Object Oriented Thought Process

1. Introduction to Object-Oriented Concepts

- Procedural Versus OO Programming
- Moving from Procedural to Object-Oriented Development
- Procedural Programming
- OO Programming
- What Exactly Is an Object?
- Object Data
- Object Behaviors
- What Exactly Is a Class?
- Classes Are Object Templates
- Attributes
- Methods
- Messages
- Using UML to Model a Class Diagram
- Encapsulation and Data Hiding
- Interfaces
- Implementations
- A Model of the Interface/Implementation Paradigm
- Inheritance
- Super classes and Subclasses
- Abstraction
- Is-a Relationships
- Polymorphism
- Composition
- Abstraction
- Has-a Relationships
- Conclusion

2. How to Think in Terms of Objects

- Knowing the Difference Between the Interface and the Implementation
- The Interface
- The Implementation
- An Interface/Implementation Example
- Using Abstract Thinking When Designing Interfaces
- Giving the User the Minimal Interface Possible
- Determining the Users
- Object Behavior
3. Advanced Object-Oriented Concepts

- Constructors
- The Default Constructor
- When Is a Constructor Called?
- What’s Inside a Constructor?
- The Default Constructor
- Using Multiple Constructors
- The Design of Constructors
- Error Handling
- Ignoring the Problem
- Checking for Problems and Aborting the Application
- Checking for Problems and Attempting to Recover
- Throwing an Exception
- The Concept of Scope
- Local Attributes
- Object Attributes
- Class Attributes
- Operator Overloading
- Multiple Inheritance
- Object Operations
- Conclusion

4. The Anatomy of a Class

- The Name of the Class
- Comments
- Attributes
- Constructors
- Accessors
- Public Interface Methods
- Private Implementation Methods
- Conclusion

5. Class Design Guidelines

- Modeling Real World Systems
- Identifying the Public Interfaces
- The Minimum Public Interface
- Hiding the Implementation
- Designing Robust Constructors (and Perhaps Destructors)
• Designing Error Handling into a Class
• Documenting a Class and Using Comments
• Building Objects with the Intent to Cooperate
• Designing with Reuse in Mind
• Documenting a Class and Using Comments
• Designing with Extensibility in Mind
• Making Names Descriptive
• Abstracting Out Non-portable Code
• Providing a Way to Copy and Compare Objects
• Keeping the Scope as Small as Possible
• A Class Should Be Responsible for Itself
• Designing with Maintainability in Mind
• Using Iteration
• Testing the Interface
• Using Object Persistence
• Serializing and Marshaling Objects
• Conclusion

6. Designing with Objects

• Design Guidelines
• Performing the Proper Analysis
• Developing a Statement of Work

> 

• Gathering the Requirements
• Developing a Prototype of the User Interface
• Identifying the Classes
• Determining the Responsibilities of Each Class
• Determining How the Classes Collaborate with Each Other
• Creating a Class Model to Describe the System
• Case Study
• Identifying the Classes’ Responsibilities
• UML Use-Cases: Identifying the Collaborations
• UML Class Diagrams: The Object Model
• Prototyping the User Interface
• Conclusion

7. Mastering Inheritance and Composition

• Reusing Objects
• Inheritance
• Generalization and Specialization
• Design Decisions
• Composition
- Representing Composition with UML
- Why Encapsulation Is Fundamental to OO
- How Inheritance Weakens Encapsulation
- A Detailed Example of Polymorphism
- Object Responsibility
- Conclusion

8. Frameworks and Reuse: Designing with Interfaces and Abstract Classes

- Code: To Reuse or Not to Reuse?
- What Is a Framework?
- What Is a Contract?
- Abstract Classes
- Interfaces
- The Compiler Proof
- Making a Contract
- System Plug-in-Points
- An E-Business Example
- The Non-Reuse Approach
- An E-Business Solution
- The UML Object Model
- Conclusion

9. Building Objects

- Composition Relationships
- Building in Phases
- Types of Composition
- Aggregations
- Associations
- Using Associations and Aggregations Together
- Avoiding Dependencies
- Cardinality
- Multiple Object Associations
- Optional Associations
- Conclusion

10. Creating Object Models with UML

- What Is UML?
- The Structure of a Class Diagram
- Attributes and Methods
- Access Designations
- Inheritance
- Interfaces
• Composition
• Aggregations
• Associations
• Cardinality
• Conclusion

11. Objects and Portable Data: XML

• Portable Data
• The Extensible Markup Language (XML)
• XML Versus HTML
• XML and Object-Oriented Languages
• Sharing Data Between Two Companies
• Validating the Document with the Document Type Definition (DTD)
• Integrating the DTD into the XML Document
• Using Cascading Style Sheets
• Conclusion

12. Persistent Objects: Serialization and Relational

• Databases
• Persistent Objects Basics
• Saving the Object to a Flat File
• Serializing a File
• Implementation and Interface Revisited
• What About the Methods?
• Using XML in the Serialization Process
• Writing to a Relational Database
• Accessing a Relational Database
• Loading the Driver
• Making the Connection
• The SQL Statements
• Conclusion

13. Objects and the Internet

• Evolution of Distributed Computing
• Object-Based Scripting Languages
• A JavaScript Validation Example
• Objects in a Web Page
• JavaScript Objects
• Web Page Controls
• Sound Players
• Movie Players
• Flash
14. Objects and Client/Server Applications

- Client/Server Approaches
- Proprietary Approach
- Serialized Object Code
- Client Code
- Server Code
- Running the Proprietary Client/Server Example
- Nonproprietary Approach
- Object Definition Code
- Client Code
- Server Code
- Running the Nonproprietary Client/Server Example
- Conclusion