## General

1. Practise, practise, practise – it is the most effective means of learning
2. Help each other, remembering to always have mutual respect
   1. Learners – Try to provide as much specific information and some evidence of effort when asking for help or advice
   2. Teachers / Forum members – If the information is missing from the learner’s initial request, inform him/her in a cordial fashion. Remember how you felt when you started.

1. Share information – If you find something useful, share it quickly. It isn’t a competition
2. Groups and group programming
   1. Find others in your class to go through code, in a Peer review manner
   2. Paired programming – Become aware of it and use
   3. Use GitHub and other code sharing facilities
3. Know your strengths and weaknesses and use this - If in a class, you may be strong in other areas than programming. Identify people that are strong in programming and offer to help them in areas other than programming.
4. Learners should not be afraid to ask for help and advice. Inversely, teachers should make it known at that start that they are open to receiving requests for help and advice, no matter how trivial the learner may think it is or if they feel that they might sound stupid asking that question.
5. No matter what the topic being studied, always be clear of
   1. Why you are learning a specific component of the topic
   2. What is the basic functionality of that component
   3. Where to find information on the component quickly
   4. How this component interacts with other related components
6. Pass it on
   1. Teachers – Remember the most effective teachers in your life. Were they the most intelligent or the ones that made time and treated you with humility when you asked a question (which they probably heard 1000 times before)?!
   2. Learners – At some stage you WILL become an expert. Remember the help you received along that journey and reciprocate that to new learners.

## Problem Solving & Coding – DO

1. Work with values you know – Start off with values that will produce known results. Then, as you confirm that your code can deal with the simpler cases, integrate functionality to deal with more difficult cases
2. Break problems down to their most basic of components. A computer must take baby steps and while humans can subconsciously do very complex tasks e.g. arithmetic, these steps must be simplified further in the programming steps. Work out what a human would do (in English) to solve this problem and examine what steps you took
3. Draw diagrams to represent code scenarios – As well as solving problems, they should also clear up any assumptions and verify the various states of data
4. After breaking down the problem to its most basic of steps, build it back up again slowly. Verify that the smaller components compile and then build on that.
5. Think of cases that will break your code e.g. pass null value
6. IF you can’t find the issue, talk yourself through it – rubber duck debugging

## Problem Solving & Coding – DON’T

1. Don’t write all your code before compiling the code for the first time. Finding logical errors will be more difficult if there is too much of your intended algorithm in place by then
2. Think that your code will run correctly even if it compiles – It may run but chances are, will need fine tuning
3. Don’t overengineer just because yu read some difficult scenario online – it will come with time to find better ways to solve some problems.
4. Don’t Start your problem solving at the computer without some notion of a plan

## Language specifics

1. Know the interaction between the various components and their individual elements e.g. the parameters of a for loop
2. Experiment with code – Don’t wait for assignments to test your understanding of code. Use your initiative
3. Techniques & Practises
   1. Store code effectively – You will need to be able to find old, working code quickly
      1. Name should include functionality and date
      2. Versioning – You should be able to find the latest version of code
4. Get into good practises early
   1. Know the main conventions – variable naming, etc.
   2. Member[[1]](#endnote-1) names – Meaningful and preferably short (use refactoring if needed)
   3. Use of indentation
5. Keep your code simple for now. If that means more lines of code but understandable code, so be it.
6. Code should be sectionalised into atomic, functional units i.e. each part does one thing and does it well
7. Terminology – Get to know Java specific terms (insert X-ref to Glossary)
8. Error messages – Know the reasons behind common errors
9. Rather than learning off topics, know how to read (and generate) JavaDocs and API pages
10. Know your IDE - NetBeans / Eclipse
    1. Don’t rely on its automated features e.g. the insertion of {}’s, which can be done at the wrong location, kicking out functionality
    2. Refactoring tools
    3. Debugging tools
       1. Breakpoints
       2. Watches

## Resources – to be completed

1. Courses
   1. Solas eCollege
   2. Coursera, Khan Academy, Udemy
2. Equipment & Other Software
   1. Sketchpads – A3 and A4
3. Books & sources –
   1. for in depth undersanding of the language e.g. Java 8 - OCAJP books for OCA Java examination for Oracle certificate
4. Website forums & learning websites – to be listed
   1. Stackoverflow
   2. javaranch.com – amazing for newbies much much friendly than stackoverflow
5. Website forums & learning websites – to be listed
   1. GoConqr
   2. W3 Schools
   3. <http://www.homeandlearn.co.uk/java/java.html>
   4. <http://www.tutorialspoint.com/java/index.htm> - Good articles

# NetBeans Tutorial and Exercise

* 1. <http://netbeans.org/kb/docs/java/gui-functionality.html>
  2. <http://www.java2s.com/Tutorial/Java/CatalogJava.htm>

# Other Tutorials

* 1. <http://introcs.cs.princeton.edu/java/14array/>
  2. <http://quizlet.com/subject/java-programming/>
  3. <http://www.javatutorialhub.com/java-arrays.html>

NetBeans vs Eclipse debate - <http://www.ibm.com/developerworks/library/os-ecnbeans/?ca=dgr-lnxw02aNetBeans>

Overall Java language specification - Look here for detailed info on methods and built-in classes - <http://docs.oracle.com/javase/7/docs/api/overview-summary.html>

Oracle Java Tutorials - This is a step-by-step set of tutorials to learn the basics of Java <http://docs.oracle.com/javase/tutorial/java/nutsandbolts/index.html>

Tips and Tricks - [www.javapractices.com](http://www.javapractices.com)

Java World Magazine - <http://www.javaworld.com/>

1. Member – Includes Variables and Methods [↑](#endnote-ref-1)