SUMMARY OF UDEMY COURSE

JAVA TUTORIAL FOR BEGINNERS (9 SECTIONS)

RATINGS 4.5/5

SECTION 1 – INTRODUCTION (4 UNITS)

UNIT1: Introduction and Installation

- Eclipse is a specialized editor for Java program.
- Search 'jre7/8' →select 'Java SE Runtime Environment' → Download Windows x86 offline.
- Search 'jdk7/8' → select 'Java SE Development kit' → Download Windows x86 offline.
- Search 'eclipse java ide' → go to Elipse.org → Download Eclipse IDE for Java Development.

UNIT2: What Java is and how it works

- Started by creating text files using editors like notepad end with .java.
- Write Java program in Eclipse → press the button.
- Text file → (Javac JDK) →BINARY FILES → (jvm JRE) → Comp.Program.
- JVM: Java Virtual Machine, an extra layer over your existing computer that provides the kind of computing services for your Java programs to run.
- Different JDK/JRE sometimes for different applications of Java.

UNIT3: Getting a Job and What to Study After Completing Basic Java

- Certification in Java (can be pursued).
- For jobs : check out www.jobserve.
- Most jobs require commercial experience: just keep applying.
- Check out: guru.com.
- Some 3 areas of Java Specialization:
  1) Java Swing - Java desktop program, Internet Program.
  2) Creating Web Application with Java {Servelet and JSP}.
  3) Android Development in Java.
- Check out Java multithreading.

UNIT4: How to get most out of this Course

- Try to apply everything you learn, do not have to type the exact code.
- Type it yourself.
- Learn to touch type.
- Using Java for arts, search tutorials and sample codes on applications of Java.

SECTION2 – PROGRAMING CORE JAVA (48 UNITS)

UNIT1: Hello World Program (Application.java)

- Open Eclipse → Click on “Workbench” → “FILE” → New Java Project → right click on FOLDER, select CLASS → Tutorial APPLICATION .java.
- Print align method: Java way of writing to console.

UNIT2: Using Variables (Variables.java)

- Variable types: 1) int [32bit]
  2) Long [64bit]
  3) Short [16bit]
  4) Double
  5) Float [put an ‘f’ at the end]
  6) Char [Unicode, any set of characters]
  7) Boolean [True or false]
  8) Byte [8bits]
- We mainly use int for Integers
- Sysout + (Ctrl + Space)

UNIT3: Strings - Working with Text (Application.java – Tutorial 3)

- First non-primitive type: String type: representing text in Java.
- Create a new project: Tutorial 3: Finish: Create a class.
- We will not tick ‘public domain’.
- Very program in Java must have a main method.
- Main + (Ctlr + Space).
- Concatenate string with ‘+’.

UNIT4: While Loops (While_loop.java)

- Use Booleans.

UNIT5: For Loops (UseForLoop.java)

- Class name must start with a capital letter.
- “Ctlr + Shift + F” automatically format codes.
- “for ()” bracket contains the conditions for the FOR LOOP.
- “(; ;)” 2 semicolons divide into 3 sections.
- 1st section initials the codes.
- 2nd section specifies condition.
- 3rd section does the iterations.
- Instead of “I = I +1” we can use “I + +”.
- “System.out.printf “ is a format specifier.

UNIT6: IF (TryingIf.java and CombineWhileundif.java)

- Single “=” means assignment.
- Double (==) means equality.
- IF statement is mutually exclusive (forgotten the meaning check it out …).

UNIT7: Getting User Input (UserInput.java)

- Scanner class is used to get user input.
- “Ctrl + Shift + O” to get all necessary importations.

UNIT8: Do...While (OnDoWhile.java)

- Scanner object.
- “/*….*/” for multiline comments.
- Try to understand variable scope.

UNIT9: Switch (CopierSwitch.java)

- Switch allows you to take different actions depending on the value of the variables (similar to IF).
- Case is similar to else if.
- Default is similar else.
- You can only switch on some variables, mostly strings and int.

UNIT10: Arrays (DoingArrays.java)

- Int value = 7; is a value variable.
- Int [] values; is a reference variable.
- For loop to iterate through array.

