Core Java Development Course For Beginners

Course Synopsis

By : Udayan Khattry

Table of Contents

1 Introduction

1.1 Introduction to this course

2 Setup

2.1 Download, Install and Configure Software

This lecture is all about downloading and installing software that will be used throughout this course.

- You are going to download & install following:
  - JDK (Java Development Kit)
  - Notepad++
  - Eclipse IDE
- You will also learn how to configure JDK and Eclipse on your systems.

3 Let's Start coding

3.1 Write your first java program in Notepad

In this lecture, you will:

- Write your first Java Program in Notepad++ (Text Editor), compile and run it using Windows command prompt.
- Learn some details regarding class file and how to execute it.
- Learn how to check and correct compilation errors.

3.2 Write your first Java Program in Eclipse IDE

In this lecture, you will learn:

- To create a workspace folder in Eclipse.
- To open Package Explorer, Command view in Eclipse.
- To create a new Java Project and your first class.
- To compile and run java classes in Eclipse.
- To check compilation errors in Eclipse.
- Windows folder structure corresponding to Eclipse project, path for .java and .class files.
- Some of the cool shortcuts in Eclipse.

3.3 Short explanation on JVM, JRE and JDK

In this lecture, you will learn:

- Development Steps we followed to Write, compile and execute/run our first Java Program.
- Why JAVA is Write Once and Run Anywhere language?
- What are JVM, JRE and JDK?

3.4 Dissection of MyFirstClass Java Program

In this lecture, you will learn:

- Explanation of MyFirstClass line by line.
- IDENTIFIERS in Java and their naming rules.
- Various keywords in Java.
- Explanation of main(String [] args) method in java.
- Difference between System.out.println(""); and system.out.print(""); statements.
- Few more cool Eclipse shortcuts.

3.5 Comments in Java Program

This is one of the most important topics in learning Java. In this lecture, you will learn:

- About comments and why they are so important.
- Two types of comments: Javadoc and developer.
- How and where to add comments in your code?
- Various comment related syntax.
- What are TODO statements?
- To generate a help document using your Javadoc comments.
- To find out and use functionality of third party classes.
- How to open the current folder in Command prompt using Notepad++?
- How developer comments can help other developers / support staff to make changes in the code?
- Eclipse Shortcut to comment multiple lines of code.
- Download and use Java documentation of JDK 8.

3.6 Quiz01 – Let's start coding – 10 questions

4 Variables, Data Types & Operators

4.1 Literals and Variables

In this lecture you will learn:

- Integer Literals
- Floating-Point Literals
- Boolean Literals
- Character Literals
- How to use characters from other languages?
- How to change character encoding in Eclipse to print special characters on to the console.
- String Literals
- Null Literal
- What is a variable in Java?
- Variable for Primitives and Object Reference

4.2 Primitive Data Types in Java

In this lecture you will learn:

- Primitive types: byte, short, int and long
- How positive and negative numbers are stored in the memory?
- Convert decimal to binary
- 2's complement method
- Primitive types: float, double, char and boolean
- How integral values can be assigned to char type and vice versa?

4.3 Operators in Java

In this lecture you will learn:

- Arithmetic Plus (+) operator
- Implicit casting and explicit casting for primitive data types
- Remaining Arithmetic operators (-, *, /, %)
- Division(/) and remainder(%) operators with integral type and floating point numbers
- Unary operators (+, -, ++, --, !)
- Difference between Pre and Post increment/decrement operators
- Relational operators (>, >=, <, <=, ==, !=)
- Logical operators (&&, &, ||, |, ^, ?:)
- Truth table for &&, &
- Truth table for ||, |
- Truth table for ^

- Assignment operators (=, +=, -=, *=, /= and %=)
- Operator Precedence and Associativity
- How to solve an expression containing various operators?

4.4 Quiz-02 - Variables, Data Types and Operators – 10 Questions

5 Control Statement Types

5.1 Selection Statements: if & if - else

In this lecture you will learn:

- What are Control flow statements in Java
- Various Control flow statements
- if and if-else selection statements
  Interesting scenarios when curly brackets {} are not used

5.2 Selection Statements: if - else if - else

In this lecture you will learn:

- if - else if - else selection statement.
- Solving a problem using either if, if - else or if - else if - else statement and decide which one to use when.

