CONTENTS

1 Java/C# Intro, Data types, Operators & Control St.

1.1 History of Java And C# 1
1.2 JAVA : applet, bytecode and JVM 2
1.3 Installing Java Development Kit (JDK), Netbeans IDE and compiling a program 3
1.3 Variable Declarations, Data-types, Operator Basic 3
1.4 Control statement: "if" & "for" 3
1.5 The Java Keywords 4
1.6 The Java Class Libraries 4
1.7 Java's Primitive Types 4
1.8 Literals 4
1.9 Character escape sequences or backslash character constants 4
1.10 Hexadecimal, Octal, and Binary Literals 5
1.11 String literals 5
1.12 Initialization and Dynamic Initialization 5
1.13 Scope and Lifetime of Variables 5
1.14 Operators 6
1.15 Operator Precedence 7
1.16 Type Conversions and type-cast (Recall C/C++ 3.5, 3.6) 7
1.17 Spacing and parenthesis 7
1.18 Input Characters from the Keyboard 7
1.19 if-else, Nested if, if-else-if ladder 8
1.20 Switch statement(similar C/C++ switch . 8
1.21 for loop (with variations)[Recall C/C++ 2.5 , 2.11] 8
1.22 while and do-while loop (similar to C/C++) 9
1.23 Nested Loops: 9
1.24 "break" and "continue" 9
1.25 "break" and "continue" with LABEL (Replacing "goto") 9
C#_1.1 the .NET Framework 11
C#_1.2 Managed &Unmanaged Code and Common Language Specification (CLS) 11
C#_1.3 Compile and Run first program 11
C#_1.4 Variable Declarations, Data-types, Operator Basic, Basic if & for and STATEMENT BLOCK is same as JAVA [recall 1.3, 1.4 Java part]. 12
C#_1.5 The C# Keywords 12
C#_1.6 The C# Class Library 12
C#_1.7 C#'s Value Types 12
C#_1.8 Formatted Output 12
C#_1.9 Literals 13
C#_1.10 Variable INITIALIZATION, DYNAMIC Initialization are same as JAVA 13
C#_1.11 Implicitly Typed Variables 13
C#_1.12 Life-time and scope of variables: 13
C#_1.13 Operators : Following operators are same to 1.14 (Java Part)| 13
C#_1.14 Type Conversions and Type casts : Same as 1.16 (Java Part) 13
C#_1.15 Operator Precedence: 14
C#_1.16 C#'s type promotion rules 14
C#_1.17 Inputting Characters from the Keyboard 14
C#_1.18 The if Statement, Nested ifs, The if-else-if Ladder 14
C#_1.19 The SWITCH Statement, NESTED SWITCH Statements, 14
C#_1.20 For-loop and its variations, While, Do-While & Nested-loops 14
C#_1.21 Continue and Break are same as C/C++, C# doesn't support CONTINUE-BREAK LABEL (Java does) 14
C#_1.22 goto-lebel Jump/loop is Supporetd by C# as C/C++ 14

2 Class, objects & method overloading

2.1 Class in java 15
2.2 Reference Variables and Assignment 15
2.3 Methods and returning from methods 16
2.4 Methods with parameters 16
2.5 Constructor 16
2.6 General form of "new" and details about object declaration 17
2.7 Garbage collection and finalize() 17
2.8 The this reference 18
2.9 Arrays : One-Dimensional Arrays 18
2.10 Multidimensional Arrays 19
2.11 Array (advanced) 20
2.12 Data structure "stack" and "queue" with array 20
2.13 Enhanced for [For-Each Style for Loop] 21
2.14 Strings 22
2.15 Strings Are Immutable 23
2.16 Strings to control SWITCH and Command-Line arguments 24
2.17 Bitwise Operators 24
2.18 The ? ternary Operator 26
2.19 Signed binary numbers 26
2.20 Access Modifiers 27
2.21 Pass Objects to Methods 27
2.22 Two ways to Pass Arguments 28
2.23 Returning Objects 28
2.24 Method Overloading 29
2.25 Overloading Constructors 29
2.26 Recursion 30
2.27 Static in Java (Variables, Methods and Blocks) 30
2.28 QuickSort Algorithm 31
2.29 Nested and Inner Classes 32
2.30 Varargs: Variable-Length Arguments 33

