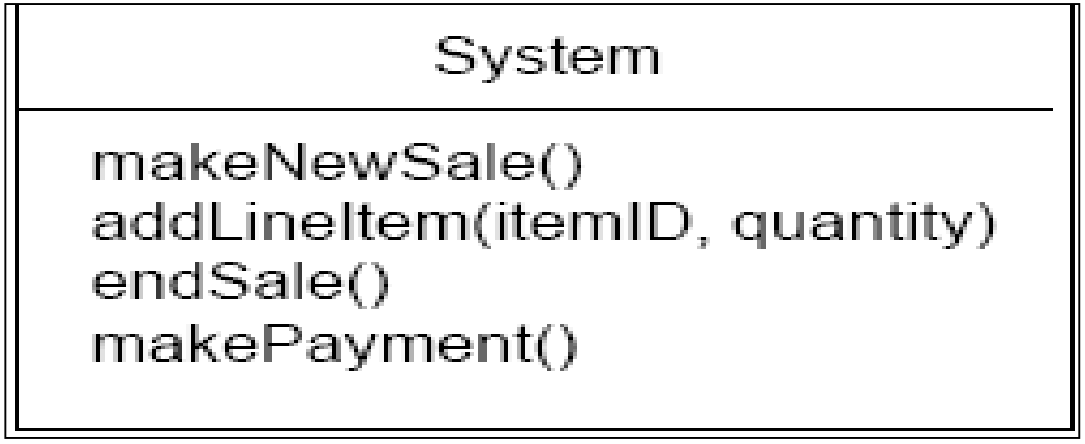
- Operation Contracts are documents that describe system behavior.

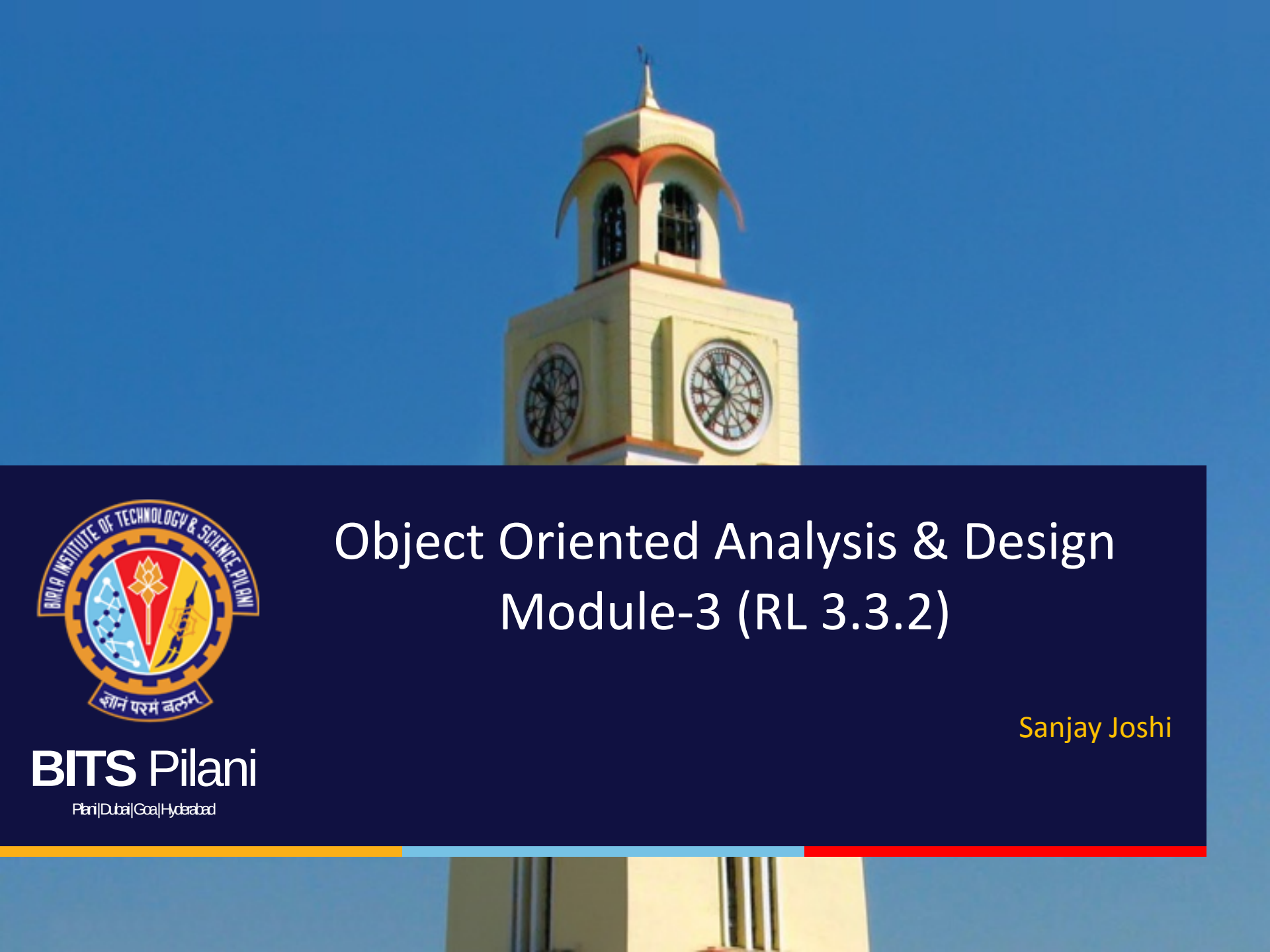
- Operation Contracts may be defined for **system operations**.
  - Operations that the system (as a black box) offers in its public interface to handle incoming system events.

- The entire set of system operations across all use cases, defines the public system interface.

System Operations and the System Interface

- In the UML the system as a whole can be represented as a class.
- Contracts are written for each system operation to describe its behavior.





Object Oriented Analysis & Design
Module-3 (RL 3.3.2)

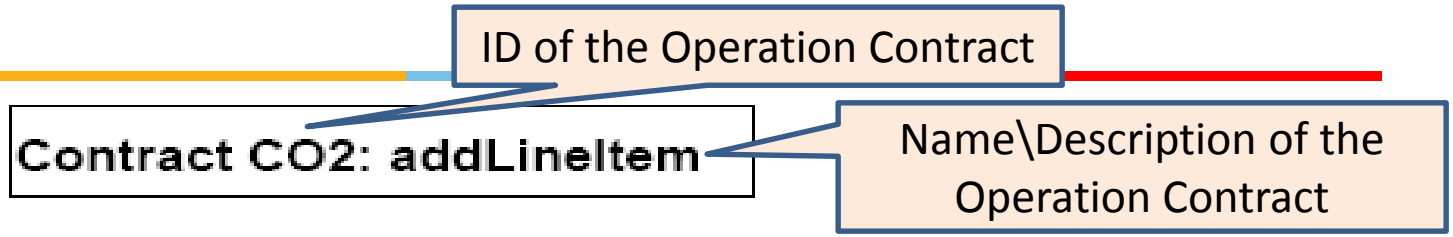
Sanjay Joshi

BITS Pilani

Pilani | Dubai | Goa | Hyderabad

Represent Operation Contract in UML

Operation Contract in UML



Operation: addLineItem (itemID: integer, quantity: integer)

Cross References: Use Cases: Process Sale.

Pre-conditions: There is a sale underway.

Post-conditions:

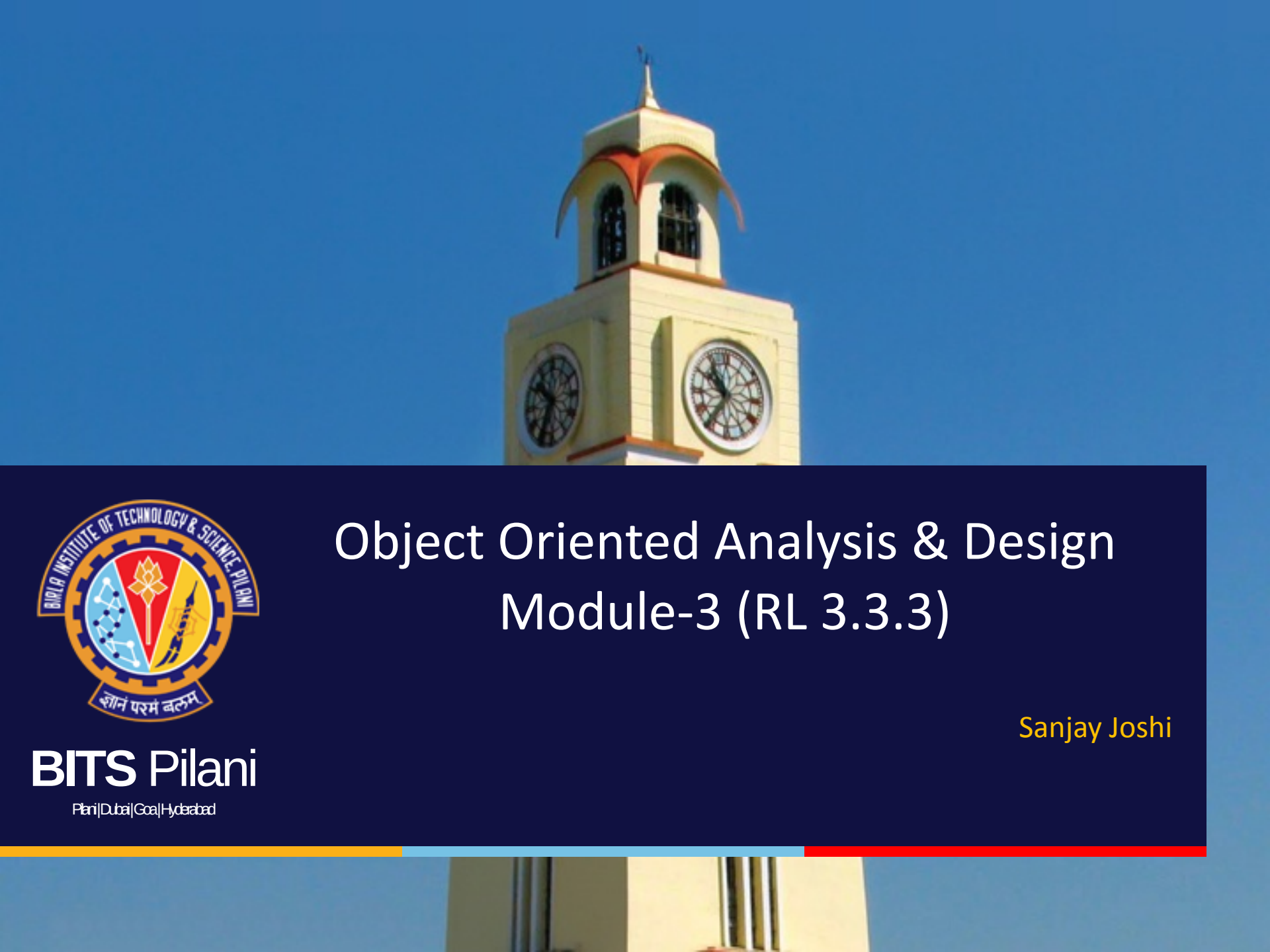
- A SalesLineItem instance *sli* was created. (instance creation)
- *sli* was associated with the Sale. (association formed)
- *sli.quantity* was set to quantity. (attribute modification)
- *sli* was associated with a ProductSpecification, based on itemID match (association formed)

Operation Signature

Use Case Reference

Prerequisite

Post Condition in Past Tense



Object Oriented Analysis & Design
Module-3 (RL 3.3.3)

Sanjay Joshi

BITS Pilani

Pilani|Dubai|Goa|Hyderabad

Writing Operation Contract for PoS

Operation Contract: addLineItem

Contract CO2: addLineItem

Operation: addLineItem (itemID: integer, quantity: integer)

Cross References: Use Cases: Process Sale.

Pre-conditions: There is a sale underway.

Post-conditions:

- A SalesLineItem instance *sli* was created. (instance creation)
- *sli* was associated with the Sale. (association formed)
- *sli.quantity* was set to quantity. (attribute modification)
- *sli* was associated with a ProductSpecification, based on itemID match (association formed)

Pre- and Postconditions

- Preconditions are assumptions about the state of the system before execution of the operation.
- A postcondition is an assumption that refers to the state of the system after completion of the operation.
  - The postconditions are not actions to be performed during the operation.
  - Describe changes in the state of the objects in the Domain Model (instances created, associations are being formed or broken, and attributes are changed)

addLineItem postconditions

- Instance Creation and Deletion

  After the itemID and quantity of an item have been entered by the cashier, what new objects should have been created?

  – A SalesLineItem instance *sli* was created.

addLineItem postconditions

- Attribute Modification

  After the itemID and quantity of an item have been entered by the cashier, what attributes of new or existing objects should have been modified?

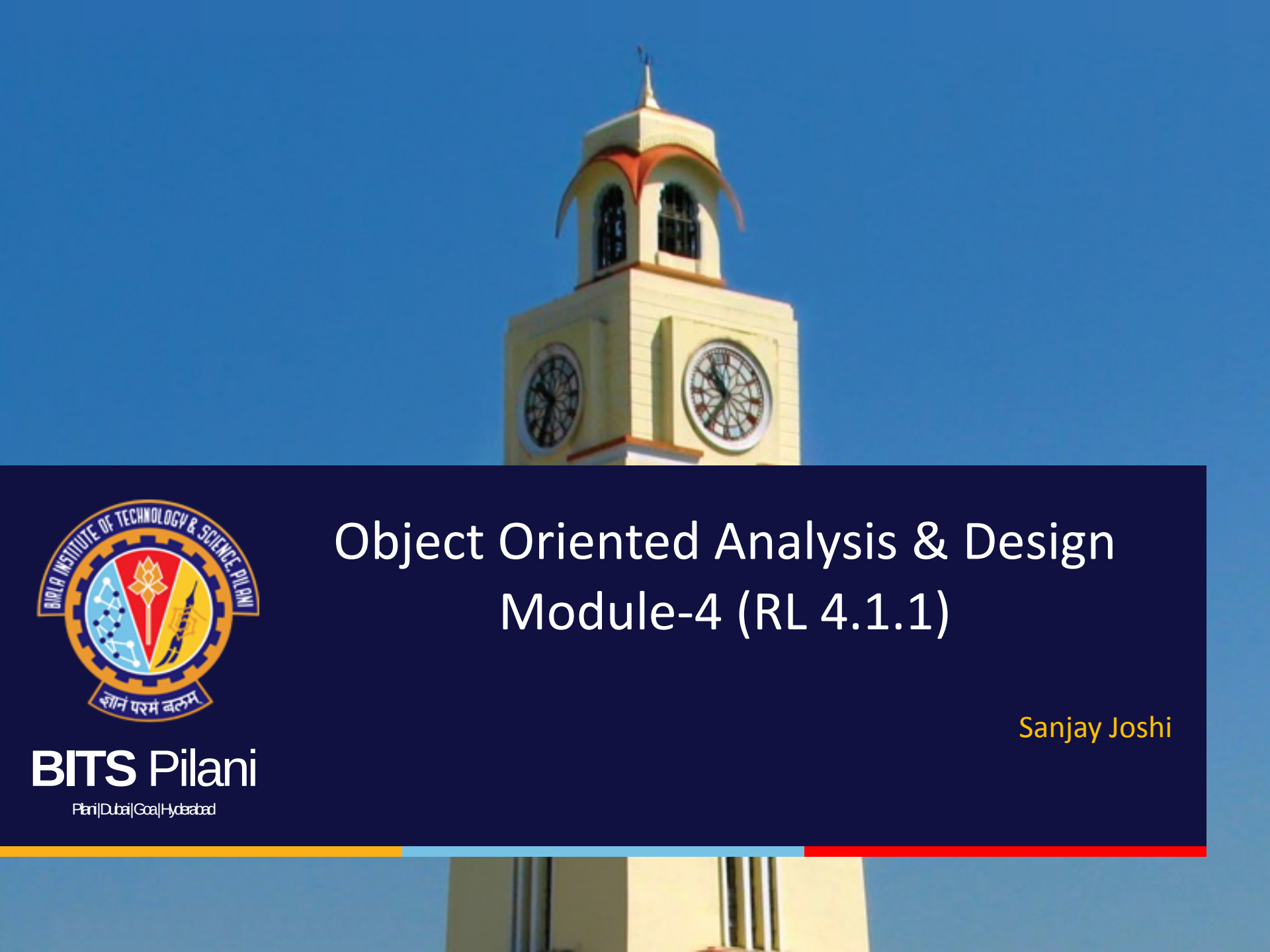
  – sli.quantity was set to quantity (attribute modification).

addLineItem postconditions

innovate achieve lead

- Associations Formed and Broken

  After the itemID and quantity of an item have been entered by the cashier, what associations between new or existing objects should have been formed or broken?
  - sli was associated with the current Sale (association formed).
  - sli was associated with a ProductSpecification, based on itemID match (association formed).



