**Research**

1. **Explain the order of lifecycle changes in activity a and activity b as activity a starts activity b.**

In the Car Registration project, I was able to run log during its run time. Therefore, the order of lifecycle for changes in activity a and b are recorded.

The order of Android lifecycle is as follow:

MainActivity onCreate() -> MainActivity onStart() -> MainActivity onResume() -> MainActivity onPause() -> DataEntryActivity onCreate() -> DataEntryActivity onStart() -> DataEntryActivity onResume() -> MainActivity onStop() -> DataEntryActivity onPause() -> MainActivity onRestart() -> MainActivity onStart() -> MainActivity onResume() -> DataEntryActivity onStop() -> DataEntryActivity onDestroy() -> MainActivity onPause() -> DataEntryActivity onCreate() -> DataEntryActivity onStart() -> DataEntryActivity onResume() -> MainActivity onStop() -> DataEntryActivity onPause() -> MainActivity onRestart() -> MainActivity onStart() -> MainActivity onResume() -> DataEntryActivity onStop() -> DataEntryActivity onDestroy() -> MainActivity onDestroy().

1. **How does serialization work?**

In Android, there are two ways to pass objects between activities. It can either implement Parcelable or Serializable interface. Serialization is a Java Interface that enables developers to implement the interface in order to send objects from one activity to another through Intent. It is an object passing technique that marks objects as serializable, so that these objects may get certain capability. When marked, it implies that the users cannot marshal the data according to their requirements. This marshaling/combining/arranging operation is done on a Java Virtual Machine (JVM) using the Java API, which helps identify Java objects member and behavior.

Serialization is a mechanism in place to convert the state of an object into a byte stream. The Serialization runtime associates a version number with each Serializable class called a SerialVersionUID, which is used during Deserialization to verify that sender and receiver of a serialized object have loaded classes for that object which are compatible with respect to serialization. If the receiver has loaded a class for the object that has different UID than that of corresponding sender’s class, the Deserialization will result in an InvalidClassException. A Serializable class can declare its own UID explicitly by declaring a field name.

1. **How does parcels work?**

Parcel is a container for a message, such as data and object references. Parcelable is an object passing technique in Android. It is a faster technique compared to Serialization, which is another type of passing objects between activities. By implementing Parcelable in Android allows objects to read and write from Parcels, which contains data. To convert an object into Parcelable, the developer will need to implement the Parcelable interface and to override the writeToParcel() methods in its own class. The developer will need to write all object members into parcel objects. After overriding the writeToParcel() method, the developer will then need to create a static Parcelable.Creator object to de-serialize the Java object. Since developers write a custom code for marshaling and unmarshalling in Parcelable, it creates less garbage objects in comparison to the Serializable technique.

1. **What is the difference in an implicit intent and an explicit intent?**

Implicit intents do not directly specify the Android components it should call, as it only specifies which action to be performed. When implicit intents are performed, Android system searches for all components that are registered for the specified action and data type. Explicit intents have the action and components Android should call. Explicit intents are often used to pass data (primitives and objects through Bundle and Parcelable technique) from one activity to another by using the putExtra() and getIntent().getExtras() methods. Explicit intents are always better to use if you know the component it’s going to and between activities. However, implicit intents are good for deep linking, broadcast receivers and content providers.

1. **What is the standard software development lifecycle?**

The standard Software Development Life Cycle (SDLC) is process standardized and used by the software development industry/community to design, develop and test high quality software. Its main goal is to produce/develop a high quality software that meets or exceeds the expectations of customers in regards to the requirements, time it takes and cost estimates.

The stages of a typical SDLC process are as follow:

* Planning and requirement analysis
* Defining requirements
* Designing the software architecture
* Product development
* Product testing
* Product deployment

The most popular SDLC models are the Waterfall model, Agile model, Iterative model, Spiral model and V-shaped model.