5.3 Selection Statements: switch - case

In this lecture you will learn about:

- switch - case statement
- Solving same problem using if - else if - else
- statement AND switch-case statement
- fall-through
- Moving default block within switch

5.4 Java Coding Challenge - 1

Includes 4 programming assignments on selection statements. Solution is also provided for reference.

5.5 Looping Statements: while

In this lecture you will learn:

- Looping statements available in Java.
- Syntax of while loop

- Working of while loop at runtime
- How while loop can reduce the lines of codes, if you have to execute certain statements multiple times.
- Various boolean expressions used in while loop.
- How to write infinite loops in Java?
- Why while(false){...} block gives compilation error?
- Why if(false){...} block doesn't give compilation error?

5.6 Java Coding Challenge - 2

Includes 4 programming assignments on while loop. Solution is also provided for reference.

5.7 Looping Statements: do - while

In this lecture you will learn about:

- Syntax of do - while loop
- Working of do - while loop at runtime
- Differences between while and do - while loop

5.8 Looping Statements: for

In this lecture you will learn:

- Syntax of for loop
- Working of for loop at runtime
- Converting while loop to for loop
- How to access loop's counter variable outside loop's body?
- How to write an infinite for loop?
- Scenarios that cause "Unreachable code" error with for loop.
- Interesting examples on Step expression.

5.9 Java Coding Challenge - 3

Includes 4 programming assignments on Looping statements. Solution is also provided for reference.

5.10 Nested Control and Labeled Statements

In this lecture you will learn:

- What are Nested Control Statements?
- Various Nested Control Statements available in Java.
- Nested if statements and nested for loops in detail.
- How Java runtime handles nested if statements?
  Interesting nested if-else example without curly brackets {}

- How to solve confusing nested if-else example asked in various tests.
- How Java runtime handles nested for loops?
- What are multi-dimensional problems and how to solve these using for loops?
- What are labeled statements?
- Which statements can be labeled in Java?
- Example of Labeled statements.

5.11 Java Coding Challenge – 4

Includes 4 programming assignments on Nested control and labeled statements. Solution is also provided for reference.

5.12 Branching Statements

In this lecture you will learn:

- What are branching statements and various branching statements available in java.
- break statement in detail.
- How java runtime handles break statement.
- How to change console buffer size in Eclipse IDE?
- Example of labeled break statement and how it is used to solve some tricy problems.
- continue statement in detail.
- Example of labeled continue statement.
- How java runtime handles continue statement.
- return statement and how java runtime handles return statement.

5.13 Quiz-03 - Control Statement Types – 18 questions

6 Java Arrays, for-each loop & Command-line arguments

6.1 One-dimensional arrays

In this lecture you will learn:

- What is an array?
- Why we need an array?
- One-dimensional arrays
- Various syntaxes to declare One-dimensional array
- How to construct / instantiate One-dimensional array?

- How Java runtime handles declaration and instantiation of One-dimensional arrays?
- Difference between primitive arrays and Object arrays.
- Default values of primitive and reference types.
- How to read and modify array elements?
- Primitive type array example.
- What is an ArrayIndexOutOfBoundsException?
- Reference type array example.
- When array syntax throws ArrayIndexOutOfBoundsException and when it is a candidate for compilation error.
- What happens when primitive type, String type and Array reference types are passed to System.out.println(); statement.
- length property of an array object.
- Relationship between length property and last index of the array object.
- for loop to iterate through one-dimensional arrays.
- More use cases of array's length property.
- Use of concatenation operator in System.out.println() statement to get desired output.
- Use of for loop to assign related values to the array elements.
- Use of multiple comma separated statements in initialization and step expressions of for loop.
- One-dimensional array object instantiation with data.
- Final solution to average calculation problem.

6.2 Java Coding Challenge - 5

Includes 5 programming assignments on one dimensional arrays. Solution is also provided for reference.

6.3 Multi-dimensional arrays

In this lecture you will learn:

- Various syntaxes to declare Multi-dimensional array
- Construct/Instantiate Two-Dimensional Arrays
- How Java runtime handles declaration and instantiation of Two-dimensional arrays?
- Array syntax throwing NullPointerException.
- What is an Asymmetrical Two-Dimensional array?
- What is a Symmetrical Two-Dimensional array?
  Syntax to construct Symmetrical Two-Dimensional array object.
- Nested for loop to iterate through Two-dimensional arrays (Symmetrical OR Asymmetrical).
- More examples of Asymmetrical Two-Dimensional arrays.

