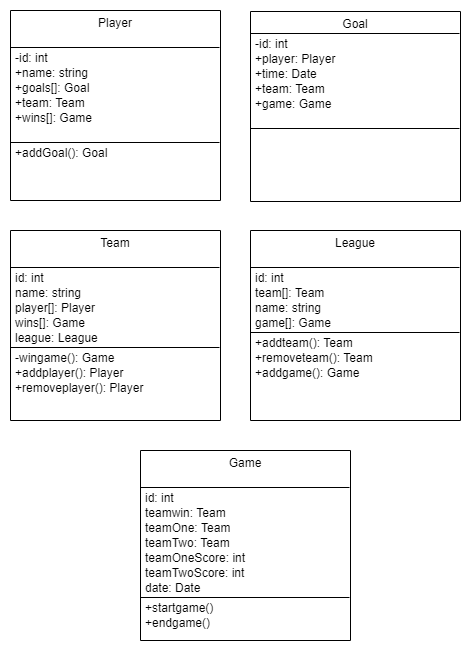
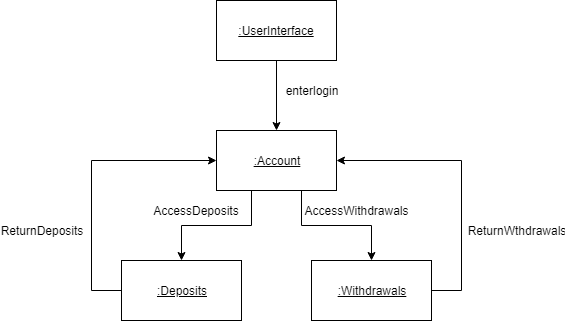
## Class Diagrams

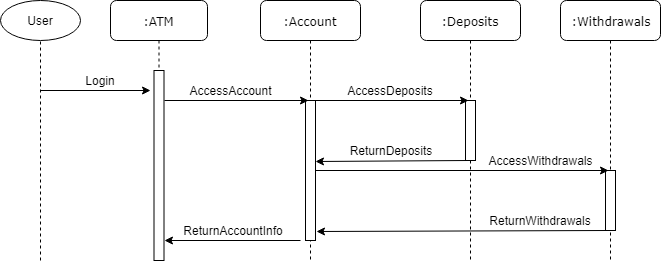


Class diagrams are used to define the structure and layout of classes that the program is utilising. They lay out the various attributes and methods that each class contains and provides a basic blueprint that allows the developer to map code from.

## Collaboration and Sequence Diagrams

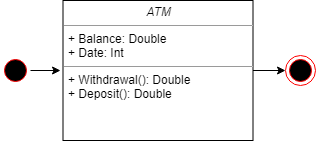


Collaboration diagrams show how objects interact and communicate with each other within the program. This is not considered a step by step process and is displayed as a broad interaction between instances.

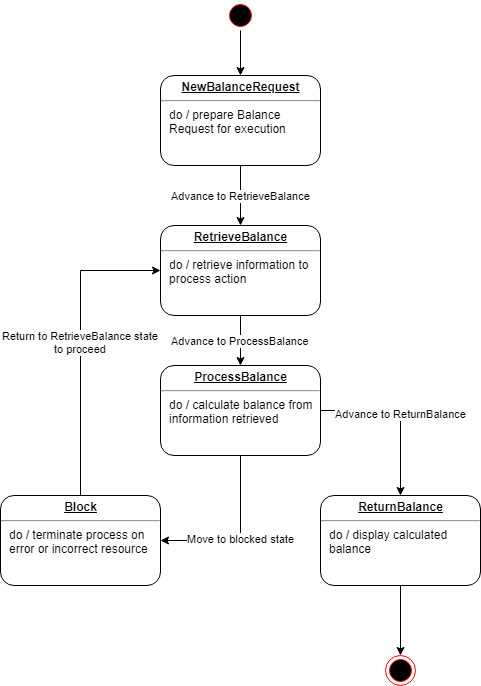


Sequence Diagrams represent the interaction between objects in logical steps through the flow of information between objects. Each step will consist of an interaction between objects, and display the directional flow of specific information.

