

©2016 Capgemini. All rights reserved.

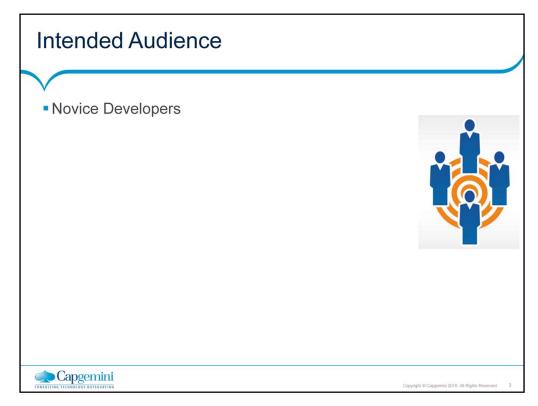
The information contained in this document is proprietary and confidential. For Capgemini only.

Course Goals and Non Goals

- Course Goals
 - To learn about how to write good program by understanding concepts like
 - Readability
 - Maintainability
 - Modularity
 - · Defensive programming
 - · Algorithm analysis and design
 - To learn about how to write pseudocode in design phase
 - To develop robust programs by performing Code Reviews and Unit Testing (test cases/results)
 - Understanding Software testing
- Course Non Goals
 - To learn any specific language features in this course.
 (Language features will be covered in subsequent modules.)







Day Wise Schedule

- Day 1
- Lesson 1: Introduction to program development with pseudocode
- Lesson 2: Good Programming Practices
- Day 2
 - Lesson 2: Good Programming Practices(contd)
 - Lesson 3: Algorithm Analysis and Design
 - Lesson 4: Exception Handling
- Day 3
- Lesson 4: Exception Handling(contd)
 - Lesson 5: Software Reviews and Testing



Table of Contents

- Lesson 1: Introduction to program development with pseudocode
 - 1.1 Introduction to Programs
- 1.2 Types of projects
- 1.3 SDLC process of waterfall model
- 1.4 Introduction to Pseudocode
 - · What is Pseudocode?
 - · Why Pseudocode?
 - How to write Pseudocode?
 - Best practices of writing pseudocode
 - Example of Pseudocode
- 1.5 Usage of variables and operators
- 1.6 Introduction to control constructs
 - Conditional Statement
 - · Looping statement
 - · Guidelines for conditional and looping statements
- 1.7 Introduction to arrays



Table of Contents Lesson 2: Good Programming Practices 2.1 Readable Naming Conventions Comments Guidelines for writing good code 2.2 Maintainable Remove Hardcoded constants

Capgemini

Table of Contents

- 2.3 Modular
 - · Introduction to subroutines
 - · Characteristics of well defined subroutines
 - · Best practices to follow when creating subroutines
 - Guidelines to follow while using arguments in subroutines
 - Best practices to follow for return values from subroutines
- 2.4 Coupling and Cohesion
- 2.5 Robust program
 - Difference between correctness and robustness



Table of Contents

- Lesson 3: Algorithm Analysis and Design
 - 3.1 Algorithm Analysis and efficiency
 - 3.2 Measuring Unit for Algorithm
 - 3.3 Order of Growth
 - Asymptotic notations
 - 3.4 Best/Worst/Average case
 - 3.5 Efficiency of algorithm



Table of Contents

- Lesson 4: Exception Handling
 - 4.1 What is exception handling?
 - Guidelines for creating exceptions
 - Importance of Exception Handling
- 4.2 Case study
- 4.3 Defensive Programming
 - · What is Defensive Programming
 - Purpose of defensive programming
 - · Techniques of defensive programming
 - Input Validation
 - Error Handling
 - Error containment



Table of Contents

- Lesson 5: Software Reviews and Testing
 - 5.1 What is software Testing?
 - 5.2 What is Debugging?
 - Debugging Techniques
 - · Difference between testing and debugging
- 5.3 Software Testing Principles
- 5.4 TestCase
 - What is Test case?
 - How to write Test case
 - Guidelines for implementing test cases
 - Example of Test case
- 5.5 Exhaustive Testing and Economics of Testing



■ Lesson 6: Software Reviews and Testing(Contd...) ■ 5.6 Testing Techniques ■ Static Testing ■ Dynamic Testing ■ 5.7 Static Testing ■ Self review ■ Peer Review ■ Peer Review ■ Group Review

■ 5.8 Dynamic Testing ■ Blackbox Testing ■ WhiteBox Testing ■ 5.9 Testing Approaches ■ Unit Testing ■ Integration Testing ■ System Testing ■ Verification and Validation testing ■ Acceptance Testing ■ Regression testing

