# Software Proficiencies

This document aims to provide clarity on the level of proficiency within the languages listed as well as providing an overview of the works completed during the period of professional development since being made redundant from AVMI.

| Language | Proficiency | Overview |
|---|---|---|
| Crestron / AMX | Advanced | ● Fully accredited within both Crestron (CAP) and AMX (ACE).<br>● Over 12 years' experience delivering systems for over 300 clients throughout the UK, Europe and worldwide.<br>● Tablet/iPad-based platforms built within a modular event driven architecture using a C++ based language used for system control, scheduling, monitoring and reporting within the corporate sector using both on-premise and cloud-based technologies. |
| Java SE / JavaFX | Advanced | ● Substantial training, research, books, courses, exercises, technical questions and projects completed to advanced during period of professional development.<br>● x3 significant projects including 'Bank Account Manager' app which provides advanced dashboard-based chart analysis of personal banking history.<br>● Research and projects available for review within my GitHub repository. |
| Spring | Intermediate | ● Training, research, books and courses completed during period of professional development.<br>● I have a solid foundation in the fundamentals of Spring such that I would be able to rapidly ramp up any proficiency within any specific Spring projects contained within the overall framework. |
| Kotlin | Intermediate | ● Training, research, books and courses completed during period of professional development.<br>● Given my advanced proficiency within core Java transferring these concepts and learning Kotlin is quick and straightforward.<br>● I have solid foundation in the fundamental components, aspects and motivation behind Kotlin such that I would be able to rapidly ramp up any proficiency within this language as required. |
| SQL | Intermediate | ● I have used SQL to varying degrees throughout my working life.<br>● This includes the project 'nationalsyndicate.org.uk' completed during 2004-2005 which had heavy use of backend databases and the more recent x2 Java/JavaFX projects contained within my GitHub repository which are built around embedded JavaDB's.<br>● Completed 'The Complete Oracle SQL Bootcamp' course on Udemy during period of professional development. |
| XML HTTP REST | Intermediate | ● XML is the foundation of x3 substantial Java/JavaFX projects contained in my GitHub repository.<br>● XML, HTTP and REST are fundamental to many modern systems consequently completed significant training, research and courses during period of professional development. |
| Git / GitHub | Intermediate | ● Training, research, books and courses completed during period of professional development. |
| Android | Basic | ● Training, research, books and courses completed during period of professional development. |
| HTML / CSS / PHP / mySQL | Basic | ● In 2004-2005 completed a substantial personal project of creating an online lottery syndicate.<br>● Users would join, create a subscription and define their ball combinations to be included the in nationwide syndicate.<br>● All of the syndicate entries would be bulk purchased securely and electronically for each draw.<br>● Built using HTML / CSS / PHP / mySQL and PayPal.<br>● Front and backend fully operational.<br>● Unfortunately it did not go live due to Camelot not willing to provide support for secure electronic bulk ticket purchasing. |
| Kubernetes / Docker | Basic | ● Training, research, books and courses completed during period of professional development. |
| Workflows / Build Tools | Basic | ● Training, research, books and courses completed during period of professional development. |

In order to provide an indication of the level of proficiency please find below a detailed breakdown of the areas of research and proficiency within the languages listed. The below corresponds to a concise summary of the collated key concepts, constructs and syntax produced during my period of professional development contained within the Research folder of this GitHub repo.

## Java SE

### Top Level

- Class
- Abstract Class
- Interface
- Enum

### Nested Types

- Types
  - Local Class
  - Inner Class
  - Static Nested Class
  - Anonymous Class
  - Lambda Expressions
  - Method Reference
- Members
  - Permitted Members
  - Access to Outer Scopes
  - Shadowing
  - Final or Effective Final
- Nesting Principles
- Memory Depiction
- Instantiation From External Scopes

### Static Nested Class

- Effective Top Level Class
- Internal Memory Depiction
- Permitted Access to Outer Scopes
- Instantiation From External Scopes