- Two-dimensional array object instantiation with data.

6.4 Java Coding Challenge – 6

Includes 4 programming assignments on multi dimensional arrays. Solution is also provided for reference.

6.5 Enhanced for-loop for Arrays

In this lecture you will learn:

- Syntax of enhanced for-loop.
- Use of enhanced for-loop to iterate through One-Dimensional array.
- How Java runtime handles enhanced for-loop for primitive and Object arrays.
- Limitations of enhanced for-loop.
- Use of nested enhanced for-loops to iterate through Two-Dimensional array.
- How Java runtime handles nested enhanced for-loops.

6.6 Command-line arguments

In this lecture you will learn:

- What are command-line arguments?
- Revision on compiling and executing Java programs in Command prompt.
- Array object of size 0.
- How to pass command-line arguments in Command prompt?
- How to pass command-Line arguments in Eclipse IDE?
- Example on practical use of command-line arguments.
- Converting String to int using parseInt method of Integer class.
- What is NumberFormatException and why it is thrown by parseInt method of Integer class.
- How to handle scenario of ArrayIndexOutOfBoundsException in command-line arguments.

6.7 Java Coding Challenge – 7

Includes 4 programming assignments on Java, Arrays, for-each loop and command line arguments. Solution is also provided for reference.

6.8 Quiz-04 – Java, Arrays, for-each loop and command line arguments – 15 questions

7 Packages, Classpath, User-input & Debugging

7.1 Java Package and Classpath

In this lecture you will learn:

- How to use the functionality of third-party classes in our program?
- Issues with classes belonging to default packages.
  package syntax and its meaning.
- What are the requirements to run java classes declared with package syntax?
- javac -help to get help with javac command.
  -d option of javac command.
- Naming convention for package.
- Usage of reversed Internet domain to get unique Java class names.
- How Java Runtime finds the class files to execute?
- -cp and -classpath options of java command [To specify classes required for execution]
- Default value of -cp and -classpath option
- How to execute java classes from other directories / drives available on your laptop/computer?
- 3 ways to use packaged class from another class:
  - Using fully qualified class name in code.
  - By using fully qualified class name in import declaration.
  - By using wild card (*) in import declaration.

- Correct order of package, import and class declaration if all three are available.
- Where to put source code (.java) files?
- Java project structure followed across industry.
- Example to use class file from another project.
- -cp and -classpath options of javac command [To specify classes required for compilation]
- Use of relative path and absolute path with -cp and -classpath option.
- Bonus: Trick to open command prompt at any folder location in Windows.
- How to specify multiple paths in classpath option?
- Revision of all the cases covered in previous two lectures.
- Hands-on exercise on using packages.

- Bonus: tree command to see complete directory structure in tree-view.
- How Eclipse IDE arranges Java and class files declared in default package?
- .classpath file in every java project created by Eclipse IDE.
- How to build projects automatically in Eclipse?
- How to create packaged classes in Eclipse?
- How Eclipse IDE arranges Java and class files declared in packages?
- Restriction on the location of java files created in Eclipse.
- How to create packages in Eclipse?
- How to move java files from default package to a particular package?
- How to refer classes from other projects in Eclipse IDE and how does it affect .classpath file?
- Difference between Fully Qualified Name and Simple Name of the class.
- How import statement maps Simple Name of the class to its Fully Qualified Name at compile time?
- Restriction on the usage of multiple import statements, to point to same Simple Name of the classes in different packages.
- Order of class name resolution by the compiler:
    - Inline usage of the fully qualified name of the class
    - Fully qualified name of the class used in import statement
    - Wildcard (*) used in import statement
- Why Classes created in default package can't be used by packaged classes?
- How to delete a java project and associated project files in Eclipse?
- Implicit import of all the classes of java.lang package.
- How to export Java Projects from Eclipse Workspace?
- How to import Java Projects into Eclipse Workspace?
- How to do a clean build in Eclipse?

7.2 User-input with Scanner

In this lecture you will learn:

- How to rename java projects in Eclipse IDE?
- How to accept byte, short, int, long, float, double and boolean values using java.util.Scanner class?
- java.util.InputMismatchException is thrown on providing out of range or incompatible values.
- How to close Scanner object?
- It is not possible to accept input on closed Scanner object, java.lang.IllegalStateException is thrown in such cases.
- How to accept String data using nextLine() method?