C#_2.1 Class 35
C#_2.2 Methods and Returning from a Method 35
C#_2.3 Methods with parameters 35
C#_2.4 Constructor and Destructor and NEW 35
C#_2. 5 this Keyword 35
C#_2.6 Arrays 36
C#_2.6 Implicitly Typed Array 36
C#_2. 7 For-each loop 36
C#_2.8 strings 36
C#_2.9 The Bitwise Operators 37
C#_2.10 The ? Operator 37
C#_2.10 C#'s Access Modifiers/Specifiers 37
C#_2.11 Pass an Object Reference to a Method 37
C#_2.12 CALL-BY-VALUE and CALL-BY-REFERENCE 37
C#_2.13 Variable Number of Arguments : params modifier 38
C#_2.14 Returning Objects 38
C#_2.15 Method and Constructor Overloading 38
C#_2.16 Returning Values from Main( ) 39
C#_2.17 Passing Arguments to Main( ) 39
C#_2.18 Recursion 39
C#_2.19 Static in C# 39

3 C# Only : Operator Overloading, Indexers, and Properties

C#_3.1 The General Forms of an Operator Method 40
C#_3.2 Indexers: The [ ] operator 41
C#_3.3 Properties 42
C#_3.4 Use an Access Modifier with an Accessor 42

4 Inheritance in Java/C#

4.1 Inheritance fundamentals 43
4.2 Constructors and Inheritance 43
4.3 Multilevel Hierarchy 44
4.4 Superclass References and Subclass Objects 45
4.5 Method Overriding 46
4.6 Abstract Methods and Abstract Classes 47
4.7 Final 48
4.8 The Object Class 48

C#_4.1 Inheritance Besics 50
C#_4.2 Constructors and Inheritance 50
C#_4.3 Multilevel Hierarchy 50
C#_4.4 Base Class References and Derived Objects 50
C#_4.5 Method Overriding & Virtual Method 50
C#_4.6 Abstract Methods and Abstract Classes 51
C#_4.7 sealed 51
C#_4.8 C# object Class 51
C#_4.9 Boxing and Unboxing 51

5 Packages and Interfaces

5.1 Packages (Encapsulation of classes) 52
5.2 Defining a Package 52
5.3 Packages and Access specifier 53
5.4 Protected Members in Packages 54
5.5 Importing Packages 54
5.6 API: Java's Standard Packages 55
5.7 More abstraction with Interfaces 55
5.8 Implementing Interfaces 55
5.9 Using Interface References 56
5.10 Variables in Interfaces 56
5.11 Interfaces Can Be Extended 57
5.12 Default Interface Methods 57
5.13 Default Method Fundamentals 57
5.14 Multiple inheritance and Interface 57
5.15 static Methods in an Interface 58

C#_5.1 Interface 59
C#_5.2 Using Interface References 59
C#_5.3 Interface Properties and Interface Indexers 59
C#_5.4 Interfaces Can Be Inherited: 60
C#_5.5 Explicit Implementations 60
C#_5.6 Structures 60
C#_5.6 Enumerations 60

6 Exception Handling & I/O

6.1 Exception Handling basics 62
6.2 try and catch 62
6.3 Try and catch advanced 63
6.4 Throw, Rethrow and Subclasses of throwable 64
6.5 Finally and Throws 65
6.6 Built-in Exceptions and Some Recent Features 66
6.7 Chained exceptions 67
6.8 Creating Exception Subclasses 67
6.9 Java I/O System 67
6.10 Byte Streams, Character Streams and Pre-defined Streams 68
6.11 Console I/O using BYTE Streams 69
6.12 File I/O using BYTE Streams 70
6.12.1 Reading from a File 70
6.12.2 Writing to a File 71
6.12.3 Automatically Closing a File 71
11.13 Reading and Writing Binary Data 72
11.14 Random-Access Files 73
11.15 Console-based I/O using Console class 73
6.16 Console I/O Using Character Streams 74
6.16.1 Reading Characters 74
6.16.2 Reading Strings 75
6.16.3 Console Output/writing Using Character Streams 75
6.17 File I/O Using Character Streams 75
6.18 TYPE WRAPPERS and SCANNER class to convert numeric strings 76