## State and Activity Diagrams

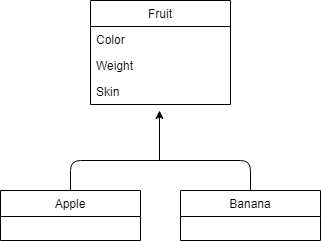


State diagrams will display the phases that a specific object will go through as a result of the actions that take place during the operation of the object. The diagram will display three states; the entry state, which shows indicates the actions of the object as it enters this state, the do state, which indicates the actions that the object will take while it is in the displayed state, and the end state, which indicates the actions that the object will take when it leaves this state.



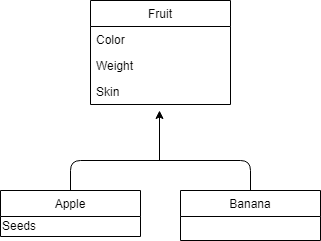
Activity Diagrams will display the process that an object will undertake when performing various actions. The program will begin in the starting state, marked by the single dot. The object will then perform the stated action, and then change state in order to perform the next action that the object undertakes. This will continue until all stated actions and completed and the objects enters its end state, signified in the diagram with a bullseye

# Generalisation and Specialisation



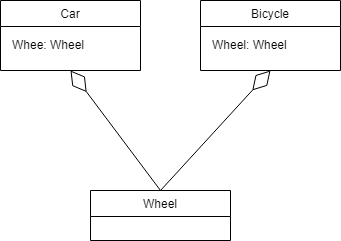
Generalisation and specialisation represent the directional flow of certain types of information between sub-classes and super-classes.

Generalisation represents the upward flow of information, from sub-classes, to superclasses. The information from those subclasses needs to generally be considered the same, or at least similar enough to be combined.

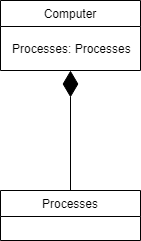


Specialisation is the flow of information from super-classes into sub-classes. This is similar to the inheritance concept, where sub-classes of a super-class will adopt the characteristics present in the superclass, however one of the subclasses will require additional characteristics that other subclasses will not. This creates a ‘specialised’ subclass, suitable for specific actions and tasks that the other subclasses are not adapted for.

# Aggregation and Composition



Aggregation is a specialised relationship between multiple classes. The aggregate class will contain a reference to a different class, and also has ownership over that class. As a result, each class that is referenced in the aggregate class is a part of the aggregate class. The reference within an aggregate class can only be one-directional, if both classes referenced each other, then they would be considered associated classes rather than aggregate. Aggregated classes exist independently of each other, they do not depend on one another to have functionality within the program and can continue to exist as separate entities within the bounds of the system.



Composition is a relationship derived from an order of necessary attribution. A class will only exist or serve a purpose while the composite class exists and serves a purpose. This is also a unidirectional flow of information that is best used to define abstract data models. If the component class inherits information from the composite class then it ceases to be a component-composite relationship.

## GUI Interaction

Java Swing is a set of integrated components within the java packages that are used when created graphical user interfaces and adding functionality via interaction and use within GUI’s. There are 5 main features that comprise Java Swing:

* GUI Components
* Pluggable Look and Feel Support
* Accessability API
* Java2D API
* Internalisation

Java FX is a set of integrated graphics and media packages that exist within the Java JDK that allow developers to employ a variety of techniques and features to design diverse client applications that operate consistently within the Java runtime environment.

## Coupling and Cohesion

Coupling and cohesion represent the strength of the relationships between elements and modules within a program. Usually, coupling and cohesion will have an inverse connection; low coupling will often have corresponding high cohesion and vice-versa.

Coupling represents the level of interdependence of software elements, and the level of interactivity they require to function. Cohesion represents the level of ownership that elements within a program have and is generally preferable to coupling as it increases readability, reliability and reusability of the code.

## SDLC’s

Software development life cycles are methodologies that provide an organisational blueprint for the development of software. There are many different SLDC’s each with their own associated benefits and risks, and should be chosen and implemented carefully based upon a number of factors such as the scope of works, project size, team size, associated risks, overheads etc. Several of the most common SDLC’s are:

* Waterfall Model
* V-Shape Model
* Iterative Model
* Spiral Model
* Agile Model
* Prototyping Model