Object Oriented Analysis & Design
Module-4 (RL 4.1.1)

Sanjay Joshi

BITS Pilani

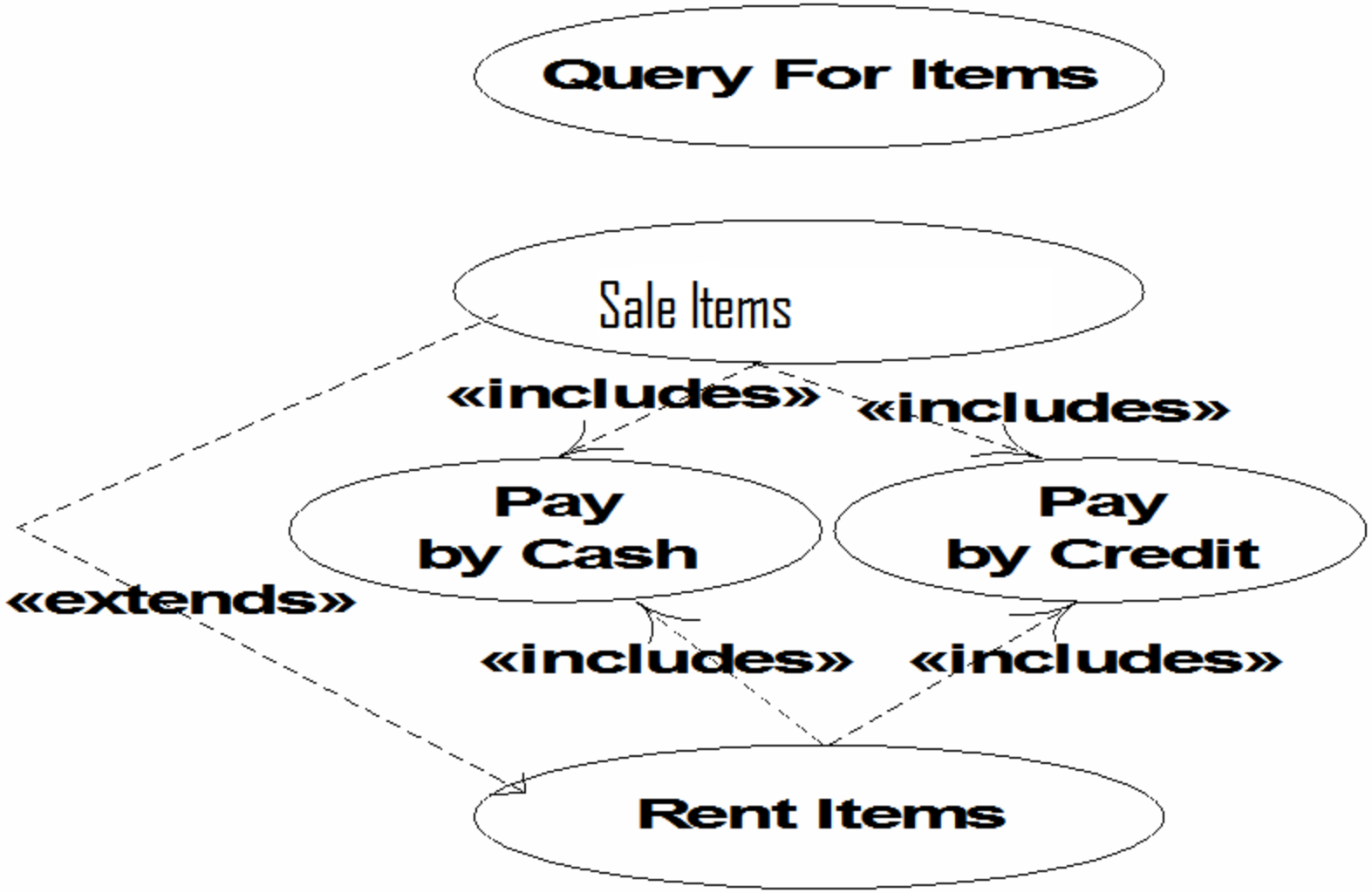
Pilani|Dubai|Goa|Hyderabad

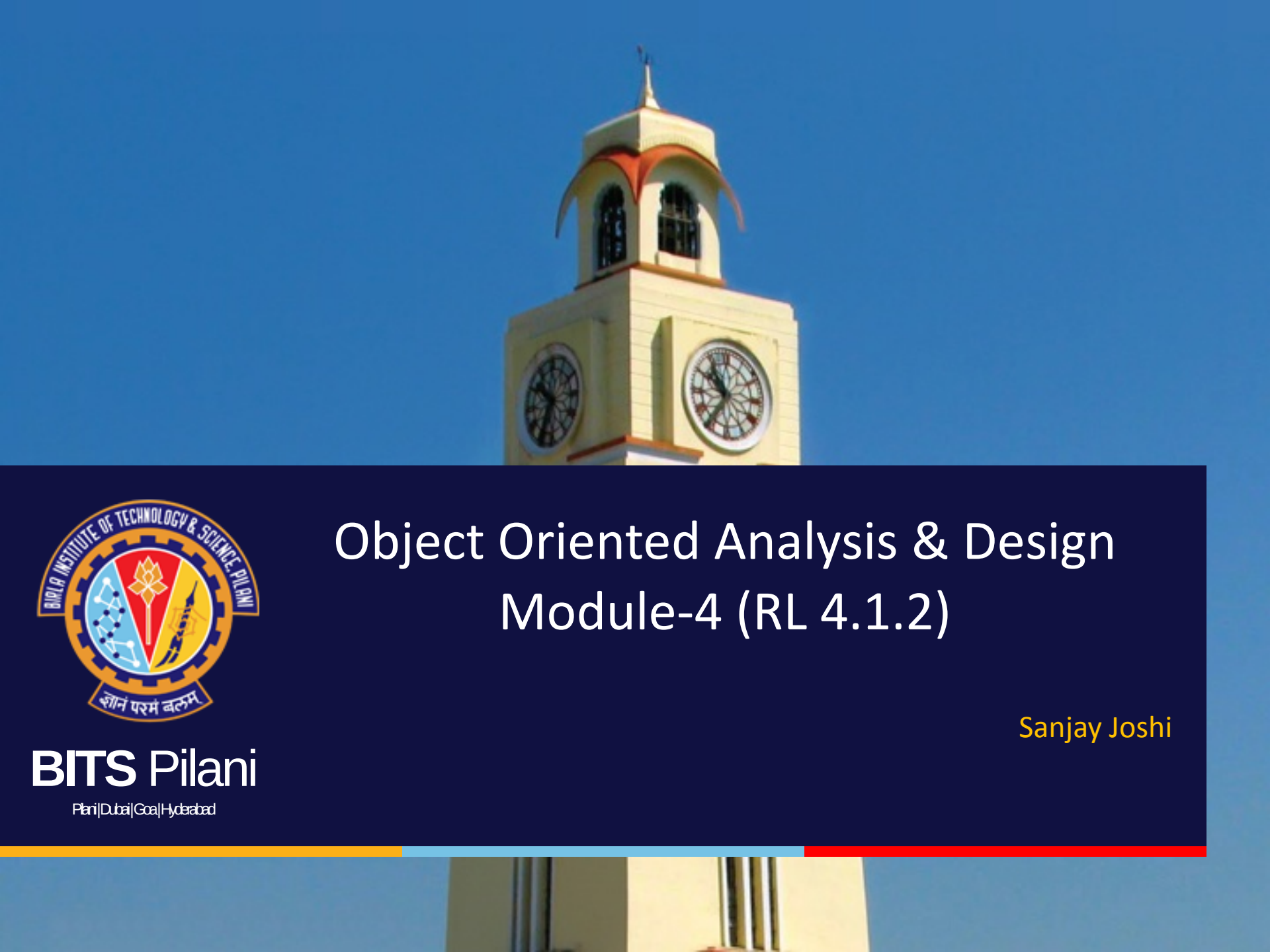
Relating Use Cases : includes, extends relationships

Relating Use Cases

- When creating the use case diagram, it can be useful (in terms of comprehension and simplification) to:
  - factor out shared sub-processes
    - use the <<includes>> relationship
  - show extensions
    - use the <<extends>> relationship

Video Store Information System





Object Oriented Analysis & Design
Module-4 (RL 4.1.2)

Sanjay Joshi

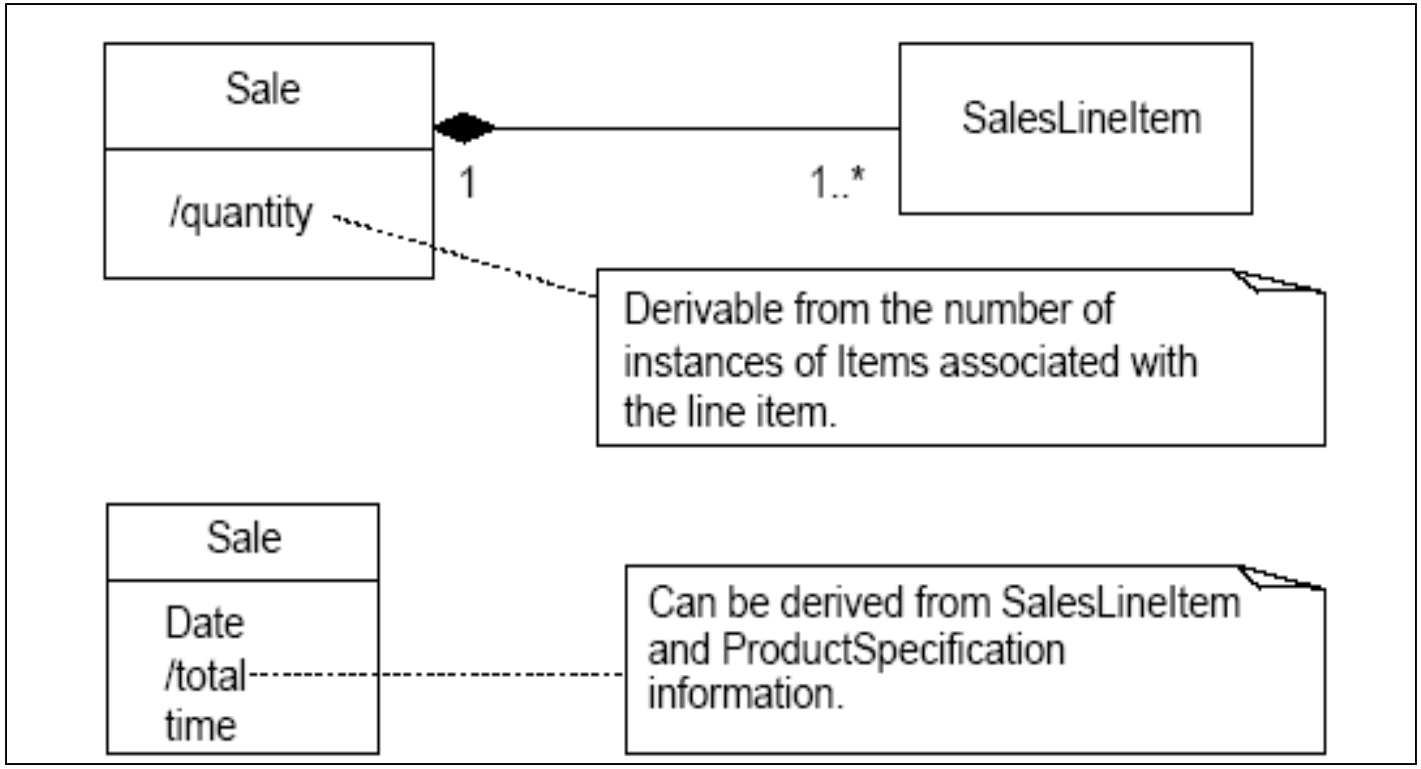
BITS Pilani

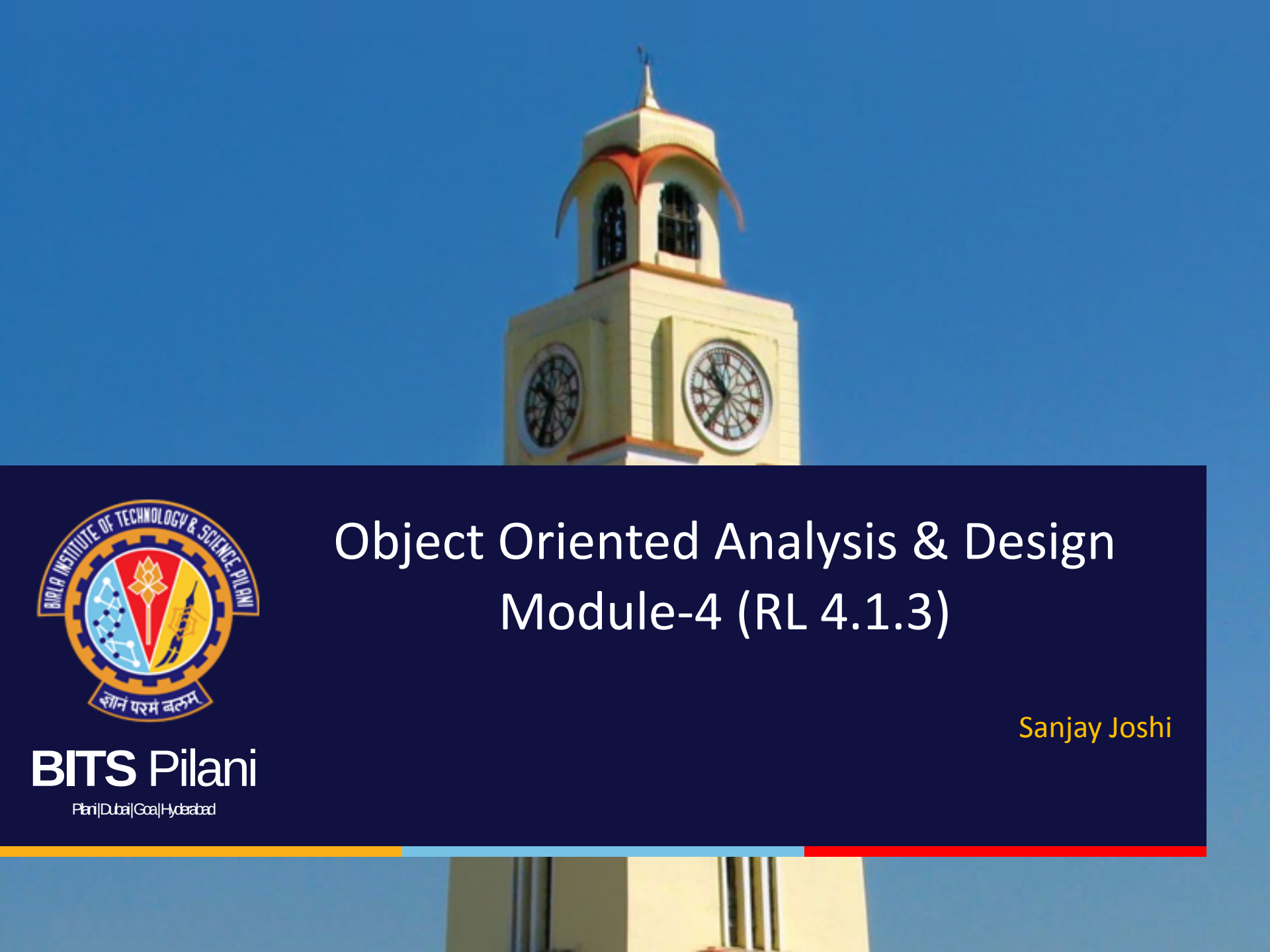
Pilani|Dubai|Goa|Hyderabad

Refining Domain Model : Derived Attributes

Derived Elements

- A derived element can be determined from others.