### Class

**Declaration / Definition**

- Header / Body Syntax
- Access Modifiers
- Memory Compostion
  - Static
  - Non-Static
- Overloading
- Overriding
- Shadowing
- Fields
  - Instance
  - Class
  - Constants
- Constructors
  - Default No Argument
  - Super Constructor
  - Constructor Chaining
- Initialisation Blocks
  - non-Static
  - Static
- Methods
  - Signature
  - Parameter List
  - Parameter Type
    - [ByValue]
    - Primitive
    - Arrays
    - VarArgs
    - Object Variable
    - Interface Variable
    - Method Ref.
    - Lambda Expression
  - Ambiguity
  - Scope / Access
  - Covariant Return Type
- Extending
  - Compatibility
- Interface Implementation
  - Single
  - Multiple
  - Generic

**Instantiation**

- Declaration
- Allocation
- Initialisation
- Variable Referencing
- Member Referencing
- Garbage Collection

## Anonymous Class

- Header / Body Syntax
- Anonymous Object
  - Extended Class
  - Inline Implementation
- Access to Outer Scopes

## Lambda Expression

- Purpose / Intended Use
- Functional Interface
- Parameters / Body Syntax
  - Zero Parameters
  - Multiple Parameters
  - Explict Parameters
  - Implicit Target Type
- Access to Outer Scopes

## Enum

**Declaration / Definition**

- Header / Body Syntax
- Enum Constants
- Enum Constructor
- Memory Compostion

**Instantiation**

- Declaration
- Referencing
- Restrictions

## Interface

**Declaration / Definition**

- Header / Body
  - Structure
  - Syntax
- Memory Composition
  - Implicit Access Modifiers
- Members
  - Permitted
  - Unpermitted
- Fields
  - Constants Only
- Methods
  - Abstract
  - Static
  - Default
- Exending
  - Multiple Inheritance
  - non-Static Members
    - Aggregation
    - Non-Ambiguity
    - Non-Clashing
  - Consolidation

**Class Implementation**

- Abstract Method Implementation
- Abstract Method Aggregation
- No Ambiguity

**Interface Variables**

- Polymorphism
- Anonymous Objects
- Compatibility

**Types**

- Normal
- Functional
- Semantic
- Annotation

## Arrays

- Declaration / Allocation / Initialisation
- Utility Classes
  - System
  - java.util.Arrays
- Utility Methods
  - Sorting
  - Collection Conversion
  - Searching
  - Copying
  - Comparison

## Blocks

- Permitted Usage
- Permitted Members
- Unpermitted Members

- Initialisation Blocks
- Initialisation Blocks
- Labelled Blocks

## Annotations

- Declaration
  - Elements
- Deployment
  - Single
  - Multiple / Repeated
- Types
  - Annotation [Predefined]
  - Annotation Type [Custom]
  - Container Annotation Type
  - Meta-Annotations
  - Type Annotation

## Static / non-Static Memory

**Component**

- Memory Composition
- Memory Depiction
- Memory Decpiction within nested components
- Memory Scope
- Memory Properties
  - Internal Composition
  - Location

**Static Memory**

- Permitted Members
  - Static Member Initialisation
  - Static Member Default Values
  - Static Member Referencing
- Permitted Referencing
- Nested Components
  - Nested Referencing
  - Outer Scope Referencing
  - Shadowing

**non-Static Memory**

- Permitted Members
- Permitted Referencing
- Default Values
- Nested Components
  - Nested Referencing
  - Outer Scope Referencing
  - Shadowing

## Exceptions

- Checked / unchecked
- Chained exceptions
- Catch / specify requirement
- Throwable
- Exception
- try-catch-finally
- try-with-resources
- RuntimeException
- Error

## Pipelines / Streams

- Aggregate Operations:
  - Source
  - Intermediate Operations
  - Terminal / Reduction Operations
- Ordering
- Laziness
- Interference
- Aggregate Operators v Iterators
- Collection Traversal
- Low Level Operation
- Side Effects

