About This Course

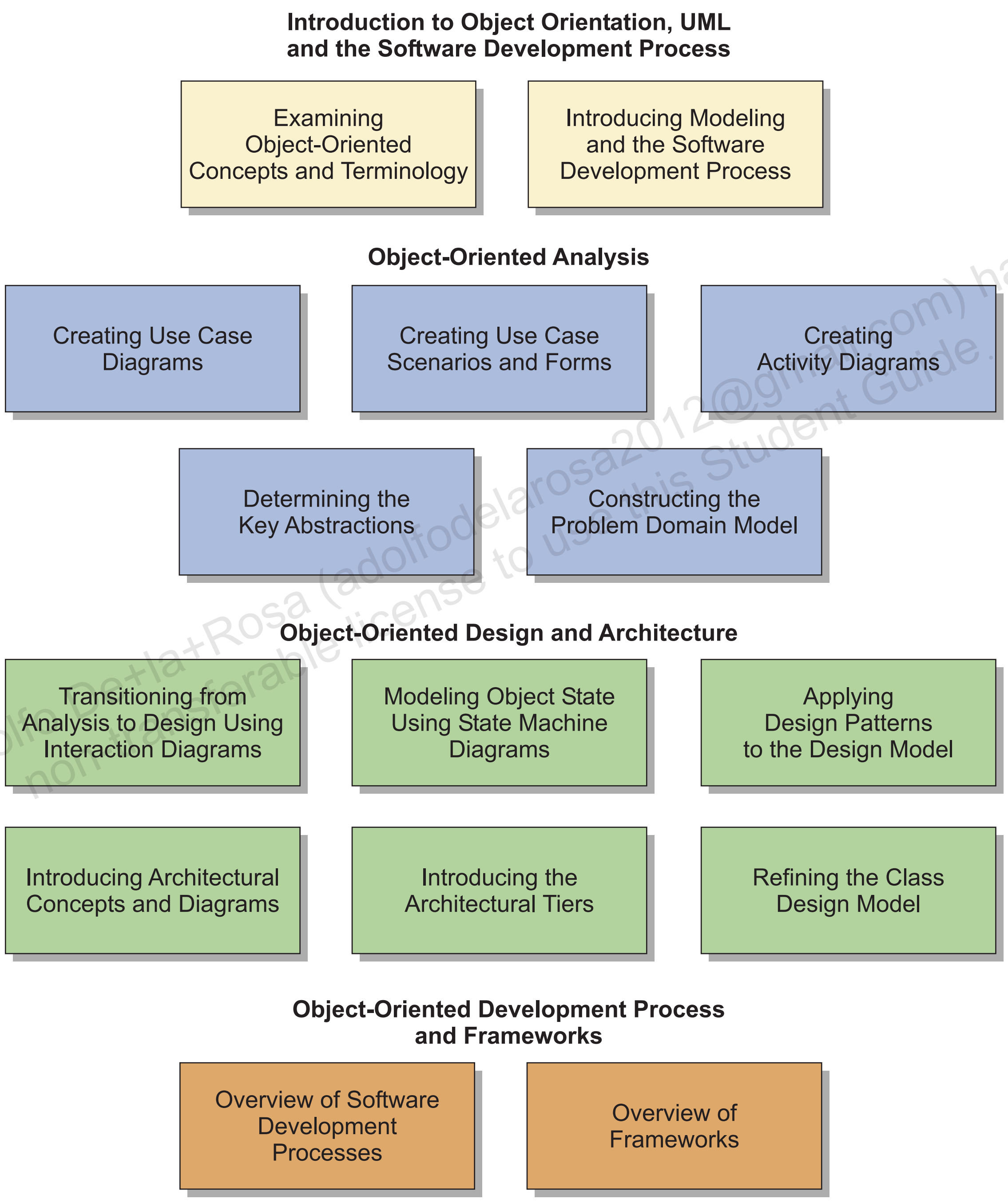
Course Goals

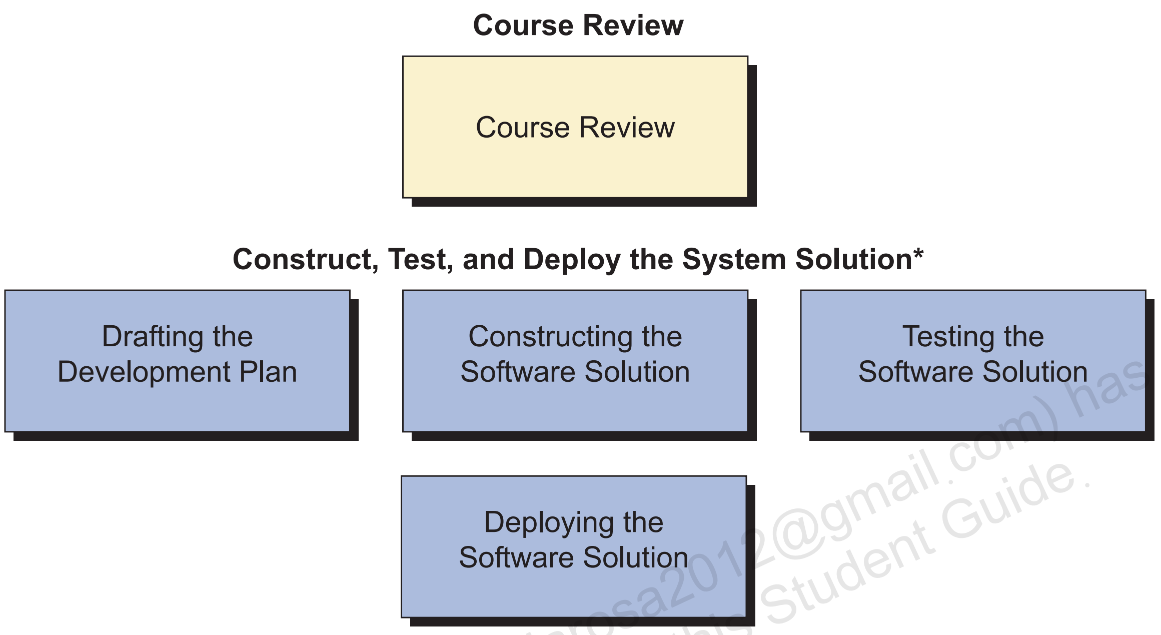
Upon completion of this course, you should be able to:

- Apply object-oriented (OO) technologies to meet your software requirements
- Create proportionate and appropriate Unified Modeling Language (UML) models or text models at each stage in the software development process
- Analyze system requirements using use cases to determine the analysis (business domain) model
- Create analysis models that capture the business requirements of the system
- Explain how to fit the design components into the chosen architecture
- Create design (solution) models that support requirements of the system
- Apply the patterns and principles used in analysis and design
- Describe common Object-Oriented Software Development (OOSD) processes

Course Map

The following course map enables you to see what you have accomplished and where you are going in reference to the course goals.


Introduction to Object Orientation, UML and the Software Development Process
Examining Object-Oriented Concepts and Terminology
Introducing Modeling and the Software Development Process
Object-Oriented Analysis
Creating Use Case Diagrams
Creating Use Case Scenarios and Forms
Creating Activity Diagrams
Determining the Key Abstractions
Constructing the Problem Domain Model
Object-Oriented Design and Architecture
Transitioning from Analysis to Design Using Interaction Diagrams
Modeling Object State Using State Machine Diagrams
Applying Design Patterns to the Design Model
Introducing Architectural Concepts and Diagrams
Introducing the Architectural Tiers
Refining the Class Design Model
Object-Oriented Development Process and Frameworks
Overview of Software Development Processes
Overview of Frameworks


Course Review
Course Review
Construct, Test, and Deploy the System Solution*
Drafting the Development Plan
Constructing the Software Solution
Testing the Software Solution
Deploying the Software Solution

*These modules are appendices.

Topics Not Covered

This course does not cover the following topics. Many of these topics are covered in other courses offered by Sun Services:

- Fundamental Java technology – Covered in SL-275-SE6: *Java™ Programming Language*
- Enterprise edition Java technology – Covered in FJ-310-EE5: *Developing Applications for the Java™ EE Platform*

Refer to the Sun Services catalog for specific information and registration.