Object Oriented Analysis & Design
Module-4 (RL 4.1.3)

Sanjay Joshi

BITS Pilani

Pilani | Dubai | Goa | Hyderabad

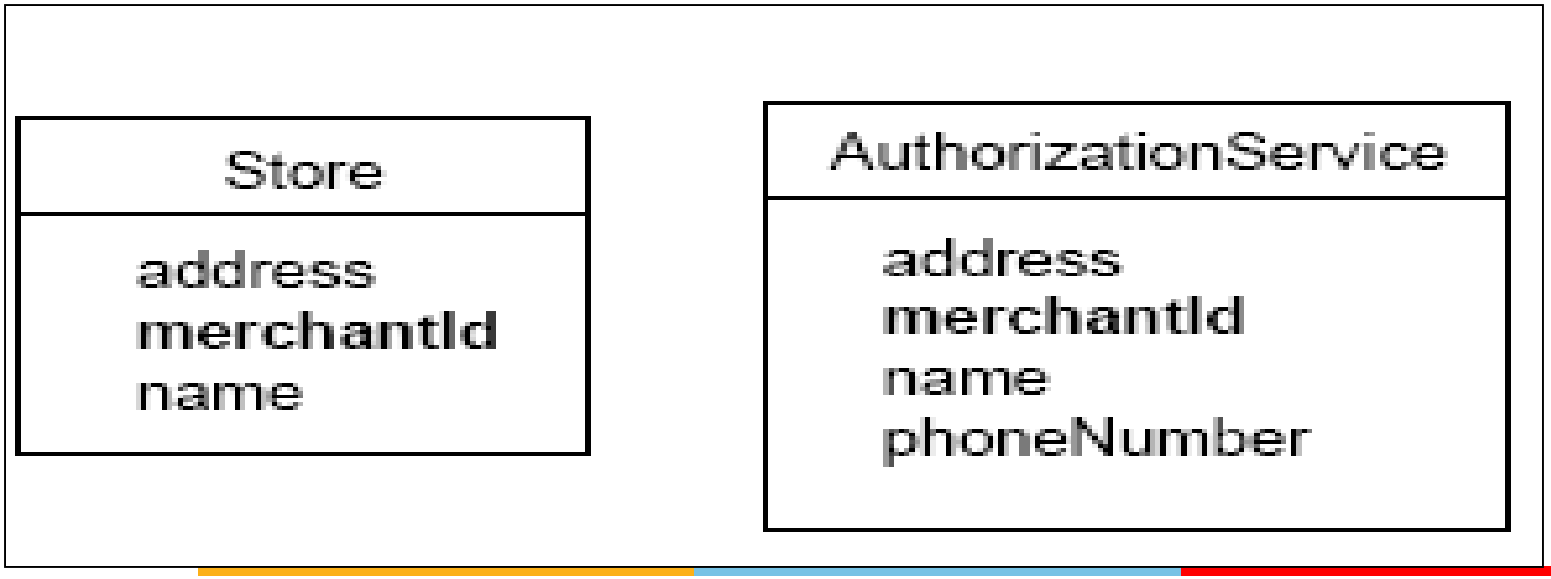
Refining Domain Model : Association Classes

Association Classes

- Authorization services assign a merchant ID to each store for identification during communications.
- A payment authorization request from the store to an authorization service requires the inclusion of the merchant ID that identifies the store to the service.
- Consider a store that has a different merchant ID for each service. (e.g. Id for Visa is XXX, Id for MC is YYY, etc.)

Association Classes

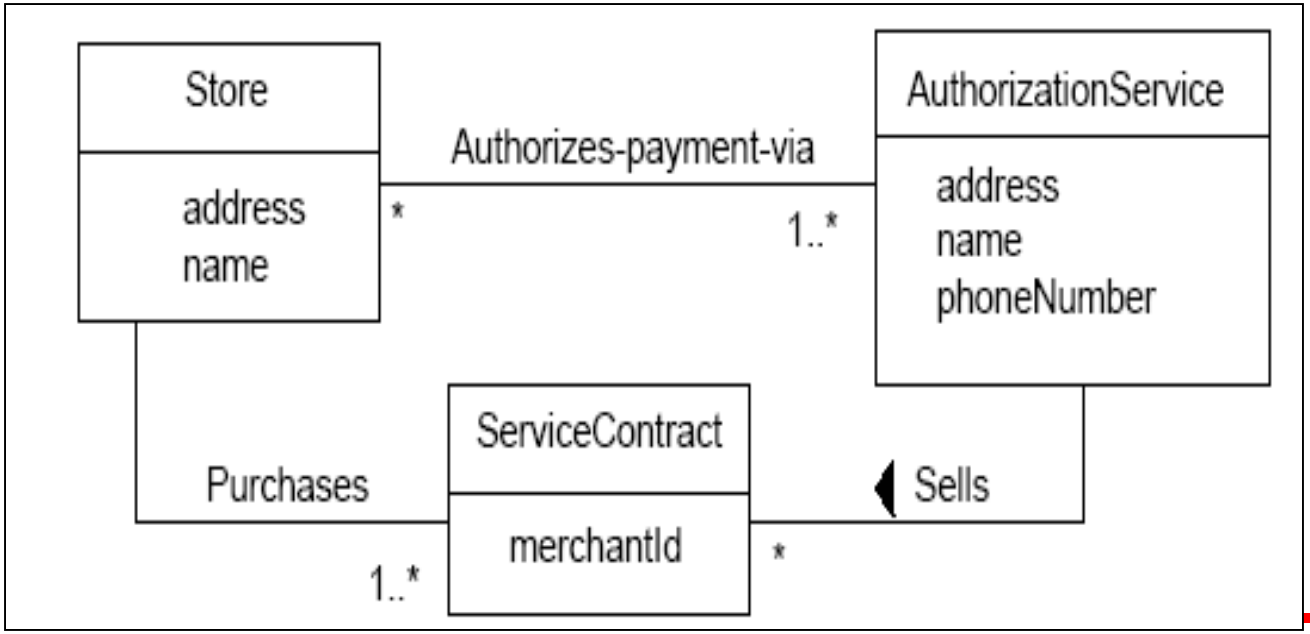
- Where in the conceptual model should the merchant ID attribute reside?
- Placing the merchantId in the Store is incorrect, because a Store may have more than one value for merchantId.
- The same is true with placing it in the AuthorizationService



Association Classes

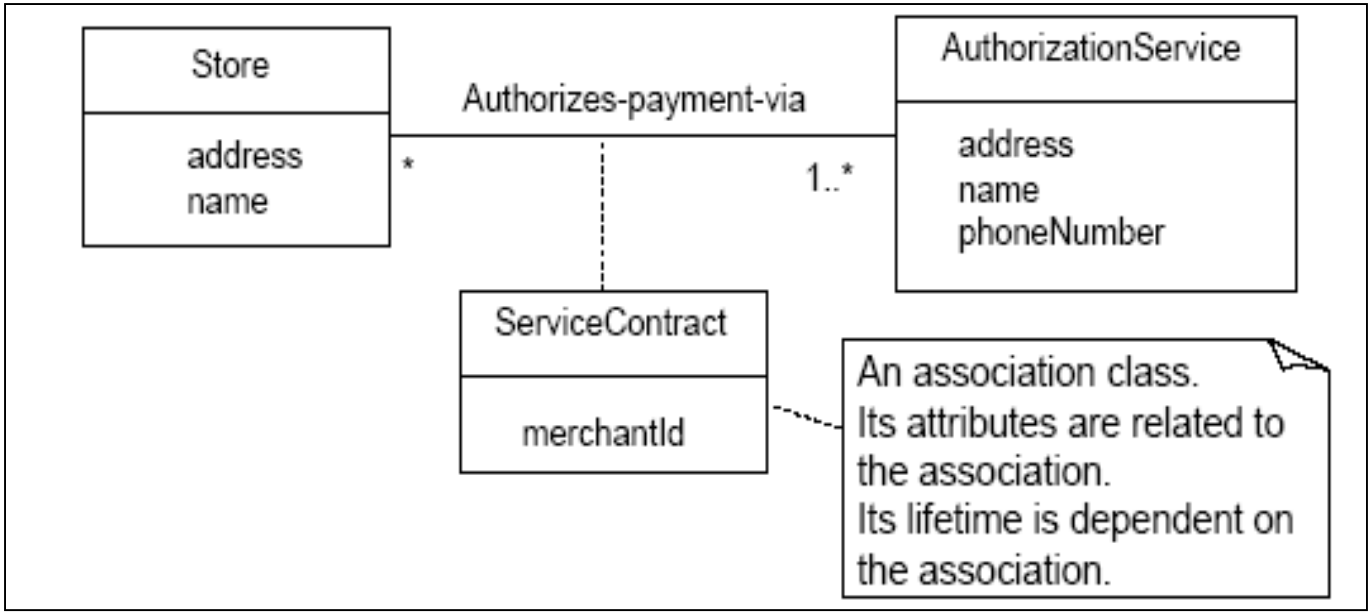
- In a conceptual model, if a class C can simultaneously have many values for the same kind of attribute A, do not place

  attribute A in C. Place attribute A in another type that is associated with C.



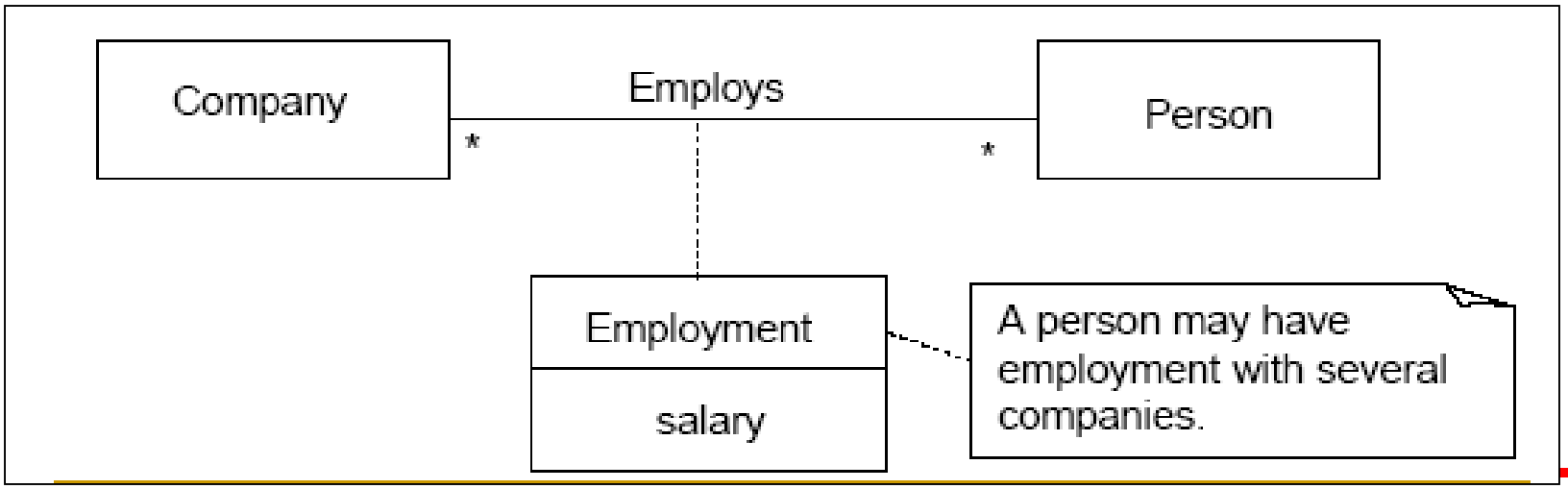
Association Classes

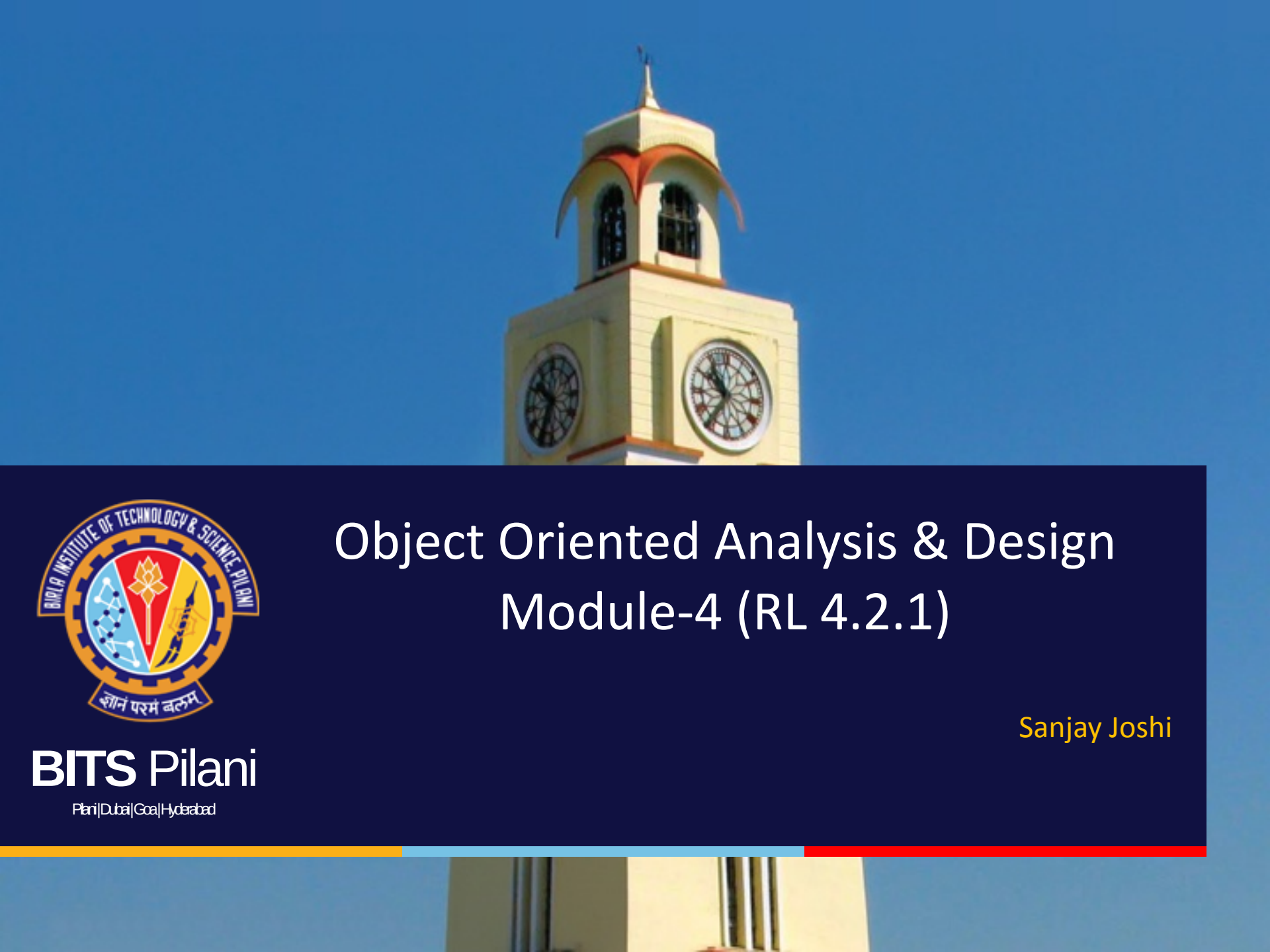
- The merchantId is an attribute related to the association between the Store and AuthorizationService; it depends on their relationship.
- ServiceContract may then be modeled as an association class.



Guidelines for Association Classes

- An attribute is related to an association.
- Instances of the association class have a life-time dependency on the association.
- There is a many-to-many association between two concepts.
- The presence of a many-to-many association between two concepts is a clue that a useful associative type should exist in the background somewhere.





Object Oriented Analysis & Design
Module-4 (RL 4.2.1)

Sanjay Joshi

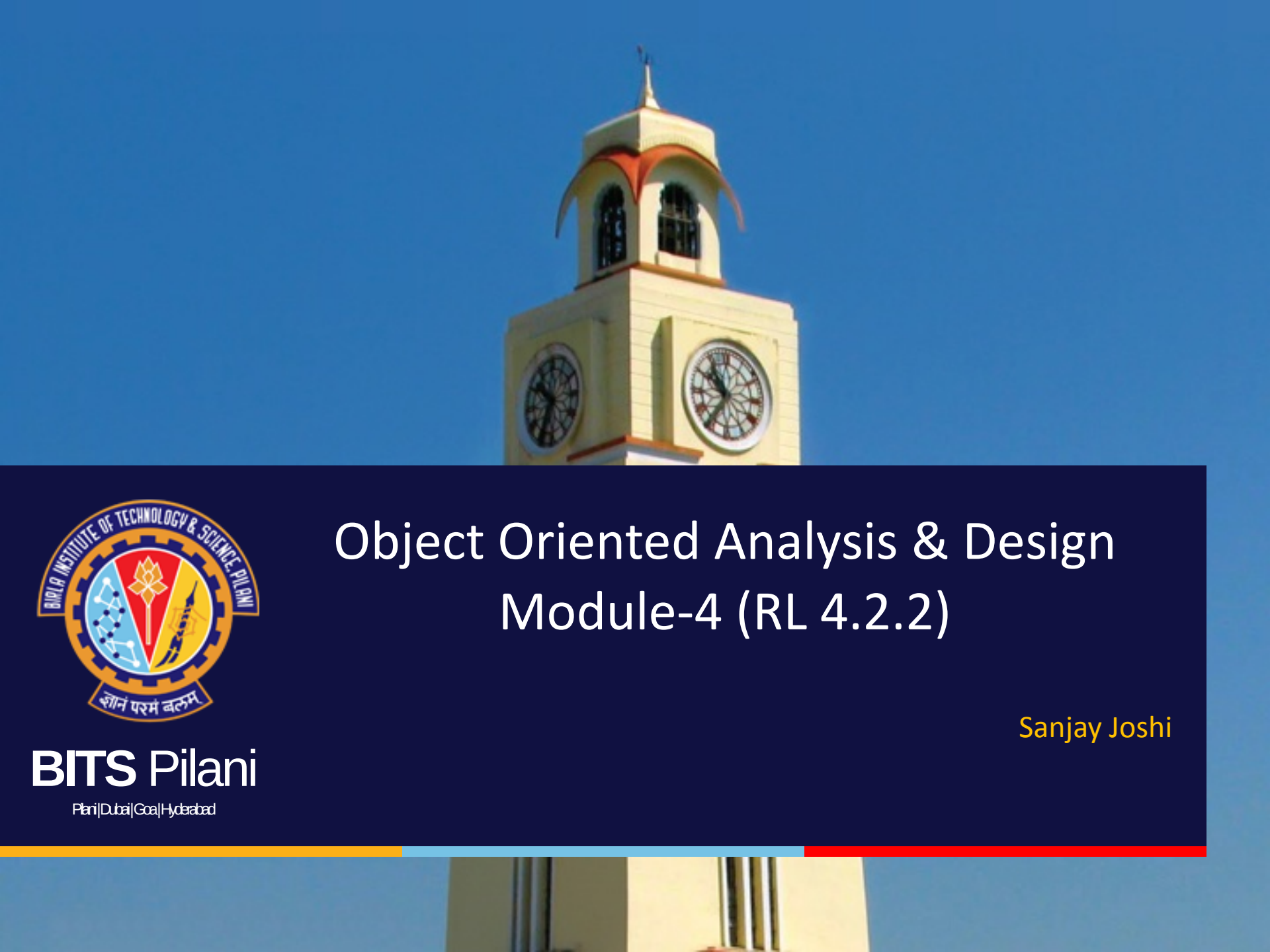
BITS Pilani

Pilani|Dubai|Goa|Hyderabad

What is Interaction Diagram?

