Summary .11

Brief history of Delphi and its evolution .17

- Origins and Launch **.18**
- First versions **.18**
- Expansion and Growth **.19**
- Transition and Change of Ownership **.19**
- Renovation under Embarcadero **.20**
- Delphi Today **.20**

Introduction to Object Orientation .23

- Basic OOP Concepts **.26**
- Class **.26**
    - Structure of a Class **.27**
    - Importance of Classes **.30**
- Object **.30**
    - Creating an Object **.31**
    - The new Operator vs. Create Method **.32**
    - Memory Management **.32**
    - Identity and State **.33**
- Method **.34**
    - Functions vs. Procedures **.35**
    - Definition and Implementation **.35**
    - Visibility of Methods **.37**
    - Virtual Methods and Polymorphism.. **.37**
    - Importance of Methods **.38**
- Properties... .39
    - Definition of Properties **.40**
    - Access Methods **.41**
    - Read-Only and Write-Only Properties **.41**
    - Properties and Encapsulation **.42**
    - Advanced Properties **.42**
- Abstraction **.43**
    - How Abstraction Works **.44**
    - Benefits of Abstraction **.45**
    - Levels of Abstraction **.46**
- Encapsulation **.47**
    - Access Modifiers **.48**
    - Principle of Least Privilege **.49**
    - Advantages of Encapsulation **.49**

- Inheritance **.51**
  - Benefits of Inheritance **.51**
  - Implementation of Inheritance in Delphi **.52**
  - Overriding Methods **.53**
  - Constructors and Destructors in Inheritance **.54**
  - Considerations about Inheritance **.54**
- Polymorphism **.55**
  - Types of Polymorphism **.56**
  - Inclusion Polymorphism in Delphi **.56**
  - Polymorphism and Interfaces **.57**
  - Advantages of Polymorphism **.59**
- Abstraction, Encapsulation, Inheritance, and Polymorphism **.60**
  - Abstraction **.60**
  - Encapsulation **.61**
  - Inheritance **.61**
  - Polymorphism **.62**
- Advantages of Using OO in Delphi **.63**
  - Improvement in Code Organization **.64**
  - Code Reusability **.64**
  - Maintainability **.65**
  - Flexibility through Polymorphism **.65**
  - Rapid Application Development (RAD) **.66**
  - Quality and Performance **.66**

Working with Classes and Objects in Delphi .69

- Inheritance Multiple vs. Interfaces **.72**
  - Multiple Inheritance **.72**
  - Multiple Inheritance and its Complexities **.73**
  - Alternative in Delphi: Composition **.75**
  - Interfaces as a Polymorphic Solution **.76**
- Abstract and Concrete Classes **.77**
  - Abstract Classes **.78**
  - Concrete Classes **.79**
  - When to Use Them **.81**
- Using TObject and Other Base Objects in Delphi **.82**
  - TObject: The Cornerstone **.83**
  - TComponent: Base for Visual and Non-Visual Components **.84**
  - Other Base Classes **.85**
- Memory Management and Property Models **.87**
  - Memory Management in Delphi **.88**
  - Manual Memory Management **.88**
  - Automatic Memory Management **.89**
  - Property Models **.90**
  - Read and Write Properties **.91**
  - Calculated Properties **.91**

- Design Patterns for Object Creation and Destruction **.93**
  - Singleton Pattern **.93**
  - Factory Method Pattern **.95**
  - Prototype Pattern **.96**
- Dynamic Methods and Properties **.98**
  - Dynamic Methods **.98**
  - Using the Dynamic Keyword **.98**
  - Dynamic Properties **.99**
  - Simulating Dynamic Properties **.100**
- Reflection and Metaprogramming **.102**
  - Reflection in Delphi **.102**
  - Metaprogramming in Delphi **.104**
  - Example of Metaprogramming in Delphi **.104**

Delphi and the Use of Interfaces .107

- Definition and Implementation of Interfaces **.110**
  - Definition of Interfaces **.110**
  - Implementation of Interfaces **.111**
  - Automatic Memory Management with Interfaces **.112**
  - Advantages of Using Interfaces **.113**
- Interfaces vs. Inheritance **.114**
  - Inheritance **.115**
  - Advantages of Inheritance **.115**
  - Disadvantages of Inheritance **.116**
  - Interfaces **.116**
  - Advantages of Interfaces **.116**
  - Disadvantages of Interfaces **.117**
  - Interfaces vs. Inheritance in Delphi **.117**
- Design Patterns with Interfaces **.120**
  - Strategy Pattern **.120**
  - Observer Pattern **.121**
  - Factory Pattern **.122**
  - Adapter Pattern **.123**
- Practical Examples of Using Interfaces **.125**
  - Example 1: Plugin System **.125**
  - Example 2: Data Access Abstraction **.127**

Conclusion .131