- What is the issue of using nextLine() method after other next methods and how to resolve it?
- How Strings are internally stored as char array in the memory?
- Use of length() and charAt(int index) methods of java.lang.String class to access all the characters of particular String object.
- Use of above two methods of String class to accept char value as user input.
- Ctrl + Shift + O shortcut to import all the classes in Eclipse.

7.3 Miscellaneous – 1

In this lecture you will learn:

- To generate random numbers in Java using java.security.SecureRandom class.
- About some important methods of String class: length, charAt, equals, equalsIgnoreCase, toUpperCase, toLowerCase and trim.
- About code which can throw StringIndexOutOfBoundsException.
- What are whitespaces in Java?
- Use of Ctrl + Shift + R to Search and Open Project resources / files.
- About "Link with Editor" shortcut in Eclipse to link Package Explorer with Code Editor window.

7.4 Java Coding Challenge - 8

Includes 6 programming assignments on Packages, Classpath, and user-input. Solution is also provided for reference.

7.5 Debugging Java codes in Eclipse

In this lecture you will learn:

- What is debugging?
- How to show line numbers in the code editor?
- What is a breakpoint and how to place it?
- How to run java programs in debug mode?
- How to switch between debug perspective and Java perspective?
- Variables tab and how it helps to check current values of the variables and also to change the variable values.
- How to Resume or Terminate the program during debugging?
- Breakpoints tab and how it helps to remove selected breakpoints or remove all the breakpoints.
- How to disable a breakpoint and how to skip all the breakpoints?
- Step Over and Run to line functionality.
- How to execute a statement and see it's result during debugging?

- How to inspect a variable or an expression?
- Expressions tab and how it helps to watch an expression?
- Common mistake made by developers in using expressions tab, to watch any expression that modifies variable's value.
- How to go to the declaration of any method and come back to the calling code?
- How to go to the source code of Java class referred in your program?
- Step Into and Step Return functionality.
- What is method chaining and how to do it in Java?
- How to attach Java API source code in eclipse?
- How to find and open any source code files of Java API in Eclipse?
- How to jump to any method in the chain using Step Into Selection?

7.6 Quiz-05 - Packages, Classpath, User-input & Debugging – 10 questions

8 Object Oriented Programming (OOP) – 1

8.1 Classes and Objects

In this lecture you will learn:

- What is OOP?
- What are objects?
- What are states/properties and behaviours of objects?
- Some examples of real world objects.
- What is Class and how to create a class template based on similar objects?
- How to convert class's blueprint to Java Code?
- How to create objects in Java using new keyword?
- How to read/write properties of Java objects?
- How to invoke methods on Java objects?
- Best Practice: Separation of core business logic and Object creation or testing.

8.2 Constructors

In this lecture you will learn:

- What comes first: Class or Object?
- What are constructors and how constructors are invoked?
- What is a default constructor?
- Syntax of default constructor.

- default values are assigned to instance variables.
- Syntax of parameterized constructor.
- Code to invoke various constructors.
- Difference between parameters and arguments.
- No-argument constructor.
- What is the scenario in which default constructor is not provided by Java compiler?
- this(); calls no-argument constructor from parameterized constructor.
- this(...) with argument(s) calls parameterized constructors.
- Call to this(...); should be the first statement inside parameterized constructor.
- Assigning values to instance variables at the time of declaration.
- What is constructor overloading?
- How to copy and paste whole packages (containing files) in Eclipse IDE?
- Usage of Ctrl + O to get the list of all the methods and constructors of a class in Eclipse IDE.

8.3 Stack and Heap

In this lecture you will learn:

- STACK and HEAP memory areas.
- What goes in HEAP and what goes in STACK?
- Implicit variable "this" available to all the instance methods and constructors of the class.
- Variable "this" is not accessible to static methods.
- Difference between this() and this.
- PUSH and POP terms related to STACK.
- Validate the STACK and HEAP concepts covered in previous lecture using Debug mode.
- Expressions and display view available in debug mode.
- Content assist feature of display view.

8.4 Java Coding Challenge – 9

Includes 3 programming assignments on classes, objects, constructors, stacks and heap. Solution is also provided for reference.