## Techniques and Data Structures

**Dynamic Programming**

- 1D
- 2D
- Top-Down
- Bottom-Up / Tabulation

- Divide and Conquer
- Greedy
- Backtracking
- Path / Level Tracking
- Sliding / Dynamic Window
- Binary Search
- Big O (Time / Space)

- Linked List
- Stack
- Queue
- Deque
- Heap
  - Min Heap
  - Max Heap
  - Priority Queue
- Hash Table / Map
- Prefix Array
- Suffix Array
- Disjoint Set / Union Find

**Recursion**

- Recursive Method Structure
- Preprocessing / Postprocessing
- Base / Ongoing Case
- Call Tree
- Tail Recursion

**Graphs / Trees**

- Binary Tree
- Binary Search Tree
- Balanced Binary Search Tree
- Minimum Spanning Tree
- n-ary Tree
- Trie

**Graph / Tree Traversal**

- Directed
- Undirected
- Acyclic
- Edge List
- Adjacency List
- BFS / DFS
- preOrder
- inOrder
- postOrder

## Generics

### Application

- Class
- Abstract Class
- Interface
- Enum (Constructor)
- Constructor
- Method

### Scope

- Local
- Class / Interface

### Generic Class

- Declaration
  - Header / Body Syntax
  - Class Type Parameters
  - Local Type Parameters
  - Extension and Type Pass Up
  - Multiple Type Parameters
  - Hardcoded Type Parameters
  - Hierarchical Compatibility
- Invocation, Instantiation and Initialisation
  - Syntax
  - Parameterised Types
  - Type Inference
    - Diamond Operator
    - Raw Types (Object)

### Generic Constructor / Method

- Class Type Parameter Referencing
- Local Type Parameter Referencing
- Type Parameter Scope
- Invocation
  - Type Witness Omission
  - Type Inference

### Generic Interface

- Declaration
  - Header / Body Syntax
  - Interface Type Parameters
  - Local Type Parameters
  - Extension and Type Pass Up
    - Aggregation, Override and Overload
    - Multiple Inheritance
    - Generic / Non-Generic Inheritance
    - Non-Ambiguity
  - Interface Consolidation
    - Multiple Inheritance / Extension / Implementation
- Class Implementation
  - Class Header / Body Syntax
  - Multiple Interface Consolidation
  - Non-Ambiguity
  - Type Argument Specification
    - Generic Type
    - Hardcode
    - Object

### Type Arguments

- Bounding
  - Wildcards
  - Upper
  - Lower
  - Unbounded
- Restrictions
- Compatibility
- Extension Substituition

### Restrictions

- No Primitive Types
- No Instantiation
- No Static Fields
- No Arrays
- No Overloading (ambiguity)
- No Relational Operators
- No Casting (unless valid)

### Type Parameters

- Bounding
  - Upper
  - Unbounded
  - Minimum Implementation
  - Multiple Bounds
- Restrictions
  - Erasure
  - Type Naming Convention

## Collections

### Interface

- Collection
- Map
- List
- Queue
- Set
- Deque
- Comparable
- Comparator
- Iterator
- ListIterator

### Class

- ArrayList
- HashMap
- LinkedList
- LinkedHashMap
- HashSet
- TreeMap
- LinkedHashSet
- ArrayDeque
- TreeSet

- Overview / Benefits
- Interface Properties / Characteristics
- Modifiable / Unmodifiable
- Mutable / Unmutable
- Optional / Unsupported Methods
- View Collection
- Serializability
- Restrictions
- Optional / Unsupported Methods
- View Collection
- Traversal
  - Streams / Pipelines
  - For-Each / Iterators
- Bulk Operations
- Conversions
  - Collection / Array
  - Conversion Constructors

## Design Patterns

- Abstract Factory
- Adapter
- Bridge
- Builder
- Ch. Responsibility
- Command
- Composite
- Decorator
- Facade
- Factory Method
- Flyweight
- Interpreter
- Iterator
- Mediator
- Memento
- Observer
- Prototype
- Proxy
- Singleton
- State
- Strategy
- Template Method
- Visitor