What is Interaction Diagram?

- *Shows interaction among Objects within the system*
- *It is scenario specific diagram*
- *Interaction by means of exchange of messages*
- *It is part of Lower Level Design*
- *Is it same as SSD (System Sequence Diagram)?*



Object Oriented Analysis & Design
Module-4 (RL 4.2.2)

Sanjay Joshi

BITS Pilani

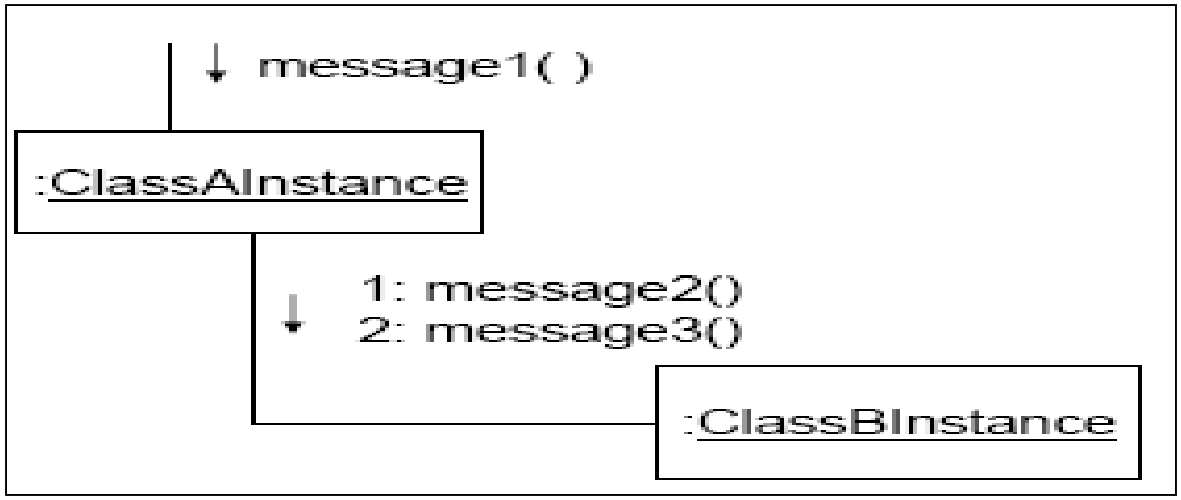
Pilani | Dubai | Goa | Hyderabad

Types of Interactions Diagram

Types of Interaction Diagrams

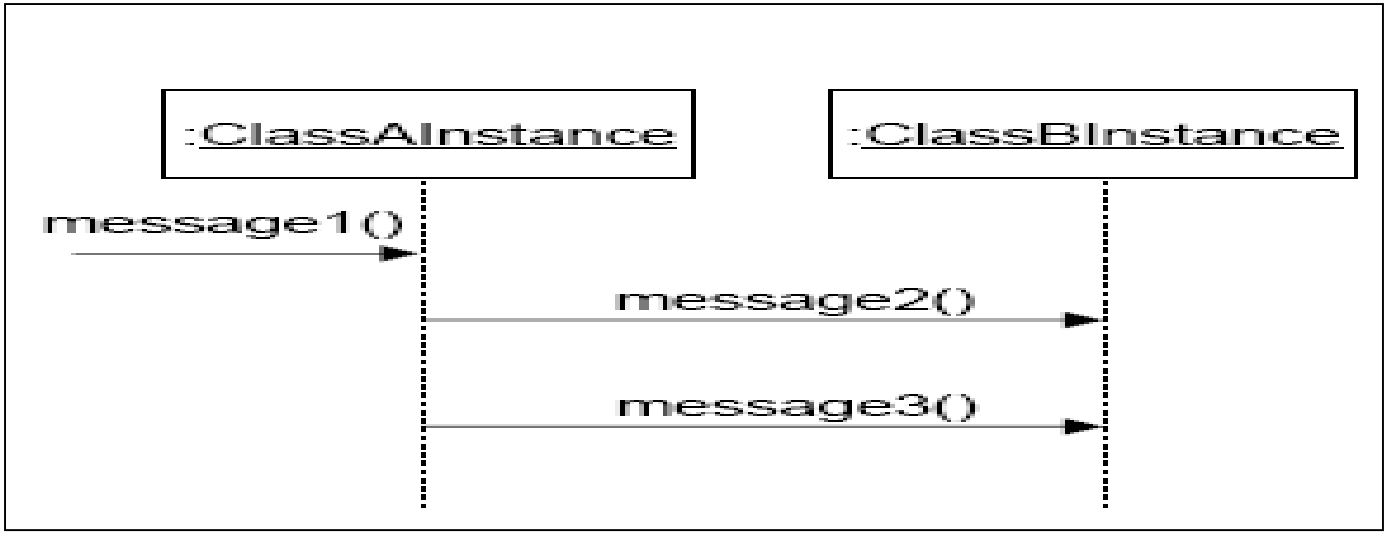
- *Two Types : Collaboration Diagram & Sequence Diagram*

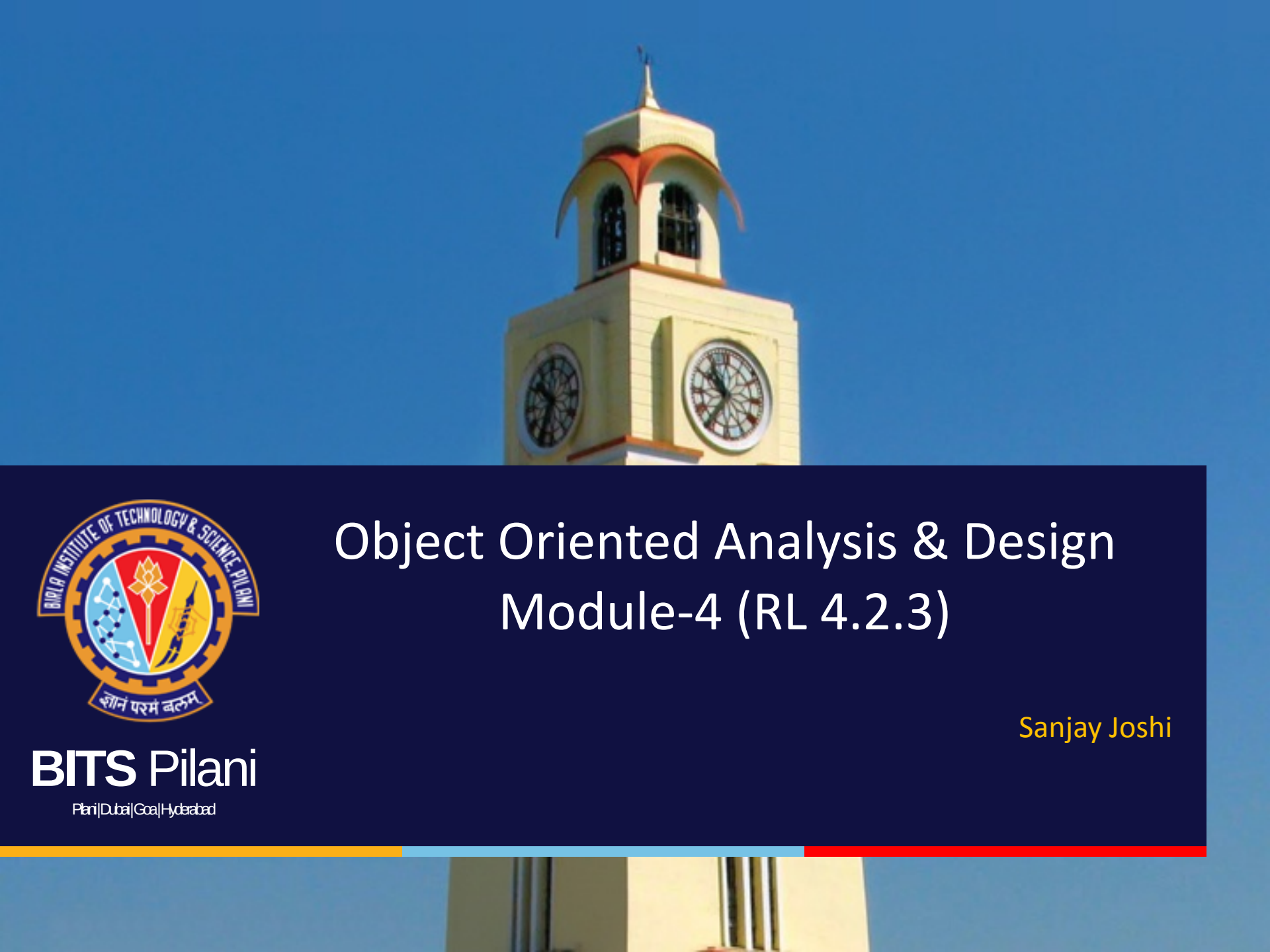
  Collaboration diagrams illustrate object interactions in a graph or network format.



Types of Interaction Diagrams

- *Sequence diagrams* illustrate interactions in a kind of fence format.





Object Oriented Analysis & Design
Module-4 (RL 4.2.3)

Sanjay Joshi

BITS Pilani

Pilani|Dubai|Goa|Hyderabad

innovate achieve lead

BITS Pilani

Pilani|Dubai|Goa|Hyderabad

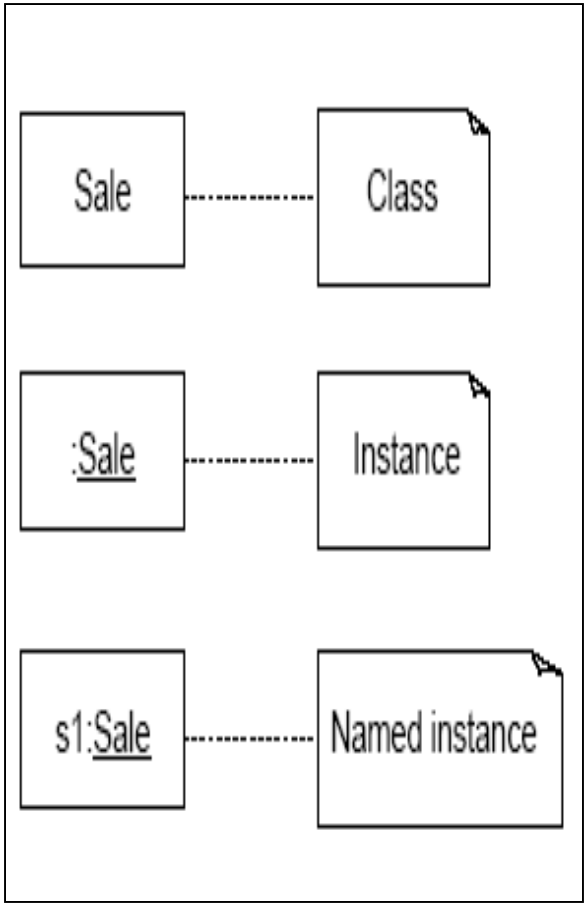
Representation of Interaction Diagrams in UML

Illustrating Classes and Instances

- To show an instance of a class, the regular class box graphic symbol is used, but the name is underlined.

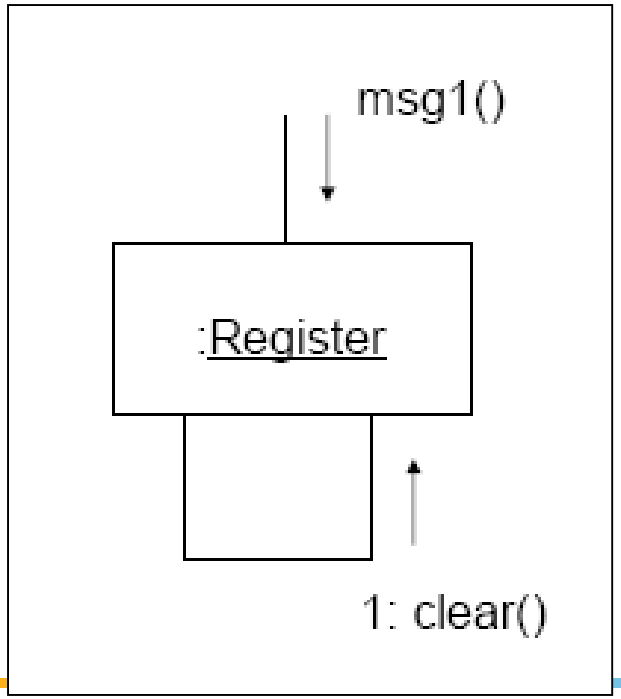
  Additionally a class name should be preceded by a colon.
- An instance name can be used to uniquely

  identify the instance.



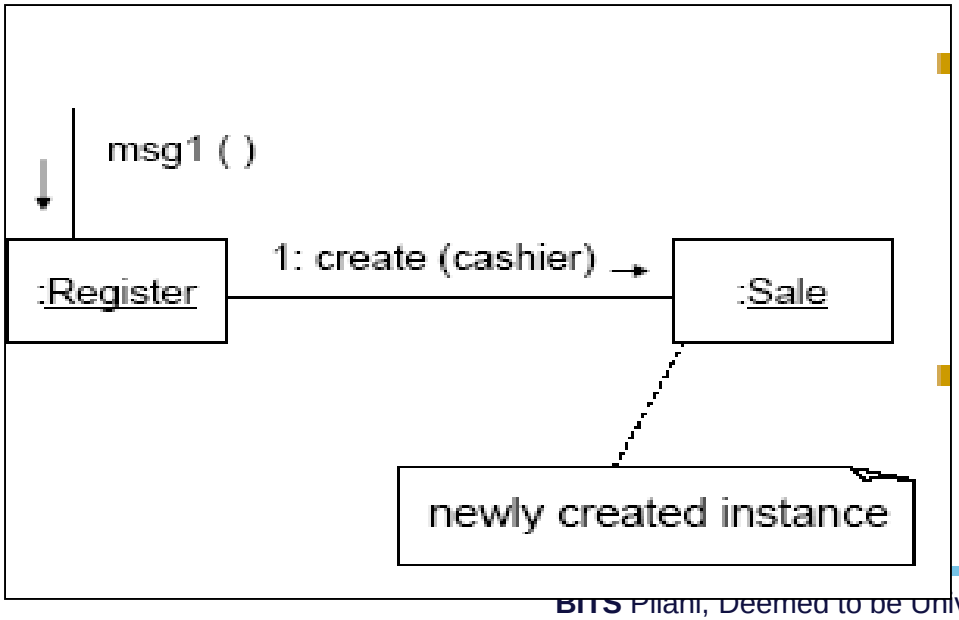
Messages to “self” or “this”

- A message can be sent from an object to itself.
- This is illustrated by a link to itself, with messages flowing along the link.