8.5 Methods

In this lecture you will learn:

- What are methods?
- Syntax for method declaration.
- A method can return primitive type or reference type.

- Use of return keyword to return the value from the method.
- Use of keyword void in method declaration, if method doesn't return any value.
- A method can accept 0 or more parameters.
- Method types: instance method and static method.
- Syntax to invoke instance and static methods.
- Type instance(non-static) and static methods in Eclipse IDE.
- Decide on the method name, its parameters and return type.
- Provide test codes for the methods.
- Content-assist feature of Eclipse IDE shows the list of all the instance/static variables and methods, so you can select, instead of typing the method/property name.
- How to decide whether method should be static or non-static?
- Correct way to call static methods.
- Method declaration and Method signature.
- One java file can have multiple classes declared in it, but only one class with public access specifier.
- NullPointerException and why it is thrown.
- How to handle NullPointerException?
- How to call a method from another method of the same class?
- instance methods can invoke both static and instance methods.
- static methods can only invoke static methods.
- Example on moving duplicate codes to a method.
- Eclipse shortcut to correct all the variable names at once.
- STACK and HEAP memories during method invocation.
- pass-by-value and pass-by-reference schemes.
- More scenarios on pass-by-value and pass-by-reference schemes.

8.6 Java Coding Challenge – 10

Includes 3 programming assignments on Methods. Solution is also provided for reference.

8.7 Variables

In this lecture you will learn:

- Need for static / class variables.
- How java runtime handles static / class variables?
- When are static variables initialized?
- Syntax to access static variable.
- LOCAL variable.
- Types of variables: "class or static variables", "instance variables" and "local or automatic variables".

- Scope of the variables.
- Lifetime of the variables.
- Usage of curly brackets within method body to shorten the scope of the variable.

8.8 Variable Shadowing

In this lecture you will learn:

- Not possible to have multiple variables of same name and type(static or instance or local) in the same scope.
- What is variable shadowing?
- Shadowing of static variables in detail.
- Full name and short hand notation of static variables.
- Shadowing of instance variables in detail.
- Full name and short hand notation of instance variables.
- Eclipse IDE uses "L" icon to denote local variables in debug mode.
- How to generate constructor code (not type) in Eclipse IDE?
- Local variables are not initialized to default values.
- Local variables needs to be assigned with some values before they can be used.

8.9 Initialization Blocks

In this lecture you will learn:

- static initialization block, its syntax and why it is needed.
- instance initialization block, its syntax and why it is needed.
- Code area within class boundary but outside methods and constructors is for declaration statements and not for other code statements.
- How Java runtime handles static and instance initialization blocks?
- What is watch-point in Eclipse debugging and its significance?
- Display view of Eclipse debugging.
- static initialization block throws ExceptionInInitializerError instead of actual Exception object.

8.10 Garbage Collection

In this lecture you will learn:

- Meaning of unreachable objects.
- What is garbage collection?
- Runtime.getRuntime().gc() and System.gc() calls.
- Why java.lang.StackOverFlowError and java.lang.OutOfMemoryError are thrown?

8.11 Java Coding Challenge – 11

Includes 5 programming assignments on variables, variable shadowing, initialization blocks and garbage collection. Solution is also provided for reference.

8.12 Quiz-06- Object Oriented Programming (OOP) – 1 – 20 questions

9 Object Oriented Programming (OOP) – 1

9.1 Encapsulation

In this lecture you will learn:

- Why it is a bad idea to declare instance variables with public access modifier?
- Meaning of "private" access modifier.
- What is Encapsulation?
- What are getter and setter?
- How to generate getter and setter codes using Eclipse IDE?

9.2 Access Modifiers

In this lecture you will learn:

- Which access modifiers can be applied to outer java classes and their effects on the classes?
- Which access modifiers can be used with members (Instance/static variables, constructors and methods) and their effects on the members?
- Usage of Eclipse IDE shortcut "Ctrl + Shift + O" to add import statements for all the used classes.

9.3 Inheritance

In this lecture you will learn:

- What is inheritance and related Java syntax?
- Class diagram of UML.
- Meaning of +, #, ~ and - signs in class diagram.
- extends keyword.
- How Java runtime creates instance of a class inheriting from another class?
- Various terminologies associated with inheritance.
- Derived/Sub/Child class can have its own set of properties and methods.
- How to override a method in Derived class?

