**Java Programming Course for Selenium Automation**

Updated – 13 Nov 2013

Resource Link –

<http://www.tutorialspoint.com/java>

<http://docs.oracle.com/javase/tutorial/java/concepts/object.html>

**Topics**

* **Java Overview**
  + History of Java (James Gosling started 1991, Sun released 1995 Java 1.0, Oracle bought Sun)
  + Object Oriented, Platform Independent, JRE (Java Runtime Environment, Multi threaded)
* **Environment Setup**
  + Download link - <http://www.oracle.com/technetwork/java/javase/downloads/index.html>
  + Setting up the path for windows
  + Download text editor – Eclipse / IntelliJ / Netbeans
* **Basic Syntax**
  + Case Sensitivity
  + Class Names
  + Method Names
  + Program File Name
* **Data types and Variables**
  + Primitive Data types
    - byte, short, int, long, float, double, boolean, char, Reference Data types
  + Variable types
    - Local
    - Instance
    - Class / Static
* **Modifiers**
  + Access Control Modifiers
    - private – visible to the class only
    - public – visible to the world
    - protected – visible to the package and subclasses
    - If no modifiers are declared then visible only to the package
  + Non Access Modifiers
    - The static modifier for creating class methods and variables
    - The final modifier for finalizing the implementations of classes, methods, and variables
    - The abstract modifier for creating abstract classes and methods

* **Objects , Classes, Constructor**
  + Link - http://docs.oracle.com/javase/tutorial/java/concepts/object.html
  + A Objects is programming entities, which is created from Class.
  + A class is a blue print from which individual objects are created
  + Constructor in Java is block of code which is executed at the time of Object creation.
    - Same as Class name
    - No return type
    - Multiple Constructor can be written for a class by number of parameters, type of parameters
* **Basic Operators**
  + Arithmetic
    - +, -, \* , /, %, ++, --
  + Relational
    - ==, !=, >, <, >=, <=
  + Logical
    - && , ||, !
  + Assignment
    - =, +=, -=, \*=, /=
  + Conditional
    - variable x = (expression) ? value if true : value if false
  + instanceOf
    - objects intanceof class
* **Loop Control**
  + while
  + do … while
  + for
    - break, continue
* **Decision Making**
  + If, elseif, else, nested if, switch
* **Escape Sequences**
  + \t, \b, \n, \r, \f, \’, \”, \\
* **Character and Number methods**
  + isLetter(),toLowerCase(),parseInt(),parseInt()
* **String**
  + Strings, which are widely used in Java programming, are a sequence of characters. In the Java programming language, strings are objects
  + Concatenating String
  + String built in method - length() , equals(Object anObject), equalsIgnoreCase(String anotherString), toUpperCase()
* **Array**
  + Java provides a data structure, the array, which stores a fixed-size sequential collection of elements of the same type
  + Declaring Array Variables
  + Creating Arrays
* **Regular Expression**
  + A regular expression is a special sequence of characters that helps you match or find other strings or sets of strings, using a specialized syntax held in a pattern. They can be used to search, edit, or manipulate text and data.
* **Method**
  + A Java method is a collection of statements that are grouped together to perform an operation.
  + Modifiers, Return Type, Method Name, Parameters, Method Body
  + Calling a Method
  + Passing Parameters by Values
  + Overloading Methods
  + The Scope of Variables
  + Using Command-Line Arguments
  + Variable Arguments(var-args)
* **Streams, Files and I/O**
  + A stream can be defined as a sequence of data. The InputStream is used to read data from a source and the OutputStream is used for writing data to a destination.
  + Reading and Writing Characters from Console
* **Exceptions**
  + An exception is a problem that arises during the execution of a program
  + Catching Exceptions
    - try, catch, finally
* **Inheritance**
  + Inheritance can be defined as the process where one object acquires the properties of another. With the use of inheritance the information is made manageable in a hierarchical order. When we talk about inheritance, the most commonly used keyword would be **extends** and **implements**.
* **Overriding**
  + If a class inherits a method from its super class, then there is a chance to override the method provided that it is not marked final.
  + Rules for method overriding
* **Polymorphism**
  + Polymorphism is the ability of an object to take on many forms. The most common use of polymorphism in OOP occurs when a parent class reference is used to refer to a child class object.
* **Abstraction**
  + Abstraction refers to the ability to make a class abstract in OOP.
  + Abstract Class
    - An abstract class is one that cannot be instantiated. All other functionality of the class still exists, and its fields, methods, and constructors are all accessed in the same manner. You just cannot create an instance of the abstract class.
  + Abstract Method
    - An abstract method consists of a method signature, but no method body.
* **Encapsulation**
  + Encapsulation is one of the **four** fundamental OOP concepts. The other three are inheritance, polymorphism, and abstraction.
  + Encapsulation is the technique of making the fields in a class private and providing access to the fields via public methods. If a field is declared private, it cannot be accessed by anyone outside the class, thereby hiding the fields within the class. For this reason, encapsulation is also referred to as data hiding.
* **Interfaces**
  + An interface is a collection of abstract methods. A class implements an interface, thereby inheriting the abstract methods of the interface.
  + An interface is not a class. Writing an interface is similar to writing a class, but they are two different concepts. A class describes the attributes and behaviors of an object. An interface contains behaviors that a class implements.
  + Implementing Interfaces
  + Extending Interfaces
  + Extending Multiple Interfaces
* **Packages**
  + Packages are used in Java in order to prevent naming conflicts, to control access, to make searching/locating and usage of classes, interfaces, enumerations and annotations easier, etc.
  + A Package can be defined as a grouping of related types(classes, interfaces, enumerations and annotations ) providing access protection and name space management.
  + Creating a package, The **import** Keyword
* **Data Structures**
  + Dictionary
    - The Dictionary class is an abstract class that defines a data structure for mapping keys to values.
  + Hashtable
    - The Hashtable class provides a means of organizing data based on some user-defined key structure.
* Properties
* System.getProperties( ) when obtaining environmental values.
* Enumeration
* The Enumeration interface defines a means to retrieve successive elements from a data structure. For example, Enumeration defines a method called nextElement that is used to get the next element in a data structure that contains multiple elements.
* **Collections**
  + A collections framework is a unified architecture for representing and manipulating collections.
  + Map, Hashmap, List, ArrayList
* **Generics**
  + Java Generic methods and generic classes enable programmers to specify, with a single method declaration, a set of related methods or, with a single class declaration, a set of related types, respectively.
  + Generic Methods, Classes
* **Multithreading**
  + Java provides built-in support for multithreaded programming. A multithreaded program contains two or more parts that can run concurrently. Each part of such a program is called a thread, and each thread defines a separate path of execution.