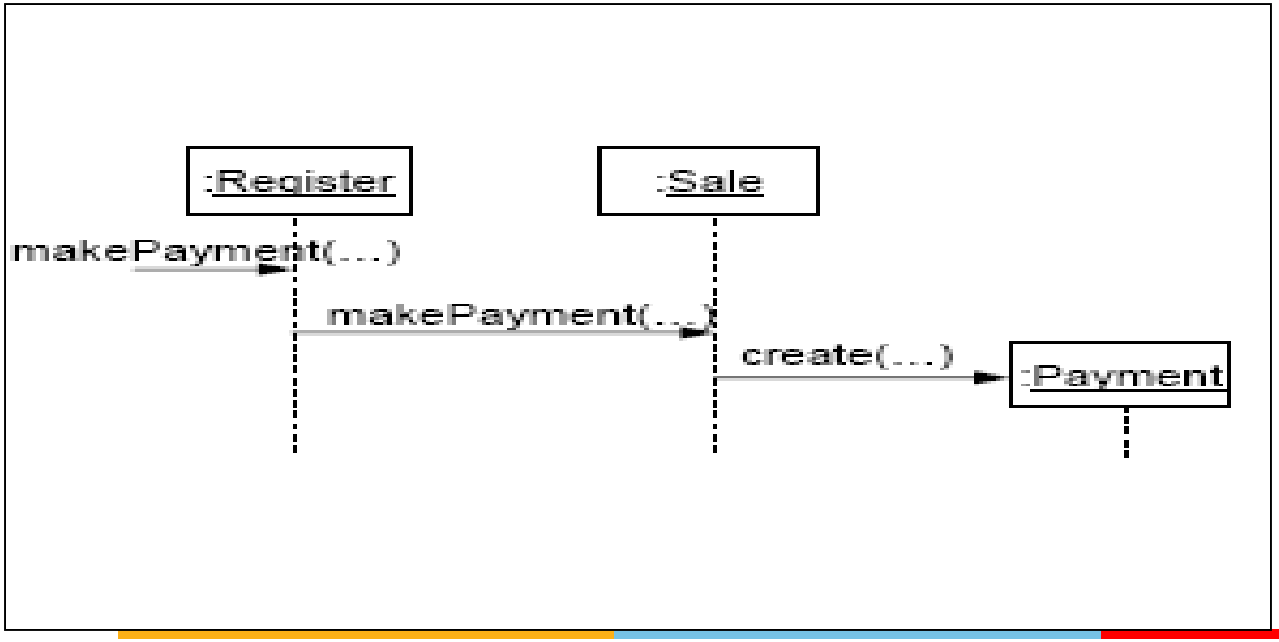
Creation of Instances

- The language independent creation message is create, being sent to the instance being created.
- The create message may include parameters, indicating passing of initial values.



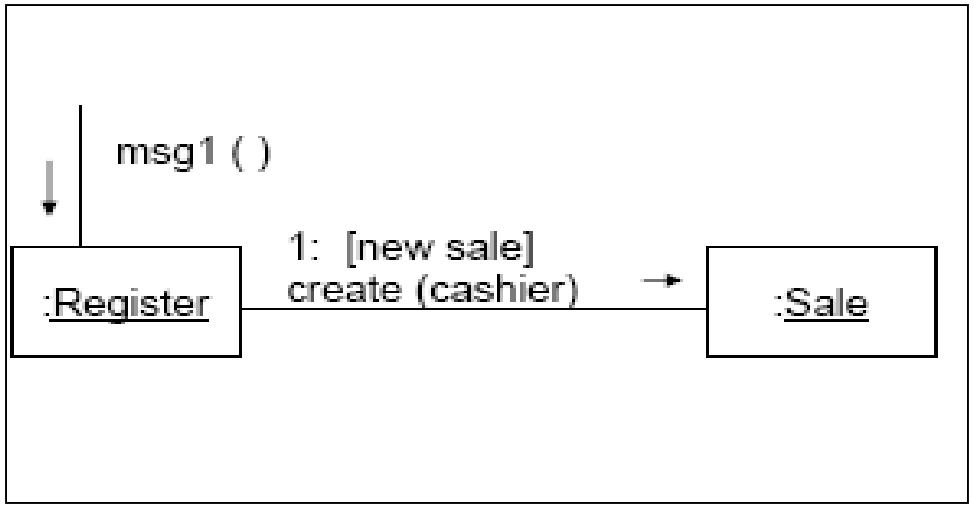
Creation of Instances

- An object lifeline shows the extend of the life of an object in the diagram.
- Note that newly created objects are placed at their creation height.

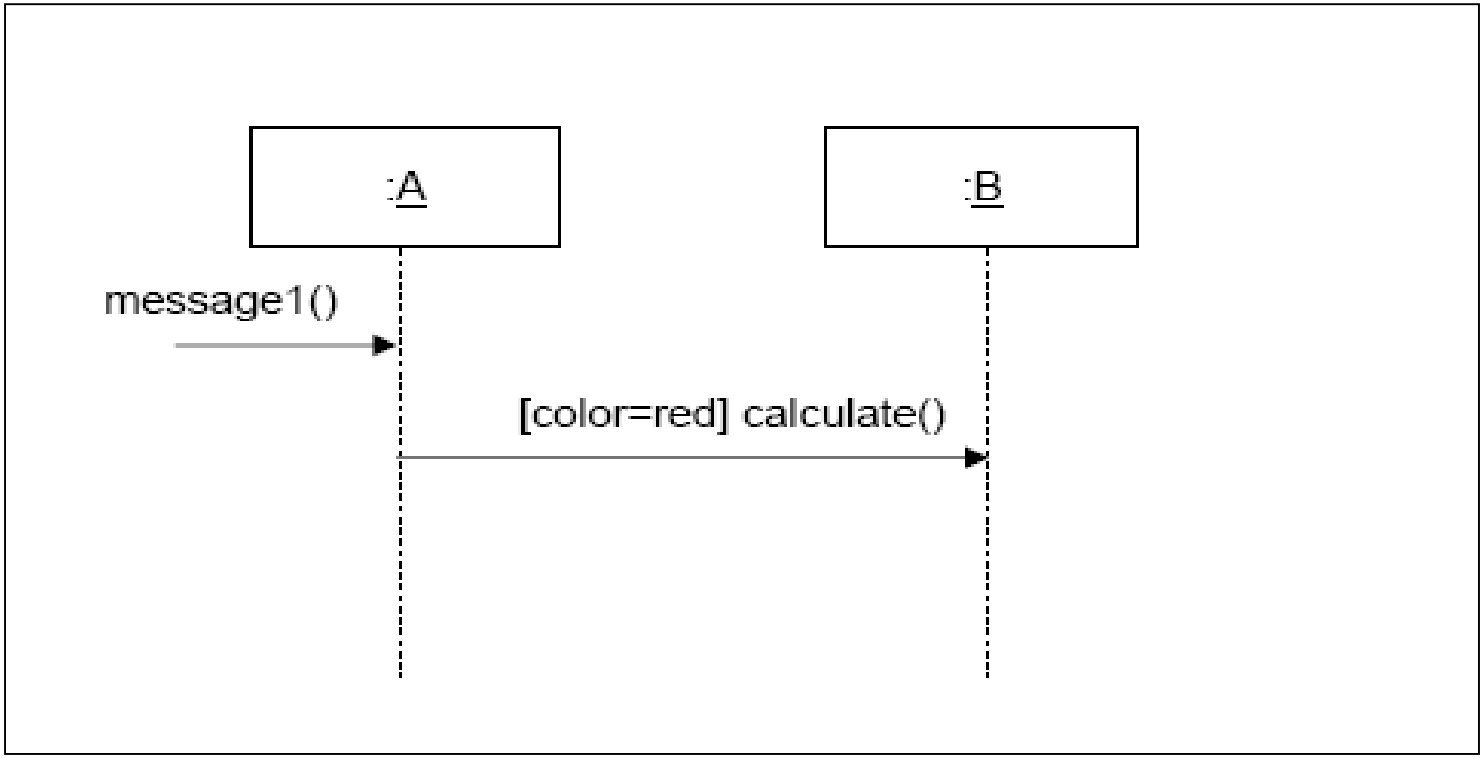


Conditional Messages

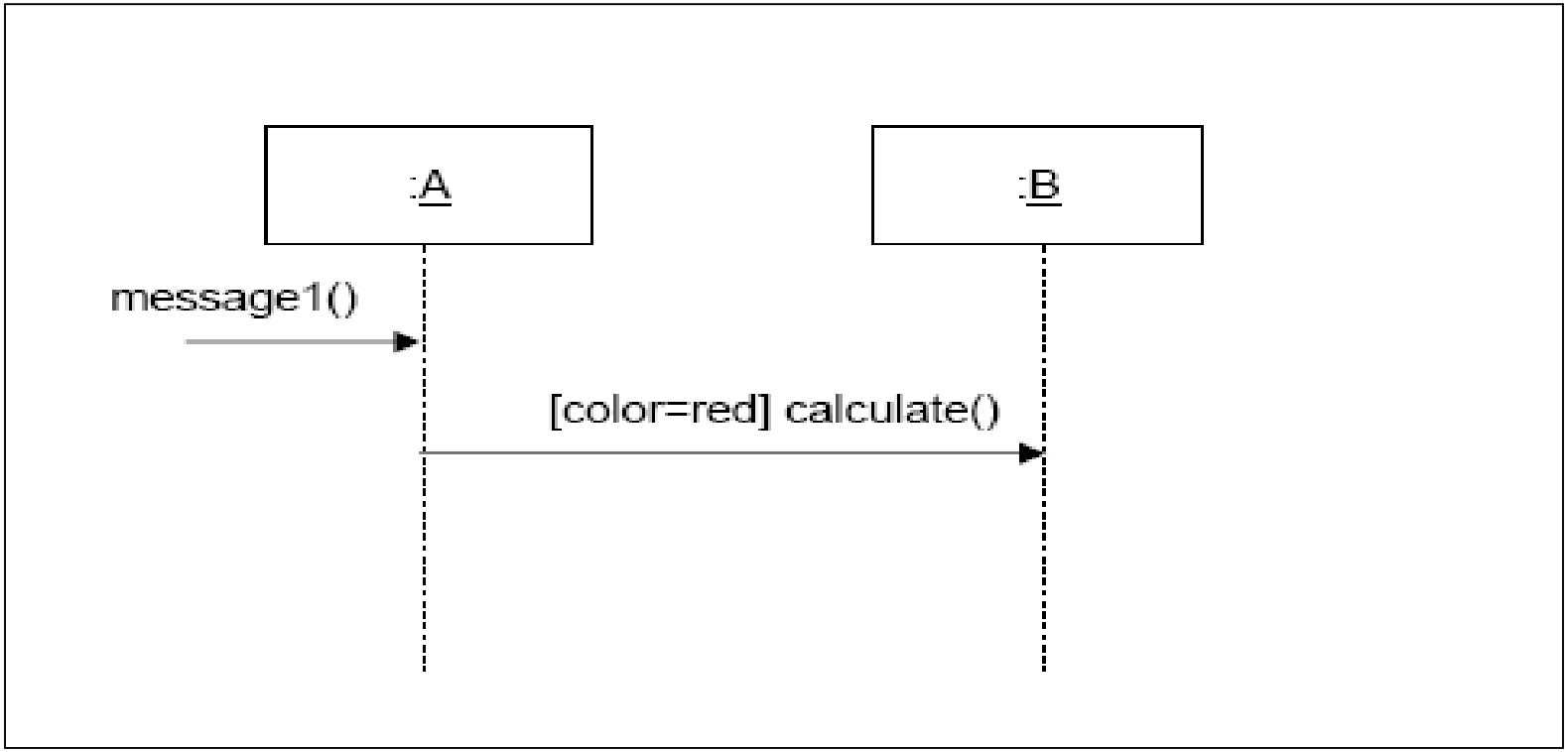
- A conditional message is shown by following a sequence number with a conditional clause in square brackets, similar to the iteration clause.
- The message is sent only if the clause evaluates to true.



