



Matt Weisfeld

Fifth Edition

# The Object-Oriented Thought Process



# The Object-Oriented Thought Process

---

Fifth Edition

# Object-Oriented Thought Process, The

## Table of Contents

Cover

Half Title

Title Page

Copyright Page

Dedication

Contents at a Glance

Table of Contents

Introduction

- This Books Scope

- Whats New in the Fifth Edition

- The Intended Audience

- The Books Approach

- Source Code Used in This Book

1 Introduction to Object-Oriented Concepts

- The Fundamental Concepts

- Objects and Legacy Systems

- Procedural Versus OO Programming

- Moving from Procedural to Object-Oriented Development

  - Procedural Programming

  - OO Programming

- What Exactly Is an Object?

  - Object Data

# Table of Contents

Object Behaviors

## What Exactly Is a Class?

Creating Objects

Attributes

Methods

Messages

## Using Class Diagrams as a Visual Tool

## Encapsulation and Data Hiding

Interfaces

Implementations

A Real-World Example of the Interface/Implementation Paradigm

A Model of the Interface/Implementation Paradigm

## Inheritance

Superclasses 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

# **Table of Contents**

## Providing the Absolute Minimal User Interface Possible

- Determining the Users

- Object Behavior

- Environmental Constraints

- Identifying the Public Interfaces

- Identifying the Implementation

Conclusion

References

## 3 More Object-Oriented Concepts

### Constructors

- When Is a Constructor Called?

- Whats 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 Importance of Scope

- Local Attributes

- Object Attributes

- Class Attributes

### Operator Overloading

### Multiple Inheritance

### Object Operations

### Conclusion

# **Table of Contents**

References

## **4 The Anatomy of a Class**

The Name of the Class

Comments

Attributes

Constructors

Accessors

Public Interface Methods

Private Implementation Methods

Conclusion

References

## **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

Designing with Extensibility in Mind

Making Names Descriptive

Abstracting Out Nonportable Code

Providing a Way to Copy and Compare Objects

Keeping the Scope as Small as Possible

Designing with Maintainability in Mind

# **Table of Contents**

Using Iteration in the Development Process

Testing the Interface

Using Object Persistence

Serializing and Marshaling Objects

Conclusion

References

## **6 Designing with Objects**

Design Guidelines

Performing the Proper Analysis

Developing a Statement of Work

Gathering the Requirements

Developing a System Prototype

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

Prototyping the User Interface in Code

Object Wrappers

Structured Code

Wrapping Structured Code

Wrapping Nonportable Code

Wrapping Existing Classes

Conclusion

References

## **7 Mastering Inheritance and Composition**

Reusing Objects

Inheritance

Generalization and Specialization

# Table of Contents

Design Decisions

## Composition

Representing Composition with UML

## Why Encapsulation Is Fundamental to OO

How Inheritance Weakens Encapsulation

A Detailed Example of Polymorphism

Object Responsibility

Abstract Classes, Virtual Methods, and Protocols

## Conclusion

## References

## 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

Tying It All Together

The Compiler Proof

Making a Contract

System Plug-in Points

## An E-Business Example

An E-Business Problem

The Non-Reuse Approach

An E-Business Solution

The UML Object Model

## Conclusion

## References



# **Table of Contents**

## **9 Building Objects and Object-Oriented Design**

Composition Relationships

Building in Phases

Types of Composition

Aggregations

Associations

Using Associations and Aggregations Together

Avoiding Dependencies

Cardinality

Multiple Object Associations

Optional Associations

Tying It All Together: An Example

Conclusion

References

## **10 Design Patterns**

Why Design Patterns?

Smalltalks Model/View/Controller

Types of Design Patterns

Creational Patterns

Structural Patterns

Behavioral Patterns

Antipatterns

Conclusion

References

## **11 Avoiding Dependencies and Highly Coupled Classes**

Composition versus Inheritance and Dependency Injection

1) Inheritance

# **Table of Contents**

2) Composition

Dependency Injection

Conclusion

References

## **12 The SOLID Principles of Object-Oriented Design**

The SOLID Principles of Object-Oriented Design

1) SRP: Single Responsibility Principle

2) OCP: Open/Close Principle

3) LSP: Liskov Substitution Principle

4) IPS: Interface Segregation Principle

5) DIP: Dependency Inversion Principle

Conclusion

References

Index