UNIT11: Arrays of Strings (Arraysstrings.java)

- Array of strings = array of integers.
- String [] words = {a, v, b, g }.
- Use for loop to iterate, for (Strings I : words).
- Primitive types have lower case letters.
- Classes start with capital letters.
- Default value for string is “null”, for int = 0.

UNIT12: Multi-Dimensional Arrays (MultiDaRRAYs.java)

- Int [] [] grid = {} for 2D arrays.
- Multi- Dimensional Arrays is an array of arrays.
- String [] [].

UNIT13: Classes and Object (Classinstance.java)

- If a class is named (tagged) public the class name must match the file name
- A class is a blueprint for creating objects.

- Classes can contain:
    1. Data (state of object, instance variables)
    2. Subroutine (methods)
- Person(type) person(name) = new Person (a new Person object).

UNIT14: Methods or Subroutines (ClassMethods.java)

- Methods should always start with a lowercase letter.
- “Ctrl + Shift + F” help to properly indent code.

UNIT15: Getters and Return Values (ClassGetterReturn.java)

- Getters and return methods: int *** () {} / String *** () {}.

UNIT16: Method Parameters (MethodParameters.java)

- Pass values to methods.
- Method’s “()” is like a chute you can throw data through.
- Variables you pass into methods are called Parameters.
- Parameters have to be called in the right type and right order.

UNIT17: Setters and “this” (SettersThis.java)

- You never go wrong typing public in front of your method.
- Encapsulation is simply hiding away the instance data (like drugs in a capsule).
- Private (to enforce encapsulation) means that it cannot be accessed outside of the class.
- “this.***” means that particular instance variable ***.
- It is not necessary to use “this.***” every time.
- Use it mainly in scenarios of ambiguity where 2 or more variables have the same name.

UNIT18: Constructors (Constructor.java)

- Constructor is a special method which run every time you create an instance of your class.
- Constructor does not have a return type (void, int, String).
- Name of constructor is the same name with class.
- To initialize instance variables, we can use Constructors.
- Use “this ” to call a constructor in a constructor.

UNIT19: Static and Final (LearningStaticandFinal.java)

- Static (class variables) means that only one copy of string or int for all instance.
- Static methods cannot output instance variables.
- Static methods can access static variables.
- Instance methods can access static data or variables.

- Constant values are represented by uppercase letters.
- Final is a java word used for “Constant values” which cannot be changed or reassigned.
- Static can be used to count the number of objects created.
- Static can be used to assign ID to objects.

UNIT20: String Builder and String Formatting (FormattingString.java)

- StringBuilder is a class, it has an append method, toString method.
- It is more memory efficient way of appending text.
- Advance string formatting: “\t” means new tab, “\n” means new line.
- “System.out.printf” : %d : integers.
- %s : strings.
- %10d means field of ten characters wide.
- Floating point formatter: %f .
- %.2f means 2 decimal places.
- %10.1f means 10 width field, 1 decimal place.

UNIT21: The toString Method (MytoString.java)

- The standard methods “Object class” has?? (find out some *smiles/winks*).
- Sysout on an Object tries to invoke the “toString” method.
- Useful for debugging.
- String.format().

UNIT22: Inheritance (Folder: Tutorial Inheritance)

- Car extends Machine: Car is a child class of Machine; Car is derived or inherits from Machine.
- Most classes can be extended (Exceptions like String Class it is a final class).
- You can override methods.
- “Object class is the ultimate Grandfather of all Objects”.
- Method names must be the same in order to override a method.
- Right click: Source: Override Implementation Method: Select the method to override.
- Child classes cannot access private Parent class variables.
- “protected” means that it can be accessed by child classes and anywhere in the package.
- Try not to override variables …stick with overriding methods.

UNIT23: Packages (PracticePackages.java)

- Packages enable you to organize your code in a sensible fashion.
- Prevents conflict between class name.
- Package names are all lowercase letters, and very simple.
- Bin folders contain the classes.
- Src folders contain java files.
- Import package to use the classes from the package.

- Import package.* to import all the classes in the package.
- Right click: Source: Organize Imports = “Ctrl + Shift + O”.
- Packages within packages: ***(main package).***(sub package).

UNIT24: Interfaces (Package: tutorialinterface)

- You cannot do new ***() on an interface name.
- Interface contain the methods you want your object to have.

