NetTech India
Get Certified

Core Java

Java Basics: Define the scope of variables. Define the structure of a Java class. Create executable Java applications with a main method; run a Java program from the command line; Import other Java packages to make them accessible in your code. Compare and contrast the features and components of Java such as: platform independence, object orientation, encapsulation.

Working with Java Data Types: Declare and initialize variables (including casting of primitive data types). Differentiate between object reference variables and primitive variables. Develop code that uses wrapper classes

Working with Operators and Decision Constructs
Using Operators and Decision Constructs. Use Java operators; including parentheses to override operator precedence. Test equality between Strings and other objects using == and equals (). Create if and if/else and ternary constructs .Use a switch statement

Creating and Using Arrays: Declare instantiate, initialize and use a one-dimensional array: Declare, instantiate, initialize and use multi-dimensional array.

Using Loop Constructs: Create and use while loops. Create and use for loops including the enhanced for loop. Create and use do/while loops. Compare loop constructs. Use break and continue

Java Class Design: Implement encapsulation. Implement inheritance including visibility modifiers and composition. Implement polymorphism. Object class. Create and use singleton classes and immutable classes. static keyword on initialize blocks, variables, methods, and classes

Working with Methods and Encapsulation: Create methods with arguments and return values. Including overloaded methods. Apply the static keyword to methods and fields. Create and overload constructors; including impact on default constructors. Apply access modifiers.

Working with Inheritance: Describe inheritance and its benefits. Develop code that demonstrates the use of polymorphism; including overriding and object type versus reference type. Determine when casting is necessary. Use super and this to access objects and constructors. Use abstract classes and interfaces

Handling Exceptions: Differentiate among checked exceptions, unchecked exceptions, and Errors. Create a try-catch block and determine how exceptions alter normal program flow. Describe the advantages of Exception handling. Create and invoke a method that throws an exception. "Recognize common exception classes (such as NullPointerException, ArithmeticException, ArrayIndexOutOfBoundsException, ClassCastException)"

Working with Selected classes from the Java API: Manipulate data using the String Builder class and its methods. Creating and manipulating Strings. Create and manipulate calendar data using classes from java.time.LocalDateTime, java.time.LocalDate, java.time.LocalTime,

java.time.format.DateTimeFormatter, java.time.Period . Declare and use an ArrayList of a given type. Write a simple Lambda expression that consumes a Lambda Predicate expression

Generics and Collections: Create and use a generic class. Create and use ArrayList, TreeSet, TreeMap, and ArrayDeque objects. Use java.util.Comparator and java.lang.Comparable interfaces. Collections Streams and Filters.Iterate using forEach methods of Streams and List. Describe Stream interface and Stream pipeline. Filter a collection by using lambda expressions. Use method references with Streams

Lambda Built-in Functional Interfaces: Use the built-in interfaces included in the java.util. function package such as Predicate, Consumer, Function, and Supplier. Develop code that uses primitive versions of functional interfaces. Develop code that uses binary versions of functional interfaces. Develop code that uses the Unary Operator interface

Java Stream API: Develop code to extract data from an object using peek() and map() methods including primitive versions of the map() method. Search for data by using search methods of the Stream classes including findFirst, findAny, anyMatch, allMatch, noneMatch. Develop code that uses the Optional class. Develop code that uses Stream data methods and calculation methods. Sort a collection using Stream API. Save results to a collection using the collect method and group/partition data using the Collectors class. Use flatMap() methods in the Stream API

Use Java SE 8 Date/Time API: Create and manage date-based and time-based events including a combination of date and time into a single object using LocalDate, LocalTime, LocalDateTime, Instant, Period, and Duration. Work with dates and times across time zones and manage changes resulting from daylight savings including Format date and times values. Define and create and manage date-based and time-based events using Instant, Period, Duration, and TemporalUnit

Java I/O Fundamentals: Read and write data from the console. Use BufferedReader, BufferedWriter, File, FileReader, FileWriter, FileInputStream, FileOutputStream, ObjectOutputStream, ObjectInputStream, and PrintWriter in the java.iopackage.

Localization: Read and set the locale by using the Locale object. Create and read a Properties file. Build a resource bundle for each locale and load a resource bundle in an application.