C#_6.1 System.Exception Class 78
C#_6. 2 try and catch 78
C#_6.3 Try and catch advanced 78
C#_6.4 Finally 78
C#_6.5 Details on EXCEPTION class 78
C#_6.6 Custom exceptions: Deriving Exception Classes 79
C#_6.7 Catching Custom/Derived-class Exception 79
C#_6.8 Catching Using checked and unchecked 79
C#_6.9 C# I/O System : Predefined Streams 80
C#_6.10 The Stream Classes: Byte Stream, Character Stream and Binary Streams 80
C#_6.11 Console I/O 81
C#_6.12 File I/O (part 1): FileStream and Byte-Oriented File I/O 81
C#_6.13 File I/O (part 2):Character-Based File I/O 82
C#_6.14 Redirecting the Standard Streams 83
C#_6.15Reading and Writing Binary Data 83
C#_6.16 Random Access Files 84
C#_6.17 .NET Structure Name and Parse( ) : 84

CONTENTS

7 C# Only : Delegates, Events, and Namespaces

C#_7.1 Delegates *85*
C#_7.2 Use Instance Methods as Delegates *85*
C#_7.3 Multicasting through Delegates *85*
C#_7.4 Anonymous Methods *86*
C#_7.5 Events *86*
C#_7.6 Multicasting Event *87*
C#_7.7 Anonymous Methods with Events *87*
C#_7.8 Namespaces *87*
C#_7.10 USING directive *88*
C#_7.11 Namespaces: Advanced *88*

8 Threads, Enumerations & Autoboxing

8.1 Multithreading Introduction *89*
8.2 Thread Class and Runnable Interface *89*
8.3 Creating a Thread_1 *89*
8.4 Creating a Thread_2 *91*
8.4.1 Extending Thread or implementing Runnable -> which is better? *91*
8.4.2 Creating Multiple Threads *91*
8.4.3 isAlive() to determine when a THREAD ENDS and join() to control WAITING TIME *92*
8.5 Priorities of Threads *92*
8.6 Synchronization *93*
8.6.1 Synchronized Methods *93*
8.6.2 Synchronized Statement *94*
8.6.3 CONCURRENCY utilities and FORK/JOIN Framework *94*
8.7 Thread Communication: notify(), wait(), & notifyAll() *95*
8.8 Suspending, Resuming, and Stopping Threads *96*
8.9 Using the Main Thread *97*
8.10 Enumerations : Introduction *98*
8.11 Enumerations: Constructors, Methods, Instance Variables *99*
8.12 Restrictions of inheritance, JAVA.LANG.ENUM's ordinal() and compareTo() *99*
8.13 BOXING - UNBOXING and TYPE WRAPPERS *100*
8.14 Autoboxing/Unboxing *100*
8.15 STATIC import *101*
8.16 Annotations (Metadata) *102*

9 Generics

9.1 Generics : Fundamentals *103*
9.2 Generics : Details *104*
9.3 Wildcard Arguments *105*
9.4 Generic Methods and Generic Constructors *106*
9.5 Generic Interfaces *107*
9.6 Raw Types and Legacy Code *108*
9.7 TYPE INFERENCE using DIAMOND Operator <> *109*
9.8 Erasure *109*
9.9 AMBIGUITY Errors and RESTRICTIONS on Generic Classes *109*

C#_9.1 Generics *111*
C#_9.2 Generics Improve Type Safety *111*
C#_9.3 Constrained Types *112*
C#_9.4Multiple Constraints: Details *114*
C#_9.5 DEFAULT VALUE of a Type Parameter *114*
C#_9.6 Generic Structures *114*
C#_9.7 Generic Methods *114*
C#_9.8 Generic Delegates *115*
C#_9.9 Generic Interfaces *115*
C#_9.10 Comparing two type parameters using the = = or ! = operators *116*