## Multithreading / Concurrency

### Interface

- Runnable
- Callable
- Future
- Lock
- Condition
- ExecutorService
  - SingleThreadExecutor
  - FixedThreadPool
- ScheduledExecutorService
  - ScheduledThreadPool
- BlockingQueue<E>
- ConcurrentMap<K,V>

### States

- Livelock / Deadlock

### Class

- Thread
- ReentrantLocks
- Semaphore
- Executors
- CountDownLatch
- CyclicBarrier
- AtomicInteger
- ConcurrentHashMap<K,V>
- Exchanger<V>
- PriorityBlockingQueue<E>
- PriorityBlockingQueue<E> with Comparable Element

### Techniques

- Synchronisation Blocks
- Object Locks
- Wait / Notify
- Object Locks with Conditions
- Volatile Memory
- Producer / Consumer

### Fork / Join

- Class
  - ForkJoinPool
  - RecursiveAction
  - RecursiveTask<V>
- Sequential v Parallel (via Fork / Join)
  - Find Max
  - Mergesort

### Serial v Parallel

- Mergesort
- Find Sum
- Streams

## Packages

- Management / Organisation
- Naming Conventions
- Referencing
- Importing Static Members
- Import Wildcards
- Importing Top Level Components

## SOLID Principles

- Single Responsibility
- Open-Closed
- Liskov Substitution
- Interface Segregation
- Dependency Inversion

## JRE / JVM

- JIT Compiler
- Class Loaders
  - Bootstrap
  - Extension
  - System / Application
- Memory Allocation
  - Heap
  - Stack
  - Program Counter
- CLASSPATH
  - Source Directory
  - Java API Library

## Miscellaneous

- final
- null
- super
- this
- instanceOf
- .equals()
- .hashCode()
- Constructor Chaining

## Spring

**Projects**

- Spring Boot
- Spring MVC
- Spring Validation
- Spring Data
- Spring Batch / Integration
- Spring Security
- Spring Security For OAuth
- Spring Security Authorisation Server

**Bean Declaration / Definition**

- via XML
- via Annotation
- via Bean Method (@Bean)
- via Component Scanning (@Component)
- via Configuration Classes (@Configuration)

**Dependency / Bean Injection**

- via Constructor
- via Setter
- via Field
- via Autowiring

**Spring MVC Web Application / REST Endpoint**

- Embedded Tomcat Server
- Thymeleaf Templates

- @Controller
- @RestController
  - @RequestMapping
  - @GetMapping
  - @PostMapping
  - @PutMapping
  - @DeleteMapping
  - @PatchMapping
  - @SessionAttributes
  - @ModelAttribute
  - RestTemplate
  - ResponseEntity
  - Object Mapping
  - JSON / XML Payload
  - Pagination
  - Cross Origin Resource Sharing
  - Path Variables
  - HATEOAS
  - @ResponseBody
  - @ResponseStatus

**Testing**

- @SpringBootTest
- @WebMVCTest
- @Test

**Dev Tools**

- Auto Restart
- Auto Refresh
- No Caching
- H2 Console

**Spring Initializr**

- Project Creation / Structure
  - Source Code
  - Resources
  - Test
  - application.properties
  - application.yml
  - Manifest File
  - Executable *.jar / *.war
  - Mavern / Gradle Build
- Starter Dependency Selection

**Spring Tool Suite**

- IDE Plugin
- Spring Boot Dashboard

**Configuration**

- DSL Configuration
- Configuration Properties
- Profiles

**Lombok**

- @Data (Data Class)
- Getter / Setter Auto Populate

**Logging**

- @Slf4j
- Logback

**Persistence**

- Spring Data JDBC/JPA/...
- JDBCTemplate
- Schema (via *.sql)
- SpEL
- @Repository
- @Table
- @Data
- @Id
- @Query

**Security**

- User Authentication
- Security Filter Chains
- 3rd Party Authentication
- OAuth2
- JWT
- OpenIDConnect
- Cross Site Request Forgery
- Client Repositories