How Prepared Are You?

To be sure you are prepared to take this course, can you answer yes to the following questions?

- Do you have a general understanding of a programming language?
- Do you have an understanding of the fundamentals of the software system development process?

Introductions

Now that you have been introduced to the course, introduce yourself to the other students and the instructor, addressing the following items:

- Name
- Company affiliation
- Title, function, and job responsibility
- Experience related to requirements gathering and analysis
- Experience related to software architecture and design
- Experience related to using a software development process
- Experience related to modeling notations, such as Object Modeling Technique (OMT) or Unified Modeling Language (UML)
- Reasons for enrolling in this course
- Expectations for this course

How to Use Course Materials

To enable you to succeed in this course, these course materials contain a learning module that is composed of the following components:

- Goals – You should be able to accomplish the goals after finishing this course and meeting all of its objectives.
- Objectives – You should be able to accomplish the objectives after completing a portion of instructional content. Objectives support goals and can support other higher-level objectives.
- Lecture – The instructor presents information specific to the objective of the module. This information helps you learn the knowledge and skills necessary to succeed with the activities.
- Activities – The activities take on various forms, such as an exercise, self-check, discussion, and demonstration. Activities help you facilitate the mastery of an objective. The majority of the activities are designed to be performed in small groups.
- Visual aids – The instructor might use several visual aids to convey a concept, such as a process, in a visual form. Visual aids commonly contain graphics, animation, and video.

Conventions

The following conventions are used in this course to represent various training elements and alternative learning resources.

Icons

Additional resources – Indicates other references that provide additional information on the topics described in the module.

Discussion – Indicates a small-group or class discussion on the current topic is recommended at this time.

Note – Indicates additional information that can help students but is not crucial to their understanding of the concept being described. Students should be able to understand the concept or complete the task without this information. Examples of notational information include keyword shortcuts and minor system adjustments.

Typographical Conventions

`Courier` is used for the names of commands, files, directories, programming code, and on-screen computer output; for example:

> Use `ls -al` to list all files.
> `system% You have mail`.

`Courier` is also used to indicate programming constructs, such as class names, methods, and keywords; for example:

> The `getServletInfo` method is used to get author information.
> The `java.awt.Dialog` class contains `Dialog` constructor.

`Courier bold` is used for characters and numbers that you type; for example:

To list the files in this directory, type:

```
# ls
```

`Courier bold` is also used for each line of programming code that is referenced in a textual description; for example:

```
1 import java.io.*;
2 import javax.servlet.*;
3 import javax.servlet.http.*;
```

Notice the `javax.servlet` interface is imported to allow access to its life cycle methods (Line 2).

`Courier italics` is used for variables and command-line placeholders that are replaced with a real name or value; for example:

To delete a file, use the `rm` *`filename`* command.

`Courier italic bold` is used to represent variables whose values are to be entered by the student as part of an activity; for example:

Type **`chmod a+rwx`** ***`filename`*** to grant read, write, and execute rights for *`filename`* to world, group, and users.

Palatino italics is used for book titles, new words or terms, or words that you want to emphasize; for example:

Read Chapter 6 in the *User's Guide*.
These are called *class* options.

Additional Conventions

Java™ programming language examples use the following additional conventions:

- Method names are not followed with parentheses unless a formal or actual parameter list is shown; for example:

  "The `doIt` method..." refers to any method called `doIt`.

  "The `doIt()` method..." refers to a method called `doIt` that takes no arguments.

- Line breaks occur only where there are separations (commas), conjunctions (operators), or white space in the code. Broken code is indented four spaces under the starting code.

- If a command used in the Solaris™ Operating Environment is different from a command used in the Microsoft Windows platform, both commands are shown; for example:

  If working in the Solaris Operating Environment

  ```
  > cd SERVER_ROOT/bin
  ```

  If working in Microsoft Windows

  ```
  C:> cd SERVER_ROOT\bin
  ```