1. Record Nr. UNINA9910150176303321

Autore	Koseoglu Kerem
Titolo	Design Patterns in ABAP Objects
Pubbl/distr/stampa	Boston : , : Rheinwerk Publishing Inc., , 2016 ©2017
ISBN	9781493214655 1493214659
Descrizione fisica	1 online resource (504 pages)
Disciplina	005.1
Soggetti	ABAP Objects (Computer program language) Software patterns
Lingua di pubblicazione	Inglese
Formato	Materiale a stampa
Livello bibliografico	Monografia
Nota di contenuto	Intro -- Dear Reader -- Notes on Usage -- Table of Contents -- Preface -- Design Pattern Categories -- How to Learn -- Part I Architectural Design Patterns -- 1 MVC -- 1.1 Case Study: Read, Process, Display, and Post -- 1.2 Passing Select Options -- 1.3 Distributing Application Logic -- 1.4 Related Patterns -- 1.5 Summary -- Part II Creational Design Patterns -- 2 Abstract Factory -- 2.1 Case Study: Log Analysis -- 2.2 Related Patterns -- 2.3 Summary -- 3 Builder -- 3.1 Case Study: Jobs for Text Files -- 3.2 When to Use -- 3.3 Privacy -- 3.4 Summary -- 4 Factory -- 4.1 Case Study: FI Documents for Parties -- 4.2 Advantages -- 4.3 Related Patterns -- 4.4 Summary -- 5 Lazy Initialization -- 5.1 Case Study: Logging Errors -- 5.2 Advantages -- 5.3 Related Patterns -- 5.4 Summary -- 6 Multiton -- 6.1 Case Study: Vendor Balance -- 6.2 When to Use -- 6.3 When to Avoid -- 6.4 State Modification -- 6.5 Summary -- 7 Prototype -- 7.1 Case Study: Item Clone -- 7.2 Changing Class Properties -- 7.3 Clone Operations -- 7.4 Related Patterns -- 7.5 Summary -- 8 Singleton -- 8.1 Case Study: Subcomponents -- 8.2

  Advantages and Disadvantages -- 8.3   Related Patterns -- 8.4   Summary -- Part III   Structural Design Patterns -- 9   Adapter -- 9.1   Case Study: Project Management Tools -- 9.2   Glue Code -- 9.3   Two-Way Adapters -- 9.4   Legacy Classes -- 9.5   Summary -- 10   Bridge -- 10.1   Case Study: Messaging Framework -- 10.2   Advantages -- 10.3   Summary -- 11   Composite -- 11.1   Recursive Programming: A Refresher -- 11.2   Case Study: HR Positions -- 11.3   Advantages -- 11.4   Disadvantages -- 11.5   When to Use -- 11.6   Related Patterns -- 11.7   Summary -- 12   Data Access Object -- 12.1   Case Study: Potential Customers -- 12.2   Redundant Code Prevention.
12.3   Related Patterns -- 12.4   Summary -- 13   Decorator -- 13.1   Case Study: User Exit -- 13.2   Advantages and Challenges -- 13.3   Related Patterns -- 13.4   Summary -- 14   Facade -- 14.1   Case Study: Bonus Calculation -- 14.2   When and Where to Use -- 14.3   Related Patterns -- 14.4   Summary -- 15   Flyweight -- 15.1   Case Study: Negative Stock Forecast -- 15.2   Disadvantages -- 15.3   When to Use -- 15.4   Related Patterns -- 15.5   Summary -- 16   Property Container -- 16.1   Case Study: Enhancing an SAP Program -- 16.2   Static vs. Instance Containers -- 16.3   Sharing Variables -- 16.4   Variable Uniqueness -- 16.5   Related Patterns -- 16.6   Summary -- 17   Proxy -- 17.1   Case Study: Sensitive Salary Information -- 17.2   When to Use -- 17.3   Related Patterns -- 17.4   Summary -- 18   Chain of Responsibility -- 18.1   Case Study: Purchase Order Approver Determination -- 18.2   Risks -- 18.3   Related Patterns -- 18.4   Summary -- Part IV   Behavioral Design Patterns -- 19   Command -- 19.1   Case Study: SD Documents -- 19.2   When to Use or Avoid -- 19.3   Related Patterns -- 19.4   Summary -- 20   Mediator -- 20.1   Case Study: Stock Movement Simulation -- 20.2   When to Use -- 20.3   Disadvantages -- 20.4   Summary -- 21   Memento -- 21.1   Case Study: Budget Planning -- 21.2   Risks -- 21.3   Redo -- 21.4   Summary -- 22   Observer -- 22.1   Case Study: Target Sales Values -- 22.2   Advantages -- 22.3   Disadvantages -- 22.4   Multiple Subjects -- 22.5   Related Patterns -- 22.6   Summary -- 23   Servant -- 23.1   Case Study: Address Builder -- 23.2   Extensions -- 23.3   Related Patterns -- 23.4   Summary -- 24   State -- 24.1   Case Study: Authorization-Based Class Behavior -- 24.2   Advantages -- 24.3   Related Patterns -- 24.4   Summary -- 25   Strategy -- 25.1   Case Study: Sending Material Master Data.
25.2   Advantages -- 25.3   Passing Data to the Strategy Object -- 25.4   Optional Strategies -- 25.5   Intermediate Abstract Classes -- 25.6   Related Patterns -- 25.7   Summary -- 26   Template Method -- 26.1   Case Study: Average Transaction Volume -- 26.2   When to Use -- 26.3   Advantages and Risks -- 26.4   Degree of Abstraction -- 26.5   The "Hollywood Principle" -- 26.6   Summary -- 27   Visitor -- 27.1   Case Study: Incoming Invoice Processing -- 27.2   When to Use -- 27.3   Related Patterns -- 27.4   Summary -- A   Object-Oriented Programming -- A.1