**Messaging**

- Asynchronous Brokers
  - JMS
  - RabbitMQ
  - Kafka
  - JMSTemplate
  - RabbitMQTemplate
  - KafkaTemplate
- Push / Pull Models
- Message Converters
- Message Header / Payload
- Message Listeners

## XML HTTP REST

**XML**

- Purpose
- Standards
- XML Document
  - Prologue
  - Elements
    - Tags
    - Attributes
    - Root
    - Siblings
    - Entity Reference
  - Well Formed
  - Comments
  - Namespaces
- XMLHttpRequest
- XML Parser
- XML DOM
- XPath
- XSLT
- XQuery
- XLink
- DTD / XML Schema

**HTTP**

- Purpose
- Properties
  - Connectionless
  - Media Independent
  - Stateless
  - Versions

**MIME Type**

- Format
- Components
  - Type
  - Tree / Subtype
  - Suffix
  - Parameters
- Registration Trees
  - Standards Tree
  - Vendor / Producer Tree
  - Personal / Vanity Tree
  - Unregistered Tree

**REST API**

- Purpose
- Client / Server
- Stateless
- Uniform Interface
  - Resource Identification: URI
  - Resource Manipulation: GET, PUT, POST, DELETE...
  - Resource Description: Content-Type: application/json

# Kotlin

## Class

**Declaration / Definition**

- Constructors
  - Primary
  - Secondary
  - Init Block
- Properties
- Member Functions

- Extension
- Interface Implementation
- Delegation (by)
- Operator Overloading

**Properties**

- Backing Field
- get() set()
- value field
- Lazy
- lateinit
- val var
- Default value
- Delegation (by)

**Extension Properties**

- Creating
- Referencing
- Receiver (via this)

**Data Class**

- Purpose
- Creation
- Built-In Implementations
  - .toString()
  - .equals / ==
  - .hashCode()
  - .copy()
  - .println()
  - .component1()...
- Copying
- Destruction Declarations

**Enum Class**
**Sealed Class**
**Nested / Inner Class**

**Generics**

- Classes
- Interfaces
- Functions
- Extension Functions

- Type Arguments / Parameters
- Bound / Unbounded
- Nullable / non-Nullable

## Collections

List Mutable / Read-Only
Map Casting
Set

**Extension Functions**

| | |
|---|---|
| .filter() | .any() |
| .map() | .all() |
| .mapNotNull() | .none() |
| .find() | .associate() |
| .first() | .associateBy() |
| .firstOrNull() | |
| .count() | .flatten() |
| .partition() | .flatMap() |
| .groupBy() | |
| .groupingBy() | .zip |
| .maxBy() | .zipWithNext() |
| .minBy() | |
| .getOrPut() | |
| .sortByDescending() | |

## Objects

| | |
|---|---|
| Purpose | Declaration |
| Singleton / static | Referencing |
| Object Expressions | Companion |

## Language

History
Purpose
Java / JVM Interoperability
Java Interpretation
Java Equivalents

OOP / Functional Styling
Statically Typed
Concision
Modules / Packages
Top Level

equals() / == / ===
Constants
Pairs

Inlining
Arrays

**Access Modifiers**

| | |
|---|---|
| private | internal |
| protected | public |

**Loops**

| | | |
|---|---|---|
| in | downTo | Range |
| until | step | .. |

**Types**

| | | |
|---|---|---|
| Type Inference | Type Casting | Unit / Nothing |
| is / as / as? | Smart Casting | val / var |
| .let() | ? | Any |

**Conditionals**

- Expressions
- Comparisons
- when
  - if-else chain
  - Type Checking
  - Ranges
  - Enum
  - Pairs

**Strings**

- Templates
- Multiline
- Data Type Conversion
- Concatenation

**Common Library Functions**

| | | |
|---|---|---|
| .takeIf() | .use() | .withLock() |
| .takeUnless() | .with() / .run() | .apply() / .also() |
| .repeat() | | |

**Nullable / non-Nullable**

| | |
|---|---|
| Purpose | Elvis Operator |
| Safe Call | non-Null Assertion !! |