UNIT25: Public, Private and Protected (Package: practicingppp)

- They are known as access level specifiers.
- Public: can be accessed anywhere (bad practice to make instance variables public).
- If it is public it should be a constant (final).
- Private: can only be accessed within the class.
- Protected: can be accessed within class, sub-class and package.
- No access level specifier: can be accessed within the package.
- PPP: not pertaining to classes.
- Only one public class per java file.

UNIT26: Polymorphism (Package: polymorphismtutorial)

- Polymorphism: using a child class in place of the parent class.
- What matters is the object.
- Variables know what it can/should be able to do.
- Type of variables decides what methods you can call.

UNIT27: Encapsulation (Encapsulation.java)

- Setters and Getters
- Right click: Source: Generate Setters and Getters.
- Try not to make data public except constants to prevent “cross linkage”.
- Java API (Application Programming Interface Document) to check documentations.

UNIT28: Casting Numerical Values (CastingNummer.java)

- Byte.MAX_VALUE to check the maximum number it can take.
- Cast: to transform from one variable type to another.
- int value = (int) long Value (example of casting).
- Try not to cast a type to one of a lesser memory.

UNIT29: Up casting and Down casting (Complexcasting.java)

- Up casting: casting from subclass to superclass.
- Down casting is inherently unsafe, up casting is safe.
- You cannot change an object but you can change the reference.

UNIT30: Using Generic (Genericstuvs.java)

- A generic is a class that work with other objects (you can specify what type of object) when you instantiate.
- Arraylist: manages an array internally.
- With generic class (Arraylist) you have to specify the type.
- Some generic classes can take more than one argument.

UNIT31: Generic and Wildcards (GenericWildcards.java)

- No subclass in Arraylist.
- <?>: arraylist of unknown type.
- Up bound on wildcard: <? extends Machine> (type Machine and subclasses).
- Lower bound on wildcards: <? super Camera > (type Camera and superclass).

UNIT32: Anonymous Class (AnonymousClass.java)

- Anonymous class is a way of extending class and interface.
- AC has no name, cannot create an object.

UNIT33: Reading Text Files (ReadingIO.java)

- Double the backslashes to file directory or forward slashes.
- Using the class “File”.
- Throw file not found exception.
- Invisible character.

UNIT34: Handling Exception (HandlingException.java)

- “Ctrl + Shift + O”: to import.
- Two ways of handling exceptions:
    1. Add throws declaration.
    2. Surround with try/catch.
- Red text = Stack trace: when your throw exceptions.
- “Ctlr + Shift + F”: to auto format your code.
- Try and catch block: Exceptions are thrown only if there is error.
- Custom “catch” phrases are better than red text (stack trace).

UNIT35: Multiple Exceptions (MultipleExceptions.java and DemoException1.java)

- You can have multiple catch blocks.
- For multiple exceptions:
    1. Throw exception
    2. Try/catch
    3. Try/multiple catch

- (Exception e): catches all exceptions.
- You have to handle a child exception before a parent.

UNIT36: Runtime Exception (RenneException.java)

- Two types of exceptions:
    1. Checked Exceptions
    2. Runtime/Unchecked Exceptions (they are very serious)
- We are not being forced to check RE.
- RE point to fundamental flaws in your code.

UNIT37: Abstract Classes (Package: tutabstractclassed)

- Class hierarchy.
- Base class have things common to the sub classes.
- Public “abstract” class Machine: means it cannot be instantiated.
- Abstract methods: similar to interface, must be implemented in child class.
- A class can implement many interface but just one parent or abstract class.
- Abstract class are good for class hierarchy, not as commonly used as interface.

UNIT38: Reading files with File Reader and Buffer Reader (OlderReaders.java)

- File Reader and Buffer Reader: are old ways of reading files.
- Refactor = Rename.
- Scope of variable is limited to the characters around it.

UNIT39: Try with Resources (ExceptionmitTryResources.java)

- Try (): try with resources, it also closes the file automatically.
- Improved over OlderReaders.java.

UNIT40: Writing Text Files (WritingTextFile.java)

- Use Buffered Writer and File Writer same format with reading file.