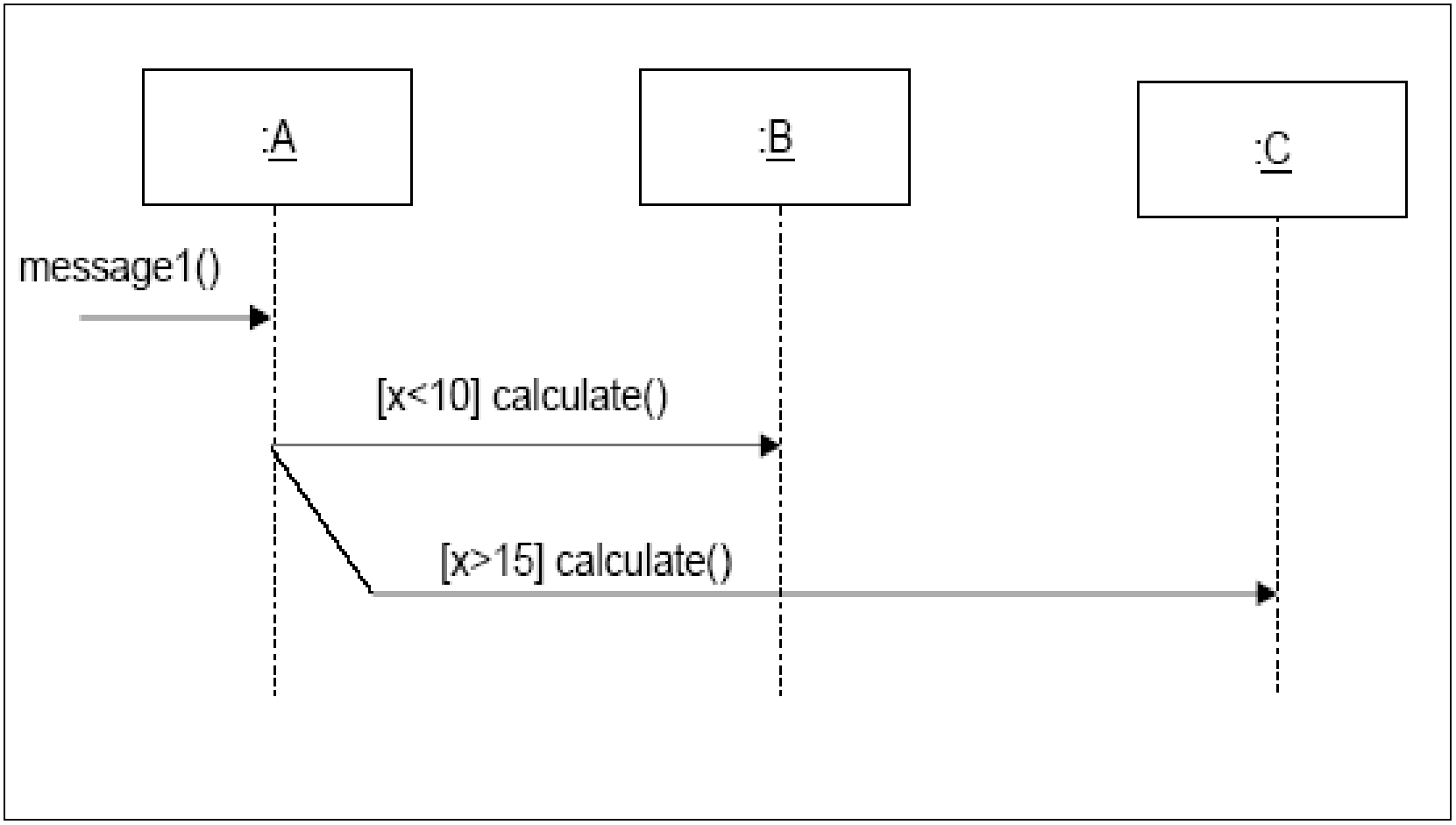
Conditional Messages



Mutually Exclusive Conditional Paths

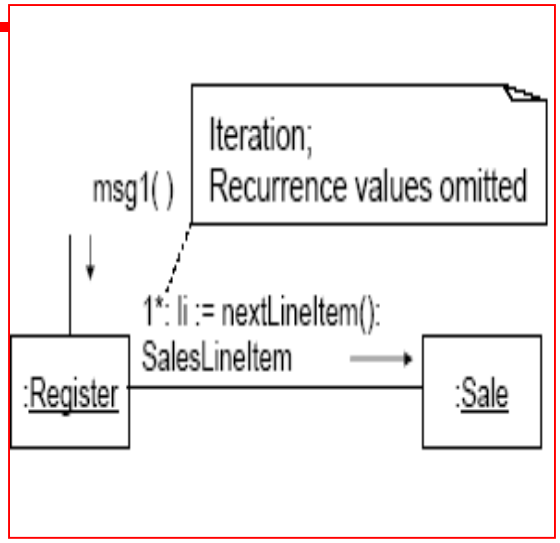


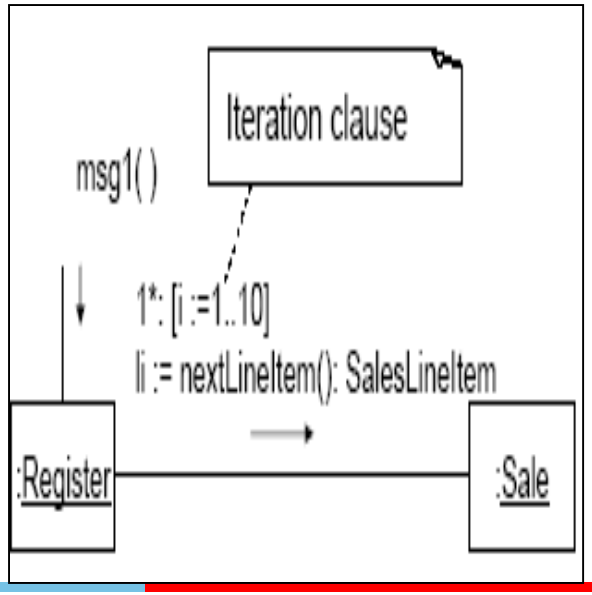
Mutually Exclusive Conditional Messages

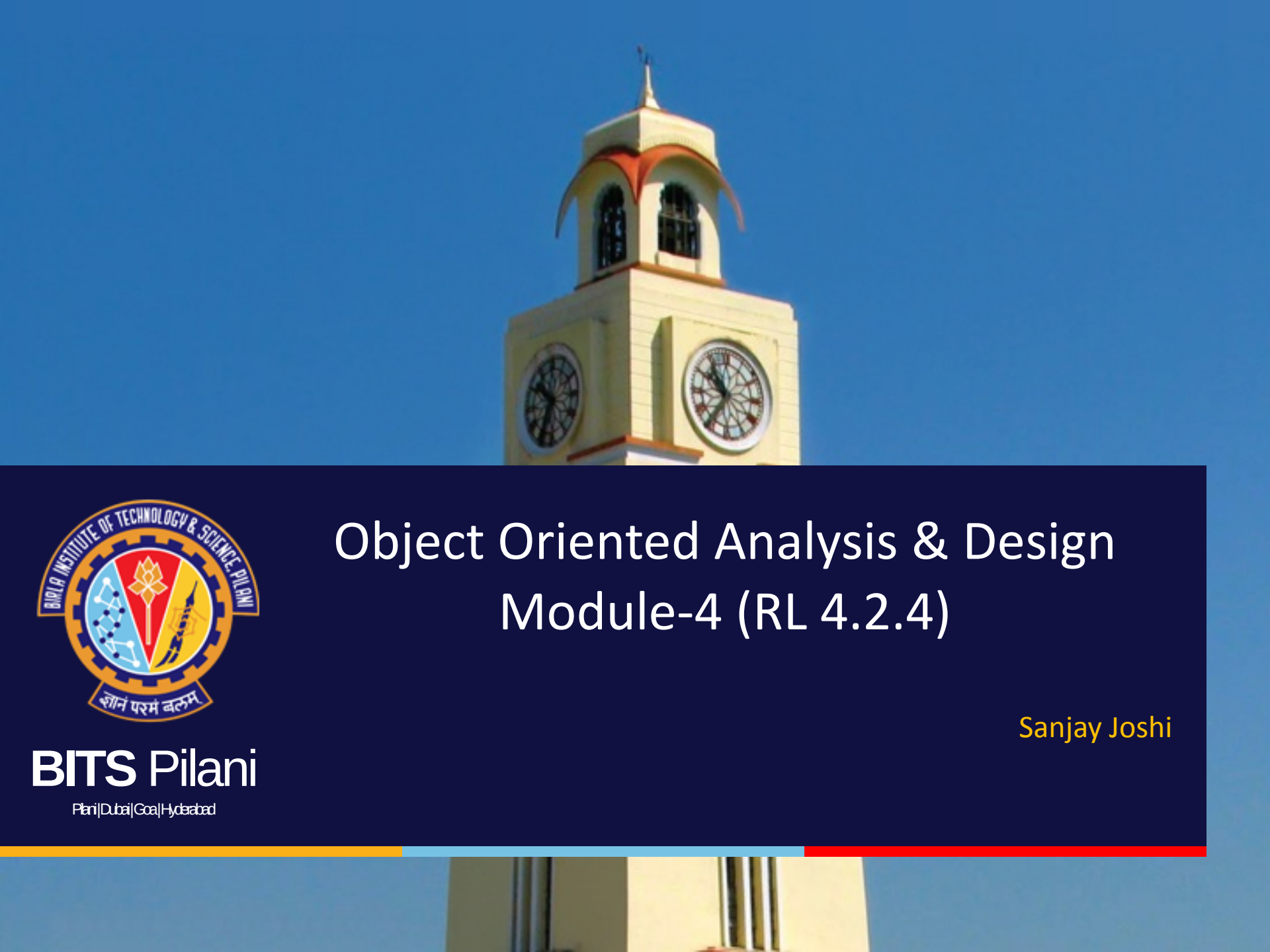


Iteration or Looping

- Iteration is indicated by following the sequence number with a star *
- This expresses that the message is being sent repeatedly, in a loop, to the receiver.
- It is also possible to include an iteration clause indicating the recurrence values.







Object Oriented Analysis & Design
Module-4 (RL 4.2.4)

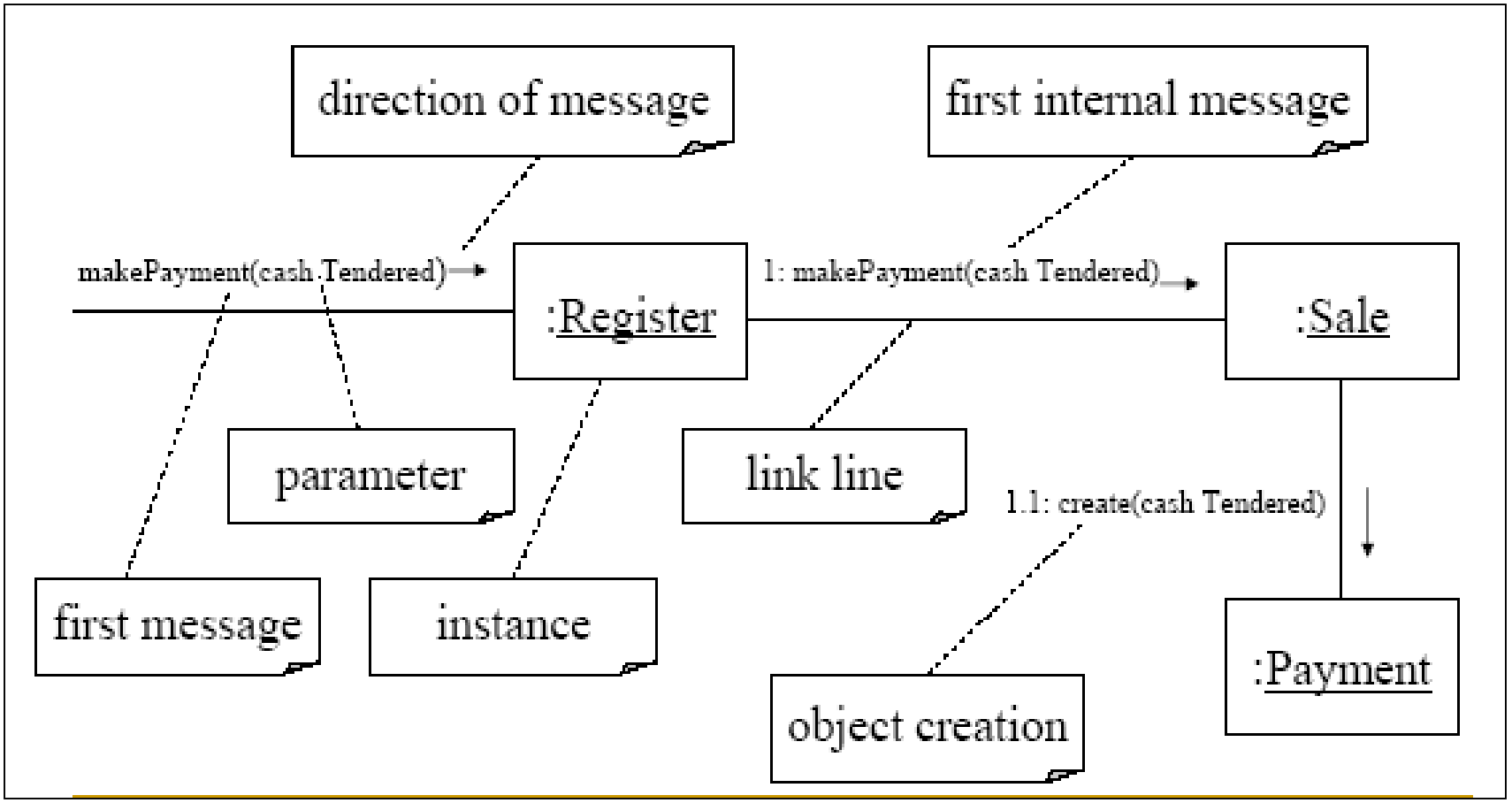
Sanjay Joshi

BITS Pilani

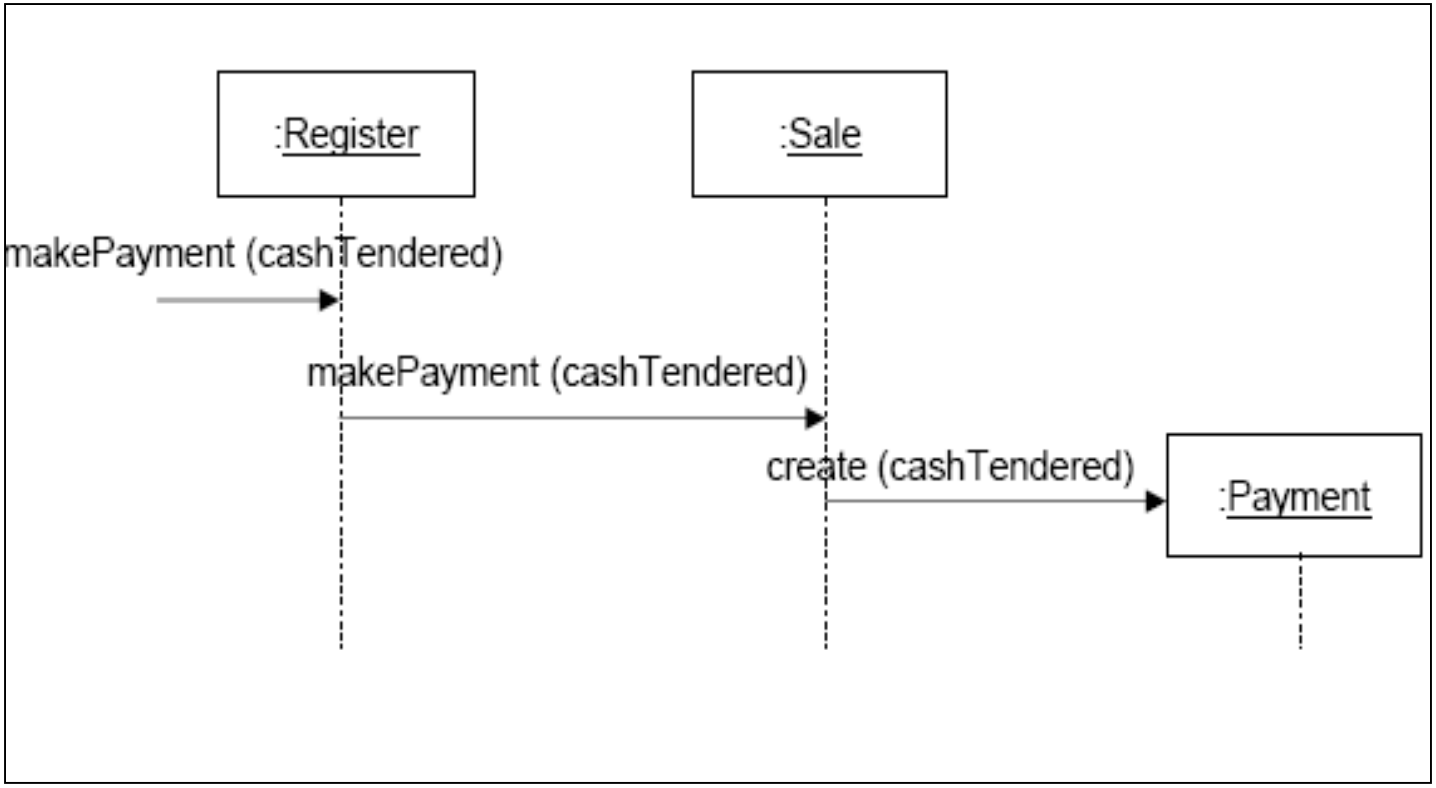
Pilani | Dubai | Goa | Hyderabad

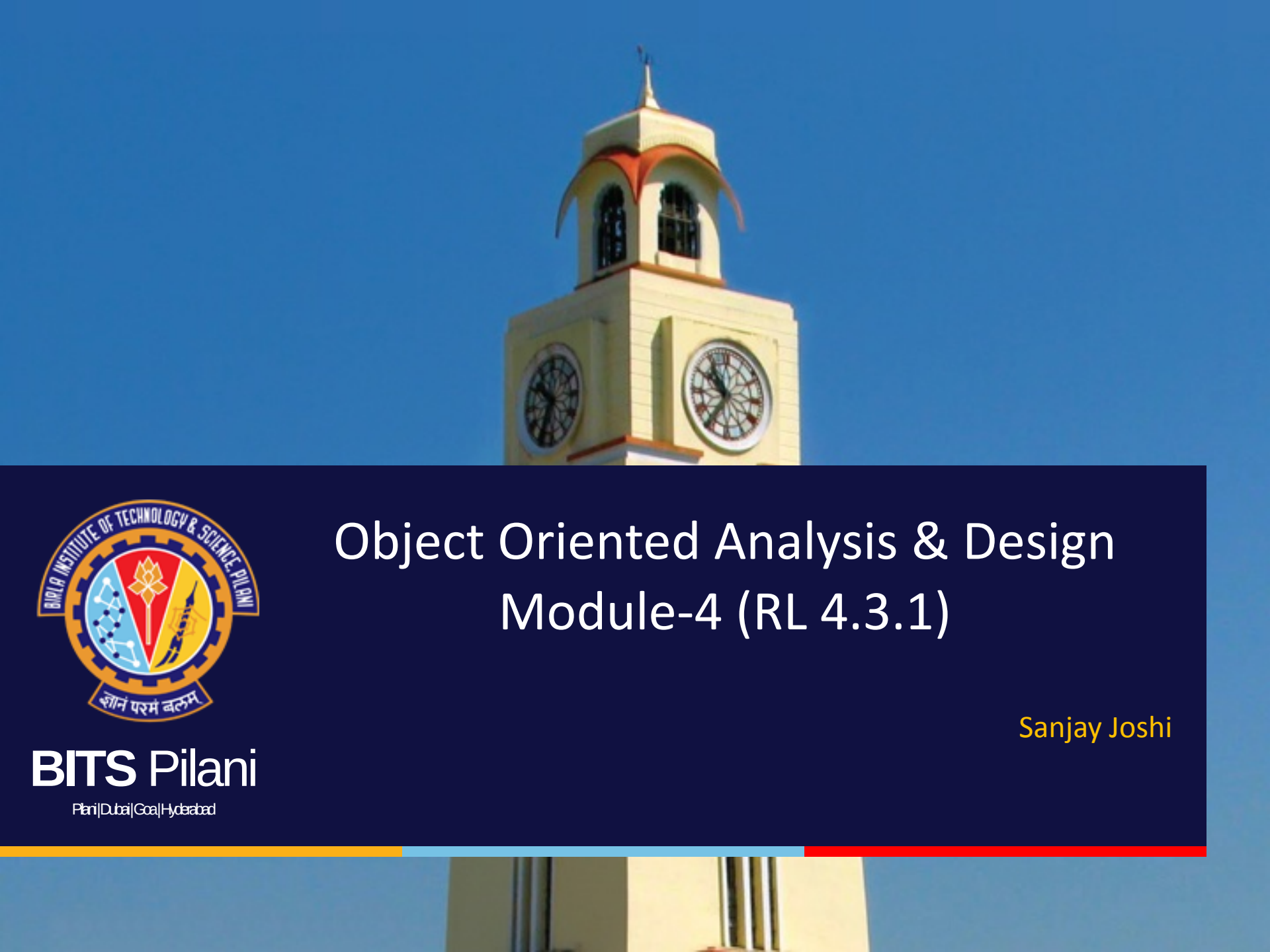
Drawing Interaction Diagrams for PoS

Collaboration Diagram: makePayment



Sequence Diagram : makePayment





Object Oriented Analysis & Design
Module-4 (RL 4.3.1)

Sanjay Joshi

BITS Pilani
Pilani|Dubai|Goa|Hyderabad

Introduction to State Transition Diagram

Introduction to State Transition Diagram

- A state diagram (also state transition diagram) illustrates the events and the states of things.

- It can be drawn for System, Subsystem or an Object in the System.