Java / Kotlin Interoperatbility
- via Annotation
- via Explicit Type Specification
- via Intrinsic Checks
- NPE Safety
- Platform Types

**Exceptions**

- Structure / Form / @Throws
- try catch
- Assignable
- Function Wrappers
- .require()

**Common Annontations**

| | |
|---|---|
| @JvmName | @JvmStatic |
| @JvmOverloads | @JvmField |

## Functions

| | | |
|---|---|---|
| Top-Level | Anonymous | Extending |
| Member | Local | Overriding |

**Forms**

| | |
|---|---|
| As Variable | Named Parameters |
| As Parameter | Default Arguments |
| As Return | Function Expressions |

**Function Types**

| | |
|---|---|
| Implicit / Explicit | nullable / non-nullable |

**Member References**

- Bound / Unbound

**Extension Functions**

| | |
|---|---|
| Purpose | Limitations |
| Creating | Invocation from Java |
| Managing | infix |

## Sequences

- Purpose
- Stream Equivalent
- Collection Alternative

- Intermediate Operations
- Terminal Operations

- Lazy
- Yield

- .asSequence()
- .generateSequence()

## Lambda Expressions

- Purpose
- Structure { }
- Chained Statements (Functional Styling)
- Trailing Lambda
- Destruction Declaration
- .run()

**Forms**
- As Variable
- As Argument
- As Return
- As Run / Invocation

**Return Control**
- via Labelling
- Whole Function

**Parameters**
- None
- Blanked _
- Single / it
- Multiple

**Lambda (with Receiver)**
- Purpose
- Structure / Difference
- Extension Function / this

## Workflows / Build Tools

**Continuous Integration > Continuous Delivery / Deployment**
- Low Risk
- Progress
- Automation
- User Feedback

**Test Automation**
- Unit Test Suite
- Regression Test Suite
- Performance Test Suite

**DevOps**
- Purpose
- Advantages
- Pipeline
  - Idea > Code > Build > Deploy > Manage > Learn > Idea...
  - Velocity
  - Quality
- Value Stream Map

**Scrum**
- Purpose
- Team / Roles
  - Product Owner
  - Scrum Master
  - Developers
- Sprint Events / Workflow
  - Plan
  - Development
  - Review
  - Retrospective
- Artifacts
  - Product Goal
  - Product Backlog
  - Increment (of Value)
- Sprint Backlog
  - Sprint Goal
  - Items
  - Plan of Delivery
  - Burndown Chart

**Maven / Gradle**
- Purpose
- Project Structure
  - *.pom / build.gradle
  - Modules
  - Dependencies
  - Plugins
    - Maven WAR
    - Maven Cargo
- Build Lifecycle
- mvnrepository.com
- Goals / Tasks

**TDD**
- Cycle: Red > Green > Refactor > Red...
- Unit Tests
  - Solution Space
  - Output Space
  - Constraint
  - Certainty / Flexibility
  - Uncertainty Principle
  - Value / Property Testing
- Test Patterns
  - Self Shunt
  - Humble Object
- Test Doubles
  - Dummy
  - Stub
  - Spy
  - Mock
  - Fake

## Git / GitHub

**Version Control**

| | |
|---|---|
| VCS | Version Control Systems |
| CVCS | Centralised Version Control Systems |
| DVCS | Distributed Version Control Systems |
| DBVC | Delta Based Version Control |

**Repository**
- Local / Remote
- Clone
- .git Folder
- .gitignore
- Patch/Patch Set
- Staging Area
- Diff
- Directed Acyclic Graph
- git Configuration
  - Global
  - User
  - Repo
- File Status
  - Untracked / Ignored
  - Tracked
  - Modified
  - Staged
  - Staged + Modified
- Working Directory
  - Clean/Dirty
  - Stashing
- Revision
- History
  - Search Metadata
  - Reflog
- Submodules
- Pull Requests

**Commits**
- Snapshot
- Patch + Metadata
- Hash ID (Raw Reference/Short Link)
- Granular / Clarity
- Best Practices / Considerations