UNIT41: The Equal Method (MethodEqual.java)

- “== “: checks for equality, checks if the reference point to the same OBJECT.
- 2 OBJECTs are not usually equal with “==”.
- Source: Generate HashCode() & equal(): (tick fields important for comparison).
- “.equal()” : equals method, it compares meanings.
- “==”: works well for int and String but not Double.
- Always use “.equals()” for strings.
- Hashcode: unique ID for each object.

UNIT42: Inner Classes (Package: innerclasses)

- You can only have one “public (top level) class” in a java file.
- Nested/ Inner class: a class inside a public class.
- Non static inner classes are used for grouping
- When creating an instance of the public(main) class you do not automatically create an instance of the inner classes.
- Static inner classes: are used when a class is needed that is not associated with the instances of he enclosing class.
- You can declare class within methods.

UNIT43: Enum (Package: enumerator, don’t really understand)

- Enum: mean enumerator, to count through.
- You don’t use “new” with enum.
- Enum constants are special objects.

UNIT44: Recursion (Recursion.java)

- Recursion: a subroutine calling a subroutine.
- Problem of using recursion: StackOverFlow Error.
- Recursion can be used for factorial calculations.
- Always make sure Recursions have stopping points.
- Loops might be better than Recursions.

UNIT45: Serialization (Package: serializationtutorial)

- Serialization: turning an object into a binary form.
- Reserialization: reverse of serialization.
- FileOutputStream, ObjectOutputStream are classes used for writing.
- You can serialize any data you like sequentially.
- FileInputStream, ObjectInputStream are classes used for reading.
- Must read with the class you write with (same ID).

UNIT46: Serializing Multiple Objects (Package: serializationtutorial)

- To make an object serializable: implement Serializable.
- @suppresswarnings(“unchecked”).
- You can have multiple object within your try with resource.
- Seems there is no difference between “.bin” and “. ser”.

UNIT47: Transient (Package: serializationtutorial)

- Transient keyword used when serializing objects.
- Transient: excludes from serialization.

- Static field are not serializable.
- Deserialization does not run any constructor.

UNIT48: Passing by Value (Package: TutorialPassingValue)

- Passing by value: is what JAVA offers.
- Variable scope is the nearest {}.
- Methods with the same name but different arguments: is called method overloading.

SECTION3 – THE JAVA COLLECTIONS FRAMEWORK (13 UNITS)

UNIT1: Collection 1- Array List; arrays the easy way (Collection1.java)

- All members of collection framework are Classes.
- Collection Framework are also known as Template.
- They cannot take a primitive type like "int".
- ".add()", ".get()", ".remove()"
- "< --- >" is used to specify the template type.
- All list objects implement the List Interface.
- "List" is an Interface.

UNIT2: Collection 2- Linked List (Collection2.java)

- Typical use of the List Interface is to pass it to a function.
- Using "List" Interface covers for ArrayList and LinkedList.
- If you want to remove or add objects at the end of the list use ArrayList.
- If you want to remove or add objects anywhere else use LinkedList.
- ArrayList manage arrays internally [#] [#].
- LinkedList consists of elements where each element has a reference to the previous and next element [#] <-> [#].

UNIT3: Collection3 – Hash Map (Collection3.java)

- Map store pairs of values: key and value.
- ".put()", ".get()"; there is a special function to iterate through a map.
- You cannot have duplicate keys in a MAP.
- Hash Map is not sorted; it does not maintain any order.

UNIT4: Collection4 – Sorted Map (Collection4.java)

- LinkedHashMap is sorted (in the order you add the elements).
- Tree Map is sorted in natural order.
- Collection Classes are organized under various Interfaces.
- Main 3 Interfaces: List, Map and Set.

UNIT5: Collection5 – Sets (Collection5.java)

- Set stores only unique items.
- Hash Set does not retain orders.
- ".add()", ".contains()", ".isEmpty()"
- LinkedHashSet retains the order in which they were added.
- Google Set Interface in Java.
- ".retainAll()" Intersection of two sets.
- ".removeAll()" Difference between two sets.

UNIT6: Collection6 – Using your Objects in Maps and Sets (Collection6.java)

- Adding ".equals()" method allows Sets and Maps to know unique objects and remove duplicates.

UNIT7: Collection7 – Sorting Lists (Collection7.java)

- "Collection.sort()" method
- Use Comparator Interface to do custom sorting.
- Create a class that implements Comparator.
- ".compareTo()".