Events, States and Transitions

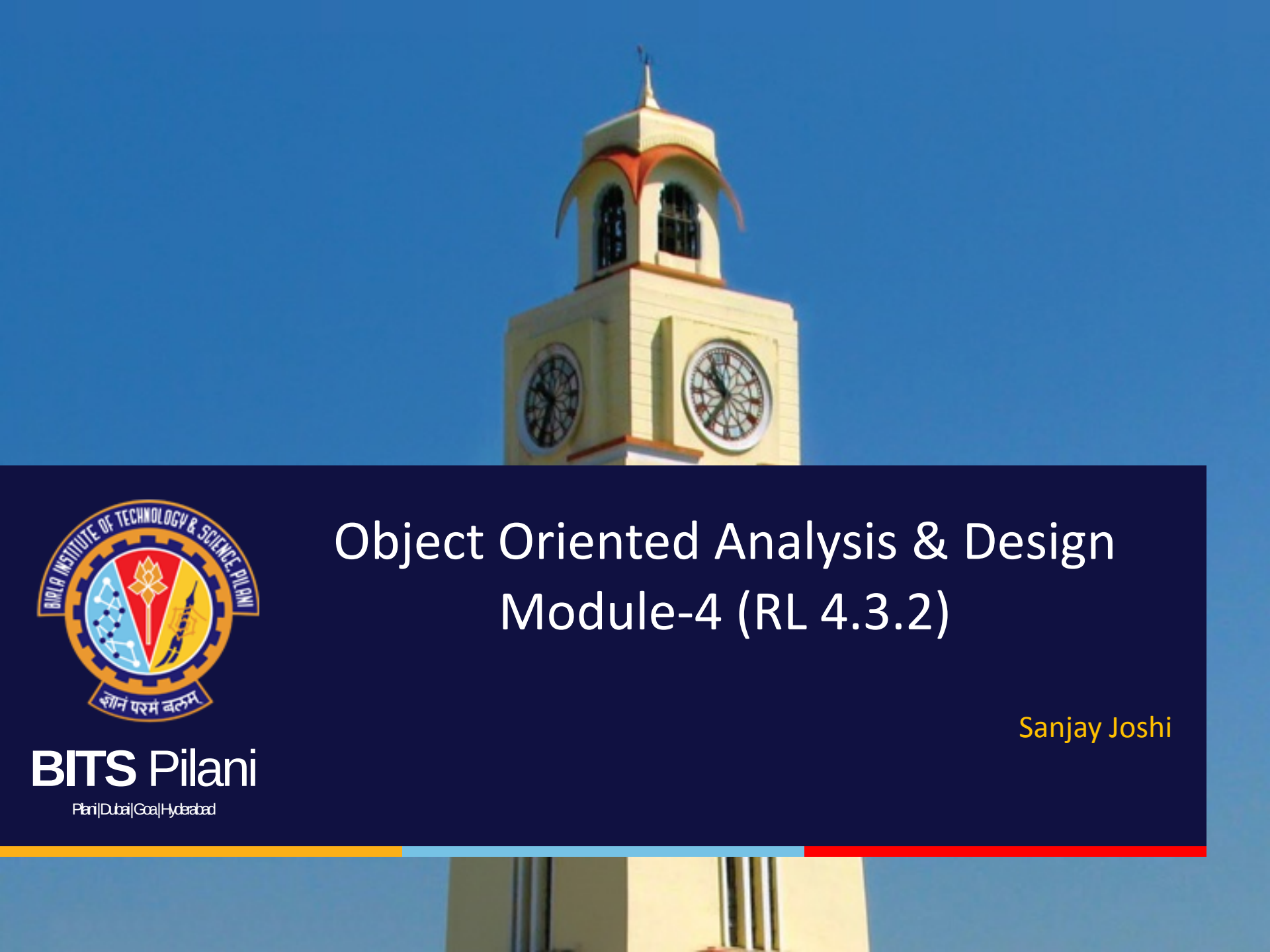
- An event is a trigger, or occurrence.
  - e.g. a telephone receiver is taken off the hook.
- A state is the condition of an entity (object) at a moment in time - the time between events.
  - e.g. a telephone is in the state of being idle after the receiver is placed on the hook and until it is taken off the hook.

Events, States and Transitions

- A transition is a relationship between two states; It indicates that when an event occurs, the object moves from the prior state to the subsequent state.
  - e.g. when an event off the hook occurs, transition the telephone from the idle state to active state.

State Transition Diagrams

- A statechart diagram shows the life-cycle of an object; what events it experiences, its transitions and the states it is in between events.
- A state diagram need not illustrate every possible event; if an event arises that is not represented in the diagram, the event is ignored as far as the state diagram is concerned.
- Thus, we can create a state diagram which describes the life-cycle of an object at any simple or complex level of detail, depending on our needs.



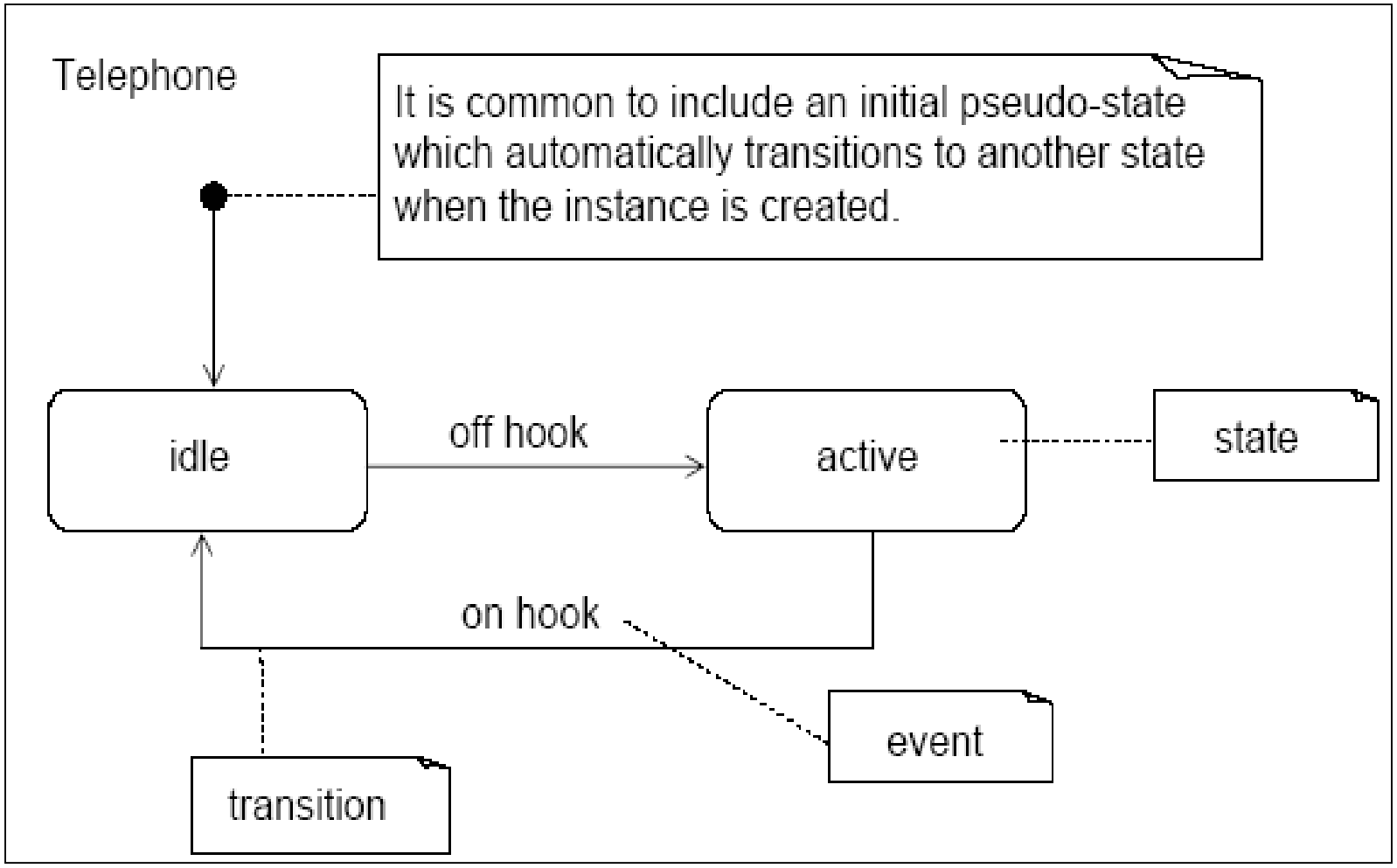
Object Oriented Analysis & Design
Module-4 (RL 4.3.2)

Sanjay Joshi

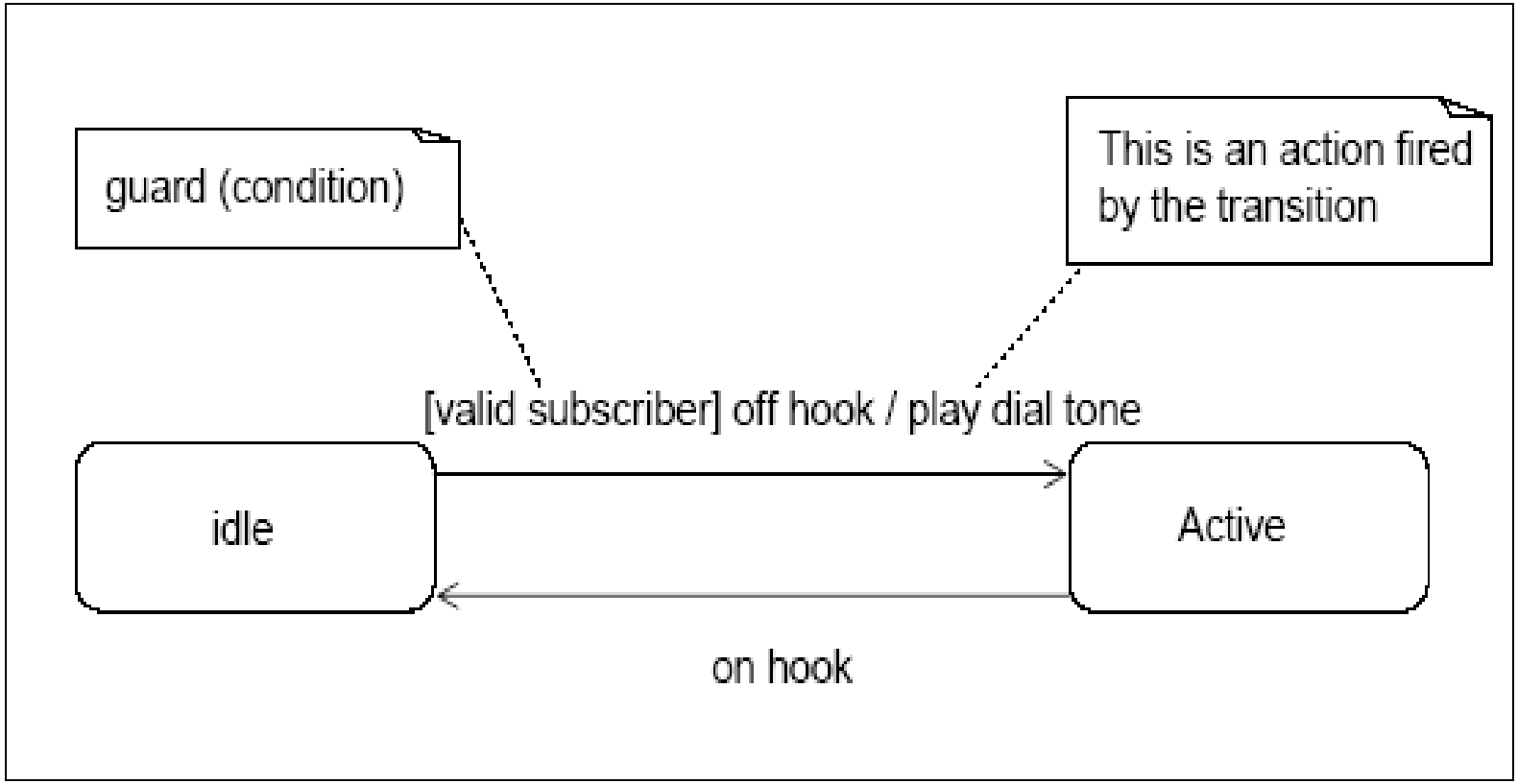
BITS Pilani
Pilani|Dubai|Goa|Hyderabad

Representing State Transition Diagram in UML

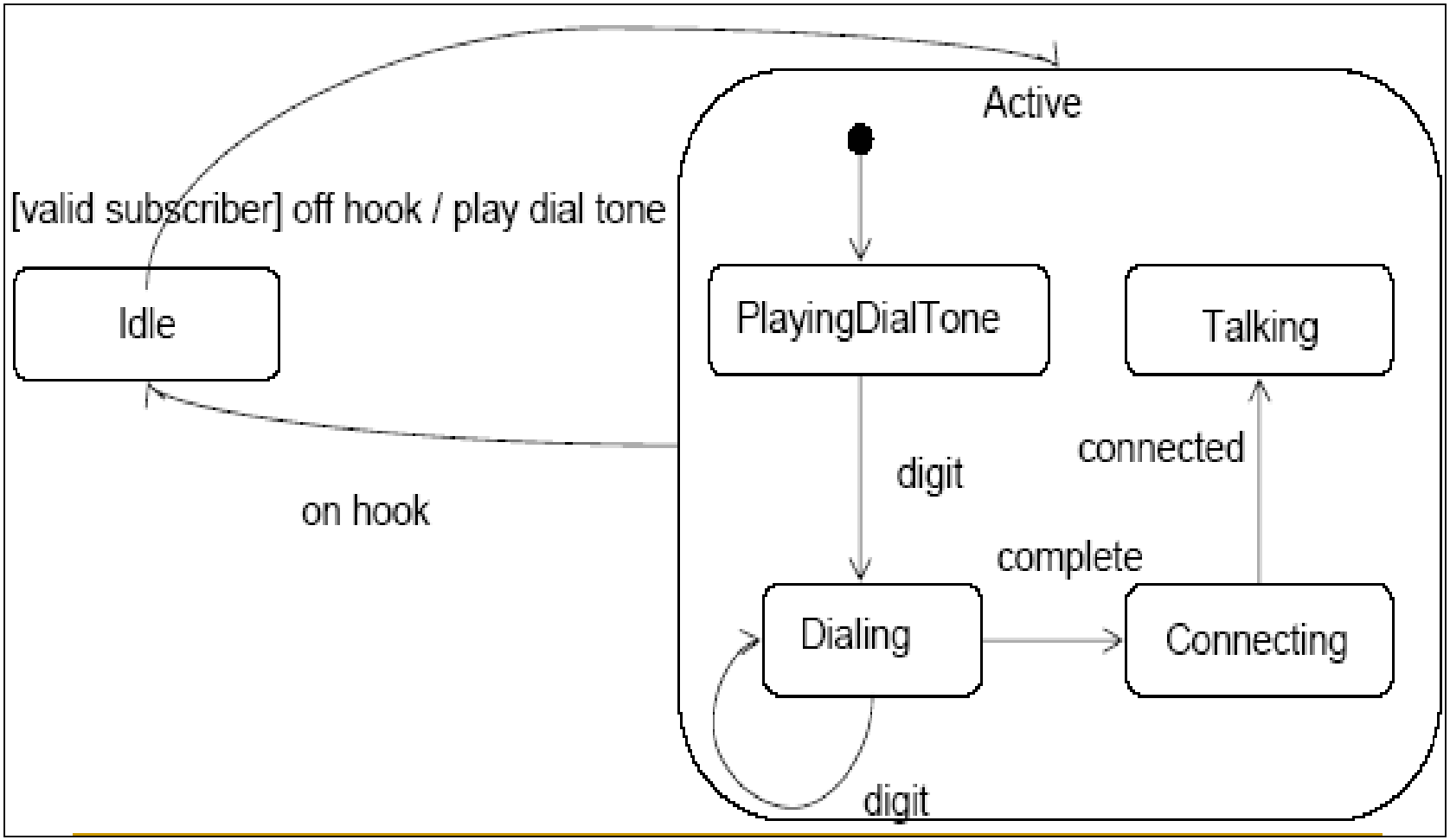
State Transition Diagram in UML



Additional State Transition Diagram Notation

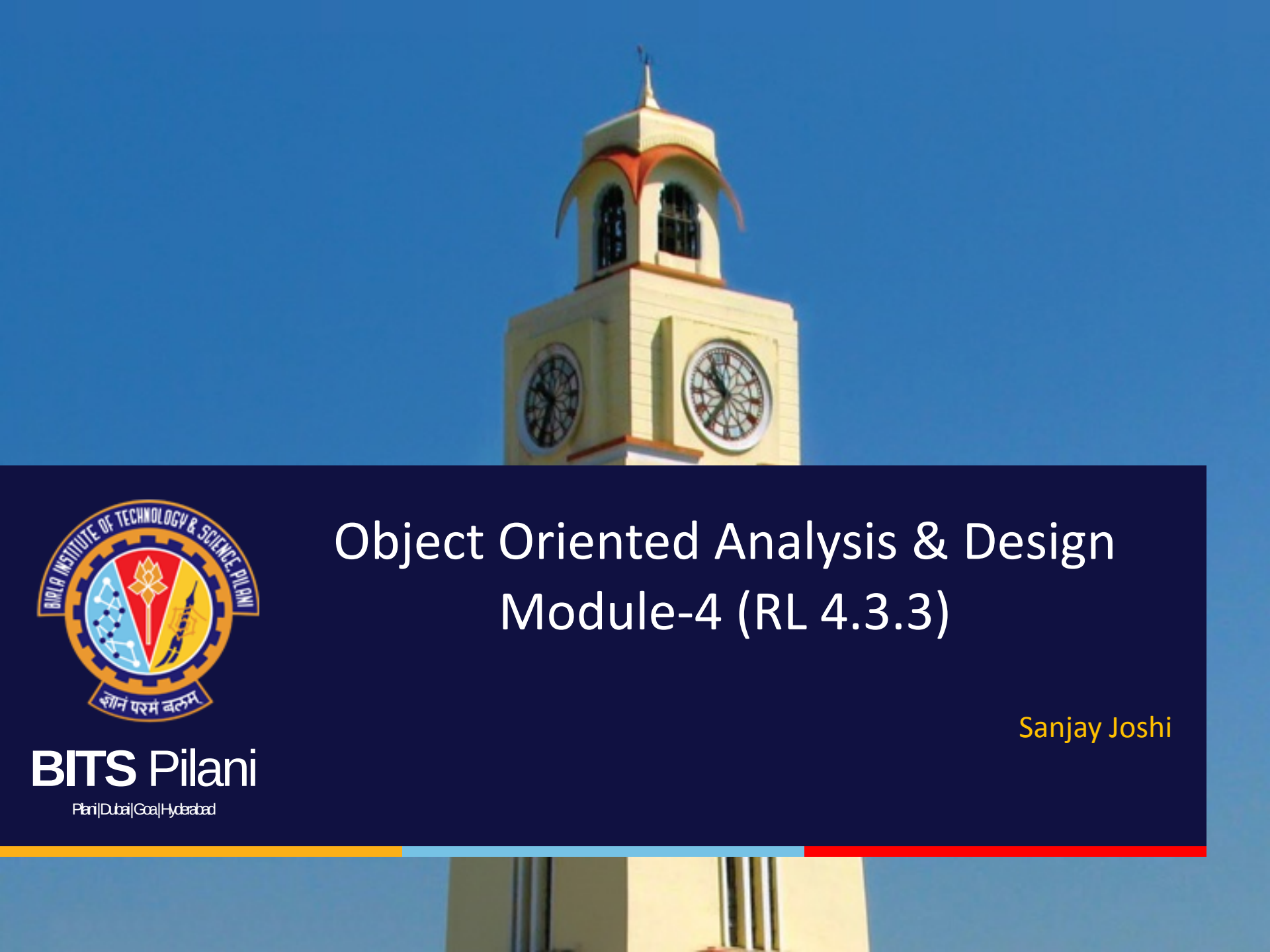


Additional State Transition Diagram Notation



Additional State Transition Diagram Notation

- A state allows nesting to contain substates. A substate inherits the transitions of its superstate (the enclosing state).
- Within the Active state, and no matter what substate the object is in, if the on hook event occurs, a transition to the idle state occurs.