- How java runtime handles method overriding scenario?
- Overridden method and Overriding method.
- @Override annotation.
- Invoke Parent class's method (from Child class) using super keyword.
- All accessible instance variables of a class are inherited by sub class.
- All accessible instance variables cannot be overridden by sub class.
- Sub class hides the super class's instance variables.
- How to access the instance variable of Super class by using super keyword?
- Similar to instance variables, static fields and static methods of a class are also not overridden.
- Constructors, initialization blocks and their order of execution in case of Inheritance.
- Syntax of the default constructor added by compiler in case of Inheritance.
- 1st statement inside every construtor should either call other overloaded constructors using this(...) or parent class constructor using super(...).
- Compiler by default adds a call to super(); no-arg constructor of parent class.
- What is constructor chaining?
- Constructors and never overridden but they are chained.
- java.lang.Object class.
- Why System.out.println() method behaves differently when String type reference variable is passed?
- toString() method of Object class and need to override it.
- What are Multilevel Inheritance and Hierarchical Inheritance?
- Deadly Diamond of Death problem in Java.
- Working of Ctrl + T keys in Eclipse IDE.
- Details of protected modifier.
- difference between default and protected modifiers.
- Usage of final modifier with methods.
- Usage of final modifier with classes.
- Some of the final classes in Java API.
- Allowed modifiers for local variables.
- Usage of inheritance concepts in real world, with example.
- When to extend from a class and when to avoid inheritance?
- What are the benefits of inheritance?

9.4 Java Coding Challenge – 12

Includes 4 programming assignments on encapsulation, access modifiers and Inheritance. Solution is also provided for reference.

9.5 Polymorphism

In this lecture you will learn:

- Polymorphism and its types.
- What is static polymorphism?
- What are Operator overloading, Constructor overloading and Method overloading?
- Method overloading rules.
- What is dynamic polymorphism?
- What are the limitations of dynamic polymorphism?
- Explicit casting of reference type, its syntax and risks involved with it.
- Why ClassCastException is thrown at runtime?
- How to avoid ClassCastException using instanceof operator?
- Explicit casting and instanceof check are not possible between siblings or unrelated classes.

9.6 Reference Type Method overloading

In this lecture you will learn:

- How overloaded methods accepting reference type parameters are invoked?
- IS-A or IS-AN relationship.

9.7 equals() method of Object class

In this lecture you will learn:

- Why it is needed to override equals(Object) method of Object class?
- How to override equals(Object) method in your classes to compare two instances?

9.8 Overriding Rules

In this lecture you will learn:

- The rules of method overriding.
- What is covariant return?
- Why overriding method cannot use more restrictive access modifiers?

9.9 Primitive Array vs Reference Array

In this lecture you will learn:

- The difference between Primitive type and Reference type arrays.

9.10 Java Coding Challenge – 13

Includes 4 programming assignments on Polymorphism and rules overriding. Solution is also provided for reference.

9.11 abstract keyword

In this lecture you will learn:

- Why abstract methods are needed in Java?
- Syntax of abstract methods and abstract classes.
- abstract classes cannot be instantiated.
- What is Implementing method?
- Use of @Override annotation with implementing method.
- Why to declare a method with final modifier?
- Is it possible to have an abstract class without abstract methods?
- Who invokes the constructor of abstract class?
- Why new A(); gives compilation error whereas new A[5]; is allowed, where A is an abstract class?
- Rules related to implementing methods.
- Access modifiers used with abstract methods.
- Why following modifier combinations are not allowed: abstract and private, abstract and final, abstract and static?

9.12 When abstract class is not enough

In this lecture you will learn:

- It is not always possible to provide clean solutions using abstract classes only, we will see one such example.
- It is advisable to use super class reference variable to refer to the sub class's instance.

9.13 interface in Java

In this lecture you will learn:

- What are core and sideline features and how to implement these?
- What is an interface and how to define it?
- How to implement an interface?
- "interface" and "implements" keywords.
- Interface reference type variable referring to the instance of implementing class.
- How the use of interfaces with abstract classes can provide clean solution rather than using abstract classes only?
- How to implement multiple interfaces?