10 Lambda Expression, Method referencing and Modules

10.1 Introduction to LAMBDA Expressions ("LE") *117*
10.2 Functional Interfaces ("FI") *117*
10.2.1 Lambda Expression with no Parameters *117*
10.2.2 Parameterized Lambda Expression *118*
10.3 Block Lambda Expressions *119*
10.4 Generic Functional Interfaces *120*
10.5 Pass an LE as an Argument *120*
10.6 Lambda Expressions and Variable Capture *121*
10.7 Exception and LE *121*
10.8 Use an array parameter in LE *122*
10.9Method References (MRf) and Constructor References (CRf) & Introducing separator '::' *122*
10.10 Predefined FIs (PREDICATE interfaces) and LEs with API Library *124*
10.11 MODULE Intro *124*
10.12 MODULE: Declaration and Use *125*
10.13 requires and exports : Details *126*
10.14 java.base And PLATFORM modules *127*
10.15 LEGACY code and the UNNAMED module *127*
10.16 The to clause and requires transitive, multi-module compilation *127*
10.17 SERVICES *128*
10.18 Example of A Module-Based Service *128*
10. 19 Runtime MODULE FEATURES *130*
10.20 Module graph *130*

11 Applet, Event, Swing and JavaFx

11.0 AWT, Swing and JavaFx *131*
11.1 Applet fundamentals *131*
11.2 How an Applet works and applet Skeleton *132*
11.3 repaint(), update() and getGraphics( ) *132*
11.4 Using the Status Window *133*
11.5 Passing Parameters to Applets *133*
11.6 The Applet Class *133*
11.7 Event Handling *134*
11.8 Using the Delegation Event Model *135*
11.9 More Java Keywords and more on "this" *136*
11.10 Swing Intro *137*
11.10.1 Components *137*
11.10.2 Containers: Swing defines two types of containers. *137*
11.11 JButton, JTextField, JCheckBox and JList *138*
11.11.1 JButton: *138*
11.11.2 JTextField: *139*
11.11.3 JCheckBox: *140*
11.11.4 JList: *140*
11.12 Use Anonymous INNER CLASSES or LE to Handle Events *141*
11.13 Create a Swing Applet *141*
11.14 JavaFX fundamentals *142*
11.15 JavaFX Label *143*
11.16Events handling: Buttons, CheckBox, ListView and TextField *144*
11.17 Effects and Transforms *146*

12 C# Only : LINQ, LEs, Pre-processors, RTTI

C#_12.1 LINQ Intro *148*
C#_12.2 QUERY *148*
C#_12.3 Executing a Query Multiple times *149*
C#_12.4 Relation between types in a QUERY *149*
C#_12.5 Query: Details (with clauses and keyword) *149*
C#_12.5.1 WHERE (Filter Values) *149*
C#_12.5.2 ORDERBY (Sort Results) *150*
C#_12.5.3 SELECT *150*
C#_12.5.4 GROUP (Group Results) *150*
C#_12.5.5 INTO (Create a Continuation) *151*
C#_12.5.6 LET (Create a Variable in a Query) *151*
C#_12.5.7 JOIN (Join Two Sequences) *152*
C#_12.6 Anonymous Types and Object Initializers *152*
C#_12.7 GROUP JOIN *153*
C#_12.8 Query Methods and Lambda Expressions (LEs) *154*
C#_12.8.1 Basic Query Methods *154*
C#_12.8.2 Lambda Expressions Introduction *154*
C#_12.8.3 Create Queries by Using the Query Methods *154*
C#_12.8.4 EXPRESSION TREE in LEs *155*
C#_12.8.5 More Query-Related EXTENSION Methods *155*
C#_12.8.6 Deferred vs. Immediate Query Execution *155*
C#_12.9 EXTENSION Methods: Details *156*
C#_12.10 Lambda Expressions: Details *156*
C#_12.11 The Preprocessor *157*
C#_12.12Runtime Type Identification (RTTI) *158*
C#_12.13 Nullable Types *159*
C#_12.14 Unsafe Code *160*
C#_12.15 Attributes *161*
C#_12.16 Conversion Operators *161*
C#_12.17 Introduction to Collections *162*
C#_12.18 List<T> *163*
C#_12.19 Queue<T> *163*
C#_12.20 Other Keywords *164*

LuHuSaHer:17112020