- GitHub Desktop
- GitHub/Git CLI

**Branching**
- Main
- Feature
- HEAD
- Local Branching
- Remote Branching
- Creation
- Checkout
- Switching
- Renaming
- Show/Status
- Push / Pull
- Tracking
- Deletion
- Reset
- Merging
  - Fast Forward
  - Merge Commit
  - Conflict
  - Abort
- Compare
- Rebase
- Interactive Rebase
- Cherry-Picking
- Upstream / Downstream
- Best Practices / Considerations

# Cloud

## Architectures

**Monolith**
- Advantages
- Disadvantages

**Microservices**
- Advantages
- Disadvantages
- Characteristics
- Inter-Communication
  - Request / Response
  - Event Driven
    - Event Messaging
    - Event Streaming
- Design Patterns
  - Backend-for-frontend (BFF)
  - Entity and Aggregate
  - Service Discovery
  - Adapter
- Design Anti-Patterns

**Serverless**
- Advantages
- Disadvantages
- Abstraction Chain

**Compute Models**
- On-Prem
- IaaS
- PaaS
- FaaS
- SaaS

**Cloud Service Providers**
- Amazon
  - AWS
  - AWS Elastic Beanstalk
  - AWS Lambda
- Microsoft
  - Azure
  - Microsoft Windows Azure
  - Azure Functions
- Google
  - Google Cloud / GCP
  - Google App Engine
  - Google Cloud Functions
- IBM
  - IBM Cloud
  - IBM Cloud Code Engine
- Oracle
- Heroku
- VMWare

## Kubernetes

**Cluster**
- Control Plane
  - cloud-controller-manager
  - kube-controller-manager
  - kube-apiserver
  - kube-scheduler
  - etcd
- Node(s)
  - kubelet
  - k-proxy
  - Container Runtime
    - Docker Engine
    - CRI-O
    - Containerd
    - Mirantis Container Runtime
  - Objects
    - Configuration: *.yml
    - Pod
    - Deployment
      - Pod Template
    - StatefulSet
    - ReplicaController
    - Volume
    - PersistentVolume
    - PersistentVolumeClaim
    - Secret
    - Service
      - ClusterIP
      - NodePort
      - Load Balancer
      - Ingress
  - Label Selector System
  - Environment Variables
  - Role Based Access Control

**Cluster Admistration:**
- kubectl
- kubeadm
- minikube

**Controller(s)**
- Node Controller
- Job Controller
- Endpoints Controller
- Service Account Controller
- Token Controller

**Cloud Integration**
- CI / CD Workflow
  - Local > GitHub > Test Suite > DockerHub > Cloud Service Provider
- Travis CI
  - .travis.yml
- Cloud Service Provider
  - Configuration
  - Integration
  - Account Verification
  - Environment Variables
  - Logs / Monitoring
- Declarative / Imperative

## Docker

**Core**
- docker-server
- docker-client
- docker-compose
- dockerHub

**Container**
- Resource Segmentation
- Start / Stop
- Status / Monitoring
- Communication Channels
- Environment Variables
- Logging / Exiting

**Image Build**
- DockerFile
  - Base Image
  - Dependencies
  - Startup Command
  - Development
  - Production
- docker-compose
  - docker-compose.yml
  - Build Context
  - Build Cache
  - Networking
  - Port Mapping
  - Restart Policy
  - Volumes

**Image**
- File System
- Startup Command

## Android

### Android Studio

**Project Structure / Files**

- Source Code
- Resources
- Libraries
- Gradle
- Manifest File
- APK file

**UI / Layouts**

- Code Editor
- Design Editor

**Emulation**

- USB Direct
- AVD

**UI**

- Layout XML
- LinearLayout
- FrameLayout
- ScrollView
- Padding/Margin
- Weighting
- Gravity
- Themes
- AppBar
- Toolbar
- Composable
- CoordinatorLayout
- AppBarLayout
- CollapsingToolbarLayout
- Bluprints
- Layout Inflation
- Layout Nesting
- Collapsing Toolbar
- Scrolling Toolbar
- Material Design
- ConstraintLayout
  - Constraints
  - Bias
  - Guidelines
  - Barriers
  - Chains
  - Flows
- Navigation Bar
- Navigation Drawer