- How Eclipse IDE differentiates among interfaces, abstract classes and concrete classes?
- Recommended naming convention for interfaces.
- Java interfaces are implicitly abstract.
- abstract methods of interface are implicitly public and abstract.
- interface variables are implicitly public, static and final.
- All the method overriding rules are valid for implementing methods as well.
- Interfaces don't have constructors or initialization blocks.
- Differences and similarities between abstract classes and interfaces.
- What are Marker and Funtional interfaces?
- Interfaces in java can extend multiple interfaces using extends keyword.
- Recommended way to access variables of super type interface.

9.14 What's new for interfaces in JDK 8.0

In this lecture you will learn:

- Why default methods were added in Java interfaces?
- Syntax of default method.
- How to access default method in implementer classes?
- Overriding default methods and how to avoid deadly diamond of death issue?
- How to call overridden default methods of interfaces?
- How to provide static methods in interfaces and how to invoke these methods?

9.15 Association & Abstraction

In this lecture you will learn:

- has-a relationship.
- How has-a relationship is depicted in a class diagram?
- Meaning of parent-child notation in association.
- Types of association: Aggregation and Composition.
- How to depict Aggregation and Composition in a class diagram?
- What is abstraction?
- What are the ways to achieve abstraction in your model?

9.16 Java Coding Challenge – 14

Includes 4 programming assignments on abstract classes and interfaces.

Solution is also provided for reference.

9.17 Quiz-07-Object Oriented Programming (OOP) – 2 – 21 questions

10 Java Exceptions

10.1 Introduction to Java Exceptions

In this lecture you will learn:

- What is an exception?
- What is default exception handler and how it handles exceptional events?
- Explanation of stack trace printed on to the console.

10.2 Java Exception Handling Framework

In this lecture you will learn:

- What is traditional exception/error handling approach?
- How business logic is mixed with error handling code in the traditional approach?
- How Java exception handling framework separates error handling logic from business logic?
- An example on Java exception handling framework using FileInputStream class.
- To provide try-catch block for error causing statement.
- How java runtime handles the code containing try-catch block?
- Hierarchy of ArithmeticException class.
- getMessage(), printStackTrace() and toString() methods of Throwable class.
- Throwable and its subtypes are allowed in catch block but Object type is not allowed.
- To provide try-catch block for the method call(causing runtime error).
- try without catch or finally block is not possible.

10.3 Checked and unchecked exceptions

In this lecture you will learn:

- Exception framework hierarchy.
- What are checked and unchecked exceptions.
- Differences between checked and unchecked exceptions.
- How to create and use custom exception classes?
- What is Handle or Declare rule for checked exceptions?
- throw and throws keywords.

- How to provide cause/message to custom exception classes?
- What does "ducking the exception" mean?

10.4 More on try-catch blocks

In this lecture you will learn:

- A method can specify multiple exceptions in its throws clause.
- How to handle multiple exceptions thrown by a method?
- Super type Exception class specified in catch block can handle all the sub types.
- Why is it not advisable to provide one catch for all exceptions?
- Does the order of catch blocks matter?
- try-catch blocks should be continuous without any statements in between.
- multi-catch statements.
- Some best practices related to exceptions.
- Methods can declare checked exceptions but may not throw those exceptions.
- How to re-throw the same exception object?
- How to throw a custom exception object?
- How to link exception objects?
- Nested try-catch blocks.

10.5 How to handle Runtime Exceptions?

In this lecture you will learn:

- ArithmeticException (/ by 0) is handled by checking the denominator value.
- NullPointerException is handled by doing not null check.
- ArrayIndexOutOfBoundsException is handled by checking index boundary.

10.6 Overriding Rules

In this lecture you will learn:

- Overriding rules related to methods throwing exceptions.
- Some tricky scenarios of polymorphic method calls.

10.7 finally block

In this lecture you will learn:

- What is the need of finally block in Java?
- Syntax of finally block.

- When does finally block get executed?
- Order of try, catch and finally block, when all the 3 blocks are present.
- try-finally without catch block is a valid Java syntax.
- The cases for which finally block is executed and the case for which it is not executed.

10.8 try-with-resources

In this lecture you will learn:

- Why try-with-resources statement was added in Java?
- Syntax of try-with-resources statement.
- Difference between closing the resource using try-with-resources statement and closing it in finally block.
- The order of closing the resources, in case we have multiple resources in try-with-resources statement.
- try-with-resources statement without catch or finally is a valid syntax.
- Resources declared in try-with-resources statement are implicitly final.