UNIT8: Collection8 – Natural Order (Collection8.java)

- Collection Interface is a super interface of List and Sets.
- You cannot use "Collections.sort()" on objects without first establishing the natural order.
- If there is a conflict between the ".equals()" method and ".compareTo()" method the Tree Set might misbehave.
- "implements Comparable < >".

UNIT9: Collection9 – Queues (Collection9.java)

- Front of queue is the HEAD.
- End of queue is the TAIL.
- It used a FIFO Structure.
- ArrayBlockedQueue can have a fixed size.
- ".add()", ".remove()", ".element()" Throw exception when out of elements.
- ".offer()", ".poll()", ".peek()" do not Throw exceptions.
- Checkout Queue API documentations.

UNIT10: Collection10 – Using Iterators (Collection10.java)

- Using For Loop is a modern way of iteration.
- Most collections implement the Iterable Interface.
- ".next()", ".hasNext()".
- To remove objects while iterating through them you need to use an "Iterator".
- Use ListIterator to add to a list.

UNIT11: Collection11 – Implementing Iterable (Package: iteratorscomplexstuff) [Quite complex]

- ".implements Iterable< >" to be able to use "For Each" loop on the class.
- For Loop calls the ".next()" method in the Iterator.

UNIT12: Collection12 – Choosing Java Collections (Collection12.avi)

- Main groups of Collections: Set, List and Maps.
- Check out the video.

UNIT13: Collection13 – Complex Data Structure (Collection13.java) [complex For Loop]

- Sometimes you don't know the type of data you will be dealing with.

SECTION4 – APPENDIX (4 UNITS)

UNIT1: Appendix1 – Eclipse Shortcuts

- "main + Ctrl + Space" = main method.
- "Ctrl + Shift + F" = Format the code block.
- "Ctrl + Shift + O" = To add Imports.
- "syserr + Ctrl + Space" = Print red in console.
- "Refactor" = "Rename".
- "Ctrl + D" = To remove an entire line.

UNIT2: Appendix2 – Getting a Job as a Computer Programmer

- Do you need a degree? Not necessary.
- Do you need experience? No, but experience is a bit of hurdle you have to overcome.
- To overcome this hurdle, try to:
    1. Learn to program computers.
    2. Have some portfolio.
    3. Beef up and format your CV well (Don't Lie).
- Strategy for finding work:
    1. Be ready to move anywhere.
    2. Get a good job search site and keep looking.
    3. Apply for everything appropriate.
    4. The more interview, the better you get at it.
- His story:
    1. He did a degree in Physics.
    2. There are so many opportunities Software Development.

UNIT3: Appendix3 – Ten Tips to make you a better Programmer

1. Learn to touch type.
2. Name variables and subroutines descriptively.

3. Type more and read.
4. Write software that interest you.
5. Read Stack traces from topline down.
6. Aim to write the smallest working program possible.
7. Google like crazy.
8. Build programs one step at a time.
9. Ensure braces always pair up (Brackets).
10. Format Code correctly.
11. Go to https://www.caveofprogramming.com.

UNIT4: Appendix4 – Debugging in Eclipse

- Using "Log Statement" in Android and "sysout" to know what's happening in one's Code.
- Set a breakpoint in your code (a point where you want your code to stop).
- Double Click on Eclipse Margin or Right Click: to set Breakpoints.
- "Run Debug".
- "Watching Variables".
- "Step over", "Step into".
- Practice with any Java File.

SECTION5 – WHAT'S NEW IN JAVA 8 (1 UNIT)

UNIT1: Lambda Expressions (Lambdastuv.java)

- Lambda Expressions are ways of passing a block of code to a method.
- In previous versions we use Interface and Anonymous Classes.
- "() -> ".
- LE are usually associated with Interfaces that have a single method.
- Functional Interface: Interface with a single method.
- LE don't have new scope.
- You can assign LE to a Functional Interface.

SECTION6 – TEST (TestFiles Folder)

- Link to test: https://www.caveofprogramming.com/java/basic-java-programming-test-your-knowledge.html.

SECTION7 – MORE

- Recommended Books: Recommended_Books.png

SECTION8 – SOURCE CODE

- Java-For-Complete-Beginners.zip.

SECTION9 – BONUS