Object Oriented Analysis & Design
Module-4 (RL 4.3.3)

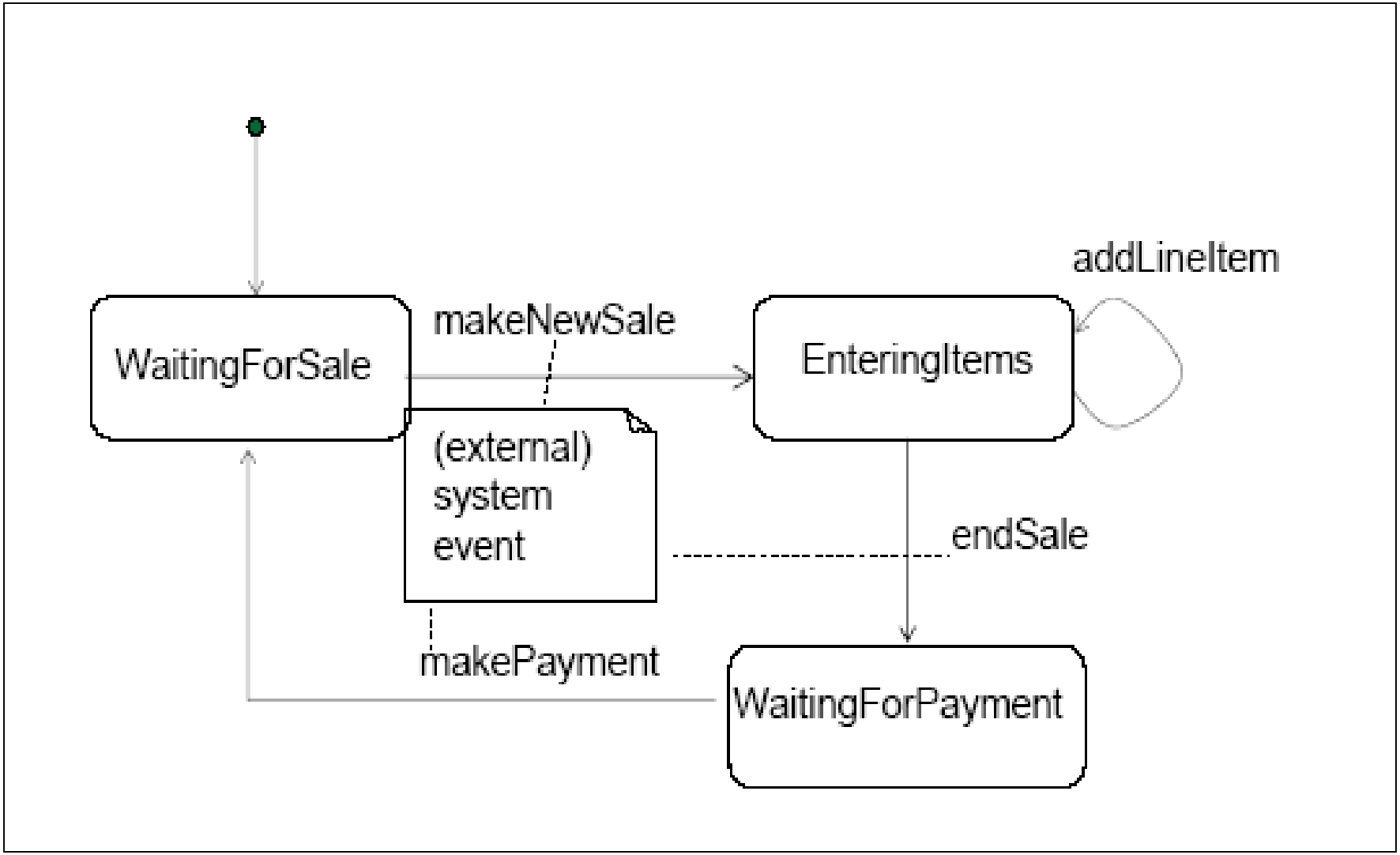
Sanjay Joshi

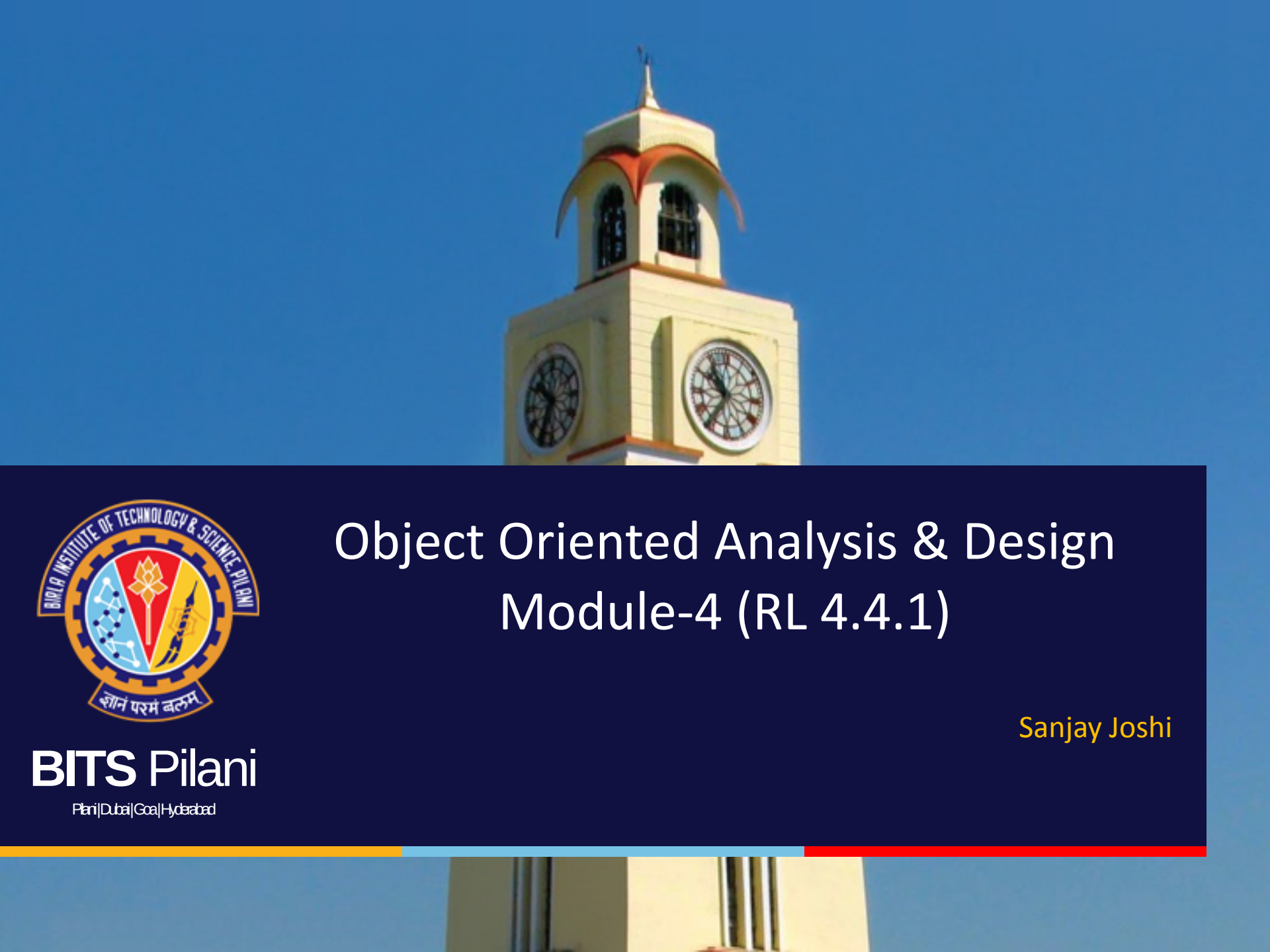
BITS Pilani

Pilani|Dubai|Goa|Hyderabad

Drawing State Transition diagram for PoS

State Transition Diagram for PoS





Object Oriented Analysis & Design
Module-4 (RL 4.4.1)

Sanjay Joshi

BITS Pilani
Pilani|Dubai|Goa|Hyderabad

innovate achieve lead

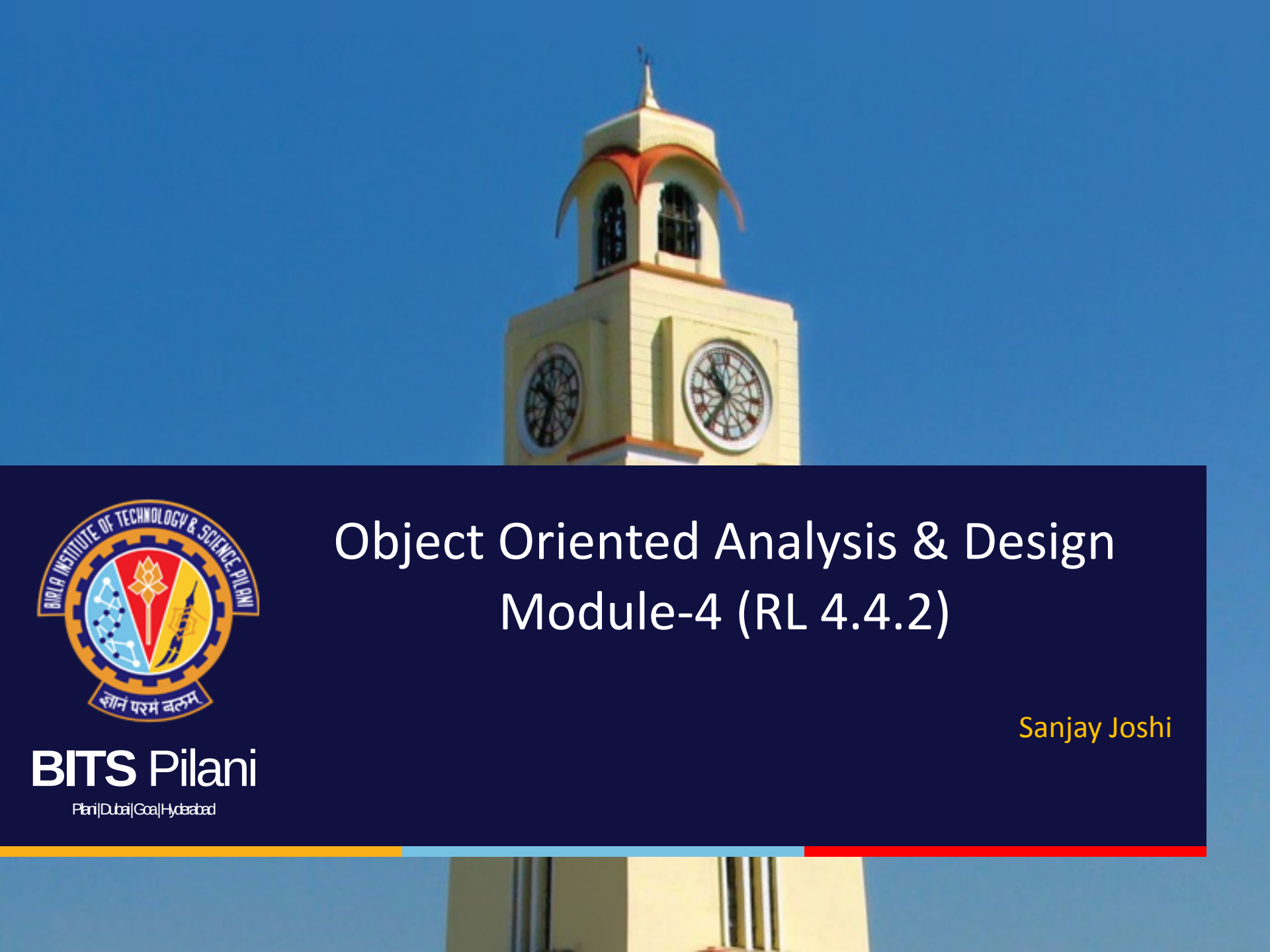
BITS Pilani

Pilani|Dubai|Goa|Hyderabad

What is Activity Diagram?

What is Activity Diagram

- They show the sequence of flow activities involved in a process.
- Used to model dynamic aspects of a system
- Like Flowchart showing flow of control from activity to activity
- Are used when you have multiple activities going on at the same time.



Object Oriented Analysis & Design
Module-4 (RL 4.4.2)

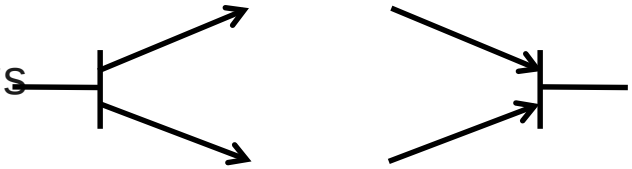
Sanjay Joshi

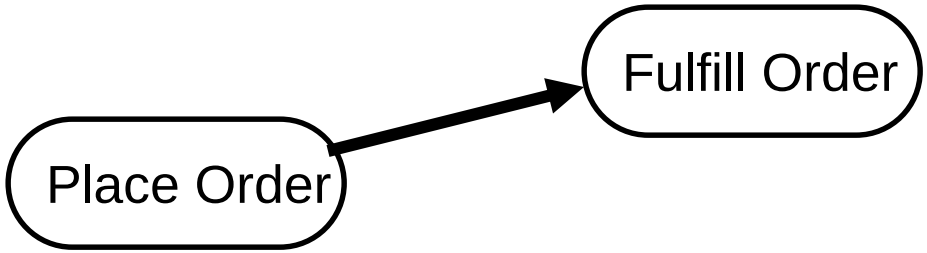
BITS Pilani
Pilani|Dubai|Goa|Hyderabad

Representing Activity Diagram in UML

Activity Diagram in UML

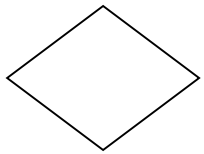
- Initial state
- Final state
- Fork and join
  - model concurrent flows
- Activities
  - Step in overall Process
- Transitions
  - Triggered by end of previous activity and initiates next activity



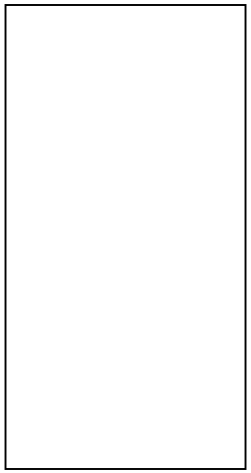


Activity Diagram in UML

- Branching: specifies alternate paths taken based Boolean expression



- swimlanes: Allow to partition the activity states into groups, each group representing the business organization responsible for those activities



Activity Diagram in UML