  Object-Oriented ABAP Development Environment -- A.2   Class -- A.3   Superclass -- A.4   Abstract Class -- A.5   Interface -- A.6   UML -- A.7   Summary -- B   Subclass Determination -- B.1   Hardcoding -- B.2   Convention over Configuration -- B.3   SAP Class Tables -- B.4   Custom Table -- C   Principles -- C.1   Object-Oriented Principles -- C.1.1   Abstraction -- C.1.2   Composition -- C.1.3   Inheritance -- C.1.4   Encapsulation -- C.1.5   Polymorphism -- C.1.6   Decoupling -- C.2   Design Principles -- C.2.1   Single Responsibility -- C.2.2   Open-Closed -- C.2.3   Liskov Substitution -- C.2.4   Interface Segregation -- C.2.5   Dependency Inversion -- C.3   Anti-Patterns -- C.3.1   Blob -- C.3.2   Copy-Paste Programming -- C.3.3   Functional Decomposition -- C.3.4   Golden Hammer -- C.3.5   Grand Old Duke of York -- C.3.6   Input Kludge -- C.3.7   Jumble -- C.3.8   Lava Flow -- C.3.9   Object Orgy -- C.3.10   Poltergeist -- C.3.11   Reinvent the Wheel -- C.3.12   Spaghetti Code -- C.3.13   Swiss Army Knife -- C.3.14   Vendor Lock-In -- D   The Author -- Index -- Service Pages -- Legal Notes.

Sommario/riassunto

Use design patterns to step up your object-oriented ABAP game, starting with MVC! Want to create objects only when needed? Call objects only when required, minimizing runtime and memory costs? Reduce errors and effort by only coding an object once? Future-proof your code with a flexible design? Design patterns are the answer! With this guide, you'll get practical examples for every design pattern that will have you writing readable, flexible, and reusable code in no time! Creational Design PatternsCreate objects with the abstract factor, builder, factory, lazy initialization, multiton, prototype, and singleton design patterns Structural Design PatternsAllow objects to interact and work together without interdependency with the adapter, bridge, composite, data access object, decorator, facade, flyweight, property container, and proxy design patterns. Behavioral Design PatternsIncrease the flexibility of your object communication with the chain of responsibility, command, mediator, memento, observer, servant, state, strategy, template method, and visitor design patterns. Highlights: MVC (model, view, controller) pattern Singleton pattern Factory pattern Builder pattern Observer pattern Visitor pattern Lazy initialization pattern Template method Strategy pattern Decorator pattern ABAP-specific examples Anti-patterns