**Architectures**

- MVC
- MVI
- MVVM
- MPC

**Views**

- TextView
- Button
- Checkbox
- Chip/Groups
- Spinner
- Radio Button/Groups
- Floating Action Button
- Toast
- Snackbars
- View Groups
- View Binding
- View Models
- View Model Factories
- View Model Provider
- Compose
- Live Data
- Mutable Live Data
- Data Binding

- Intents
- Broadcasts
- Services
- Work Manager

**Activity**

- Lifecycle State / Methods
- Lifecycle (Visibility)
- Lifecycle (Foreground)
- Save / Restore State
- Bundle
- Device Rotation

**Multiscreen**

- Fragments
- Fragment Lifecycle
- Navigation Component
- Navigation Graphs
- Navigation Host
- Navigation Controller
- FragmentContainerView
- Actions
- Safe Args / Directions / Args
- Back Stack

### HTML / CSS / PHP / mySQL

**nationalsyndicate.org.uk**

- In 2004-2005 completed a substantial personal project of creating an online lottery syndicate.
- Users would join and create a subscription.
- Users would declare their desired ball combinations to be used as entries within the syndicate.
- All of the syndicate entries would be bulk purchased securely and electronically for each draw.
- The excitement and appeal would be generated by being part of a syndicate of potentially thousands, if not hundreds of thousands of entries.
- Built using HTML / CSS / PHP / mySQL and PayPal.
- Front and backend fully operational.
- Unfortunately it did not go live due to Camelot not willing to provide support for secure electronic bulk ticket purchasing.
- Nonetheless it provided a significant and enjoyable learning experience.

# Research Materials

Please find a summary of the primary resource materials used for the research and study of the subject areas listed above:

### Primary Online Resources

| | | |
|---|---|---|
| Java SE | Oracle Java Tutorials | https://docs.oracle.com/javase/tutorial/index.html |
| | Oracle Java API | https://docs.oracle.com/javase/8/docs/api/index.html |
| Spring | Online Documentation | https://docs.spring.io/spring-framework/docs/current/reference/ |
| Kotlin | Online Documentation | https://kotlinlang.org/docs/home.html |

### Udemy Courses

- Java Programming Masterclass
- Design Patterns in Java
- Concurrency, Multithreading and Parallel Computing in Java
- Java Memory Management
- Java Application Performance and Memory Management
- Java Reflection
- Java Spring Tutorial Masterclass – Spring Framework 5
- Dynamic Programming and Data Structures
- Test Driven Development
- The Complete Oracle SQL Bootcamp

### Coursera Courses

- Kotlin for Java Developers by JetBrains

### W3Schools

- XML Tutorial

### LeetCode

- Data Structures and Algorithms
- Dynamic Programming
- Bit Manipulation
- 150+ Questions Completed

### Bibliography


| | | | |
|---|---|---|---|
| Java The Complete Reference | 8th Ed. | Herbert Shildt | Oracle Press |
| Java Cookbook | 4th Ed. | Ian F Darwin | O'Reilly |
| Spring in Action | 6th Ed. | Craig Walls | Manning |
| Cloud Native Java | 1st Ed. | Josh Long and Kenny Bastani | O'Reilly |
| Android Development | 3rd Ed. | Dawn and David Griffiths | O'Reilly |
| Pro Git | 2nd Ed. | S.Chacon B.Straub | Apress |
| Design Patterns | 1st Ed. | E.Gamma R.Helm R.Johnson J.Vlissides | Addison Wesley |
| Clean Architecture | 1st Ed. | R.C.Martin | Prentice Hall |
| Clean Craftsmanship | 1st Ed. | R.C.Martin | Prentice Hall |
| The Clean Coder | 1st Ed. | R.C.Martin | Prentice Hall |