10.9 Eclipse Debugging: Exception Breakpoint

In this lecture you will learn:

- How to debug exceptional events in Eclipse IDE?
- Usage of conditional suspension in debugging exceptional events.
- Exception breakpoints and their usage in debugging exceptional events.

10.10 Java Coding Challenge – 15

Includes 5 programming assignments on Java exception handling. Solution is also provided for reference.

10.11 Quiz-08 – Java Exceptions – 15 questions

11 String, StringBuilder & Wrapper classes

11.1 String

In this lecture you will learn:

- How String instances are different from other instances?
- Different ways to create String instances.
- What is String literal?
- What is String immutability?
- What is String Pool table?

- The working of intern() method of String class.
- Correct way to compare two String instances for equality?

11.2 Magic of String

In this lecture you will learn:

- An interesting scenario of intern() method, in which changing the position of System.out.println() statement changes the output.
- How garbage collection of String instances is different from other instances?

11.3 Methods of String class

In this lecture you will learn:

- concat, compareTo, compareToIgnoreCase, replace, substring, startsWith and endsWith methods.
- Why java.lang.String class cannot be extended?

11.4 StringBuffer and StringBuilder

In this lecture you will learn:

- The purpose of StringBuffer and StringBuilder classes.
- Difference between StringBuffer and StringBuilder classes.
- When to prefer StringBuilder class over String class?
- append, delete, insert, reverse and toString methods of StringBuilder/StringBuffer class.
- Method chaining of StringBuilder and String methods.

11.5 Wrapper Classes

In this lecture you will learn:

- What are wrapper classes and how to create instances of wrapper classes?
- Important constructors of wrapper classes.
- static valueOf methods of all the wrapper classes.
- byteValue(), shortValue(), intValue(), longValue(), floatValue() and doubleValue() methods of numeric wrapper classes.
- static parse methods of wrapper classes.
- Cases for which NumberFormatException is thrown by constructors and methods of Wrapper classes.
- How to create a wrapper object without constructor or valueOf method?
- What is Auto-boxing and Auto-unboxing?

- What are the scenarios in which two instances of wrapper classes created through boxing are same?
- Cases for which java runtime will not auto-unbox the wrapper objects.
- Why is it a good programming practice to initialize all the variables?
- Why numeric wrapper types cannot be assigned to each other?
- Why you cannot extend from wrapper classes?

11.6 Java Coding Challenge – 16

Includes 7 programming assignments on string, methods of string and wrapper classes. Solution is also provided for reference.

11.7 Quiz-09 - String, StringBuilder & Wrapper classes – 10 questions

12 Enum

12.1 Introduction to Enums

In this lecture you will learn:

- What are enums and why enums were added in Java?
- Limitations of using String constant variables for fixed set of values.
- The syntax of Java enum.
- Similarities and differences between enums and java classes.
- What is java.lang.Enum class?
- Eclipse IDE shortcut to change the selected text's case (upper or lower).

12.2 Enums: Constructors, instance variables and methods

In this lecture you will learn:

- Why variables, constructors and methods are needed in the Enum code?
- How to define variables, constructors and methods in the Enum code?
- Code to invoke the parameterized constructor of an Enum.
- When does java runtime invoke the constructor of an Enum?
- How to invoke methods defined in an Enum?
- Static values() method of Enums.
- Benefits of enums over static final variables.

12.3 Enum and switch-case block

In this lecture you will learn:

- How to associate multiple values with Enum constants?
- switch-case syntax for Enums.
- Two ways to declare Enums.
- When to use full name and when to use shorthand notation to access Enum constants?
- Benefits of enums over static final variables.

12.4 Enum: valueOf, ordinal and compareTo methods

In this lecture you will learn:

- Working of static valueOf(String) method.
- How to check the details of values() and valueOf(String) methods using Javadoc?
- How Enum constants are declared internally?
- What is ordinal value?
- Three ways to check for the equality of Enum constants.
- Working of instance methods: ordinal() and compareTo().

12.5 Override Enum methods

In this lecture you will learn:

- Why it is needed to override Enum methods?
- Syntax to override Enum methods.
- Constant specific class body and its use.

12.6 Java Coding Challenge – 17

Includes 4 programming assignments on Enum. Solution is also provided for reference.

12.7 Quiz-10 – Enum – 10 questions