1. What is the difference between abstract classes and interfaces?
2. Describe the value of an interface and why are they important?

A: Interface allows for Simpler inheritance trees, provide flexibility to update it in future with minimum changes and a class can implements multiple interfaces.

1. What are anonymous classes?

A: An anonymous class is a local class without a name. An anonymous class is defined and instantiated in a single succinct expression using the new operator. While a local class definition is a statement in a block of Java code, an anonymous class definition is an expression, which means that it can be included as part of a larger expression, such as a method call.

1. What is the difference between using == and .equals on a string?

A: "==" operator compares the value of two object references to see whether they

refer to the same String instance. Where as .equals() method compares the "value"

inside String instances (on the heap) irrespective if the two object references

refer to the same String instance or not.( https://stackoverflow.com/questions/7520432/what-is-the-difference-between-vs-equals-in-java)

1. What is composition and what is inheritance in object-oriented programming? Why would you use one over another?  
   A: Composition mean using instance variables that are references to other objects. It expresses relationships between objects. Inheritance means classes can be derived from other classes, thereby inheriting fields and methods from those classes.
2. What is polymorphism and give an example where this is useful?  
   A: Polymorphism is the ability to present the same interface for differing underlying forms (data types).
3. When would you make an object value final? When you want to make an variable static?  
   A: Once a final variable has been assigned, it always contains the same value. If a final variable holds a reference to an object, then the state of the object may be changed by operations on the object, but the variable will always refer to the same object. You want to make a variable static to create a constant value that's attached to a class not any particular object of that class..
4. What is the difference between method overloading and overriding?

A: Overloading occurs when two or more methods in one class have the same method name but different parameters. Overriding means having two methods with the same method name and parameters (i.e., method signature). One of the methods is in the parent class and the other is in the child class. Overriding allows a child class to provide a specific implementation of a method that is already provided its parent class. Polymorphism applies to overriding, not to overloading. Overriding is a run-time concept while overloading is a compile-time concept.

1. What is serialization in Java (not Android)? How do you implement it? How about in Android?
2. Do objects get passed by reference or value in Java? Elaborate on that.

A: Java is pass by value only. The method parameter values are copied to another variable and then the copied object is passed, that’s why it’s called pass by value.

1. What are generics and why are they important in Java?

A: Generics allow type (Integer, String, … etc and user defined types) to be a parameter to methods, classes and interfaces. Generics are important as it allows Type-safety, Type casting is not required and it provides compile time checking.

1. What is one design pattern that you use and find helpful in Java/Android? Explain Activity lifecycle including most notable events and the order they run.

A: I would use MVVM pattern in Android wherein ViewModels would retrieve data from the model when requested from the view via the Android data binding framework. With this pattern, Activities and Fragments become very lightweight. Moreover, writing unit tests becomes easier because the ViewModels are decoupled from the view.   
As a user navigates throughout an app, Android maintains the visited activities in a stack, with the currently visible activity always placed at the top of the stack.At any point in time a particular activity can be in one of the following 4 states:  
Running - Activity is visible and interacting with the user, Paused- Activity is still visible, but no longer interacting with user, Stopped- activily is no longer visible ad Killed- Activity has been killed by the system (low memory) or its finish() method has been called.Most notable events are OnCreate(),OnStart(),OnResume,OnPause(),OnStop(),OnDestroy() and OnRestart().

1. Explain Fragment lifecycle including most notable events and the order they run.  
   A fragment is a reusable class implementing a portion of an activity. A Fragment typically defines a part of a user interface. Most notable events are onAttach,OnCreate,OnCreateView,OnViewCreated,OnActivityCreated,OnStart,OnResume,OnPause,OnDestroyView,OnDestroy and OnDetach.
2. How would you preserve Activity state during a screen rotation?

A: User can implement onSaveInstanceState() and placing the values you need to save in the Bundle argument that gets passed to the method.

1. What are Intents and what are the major types? When would you use one over another?

A: An intent is a message that can be thought of as a request that is given to either an activity within your own app, an external application, or a built-in Android service. Major types of intents are Explicit intent that is used to launch other activities within your application. Implicit Intents are requests to perform an action based on a desired action and target data. Explicit intents are typically used within an application as the classes in an application are controlled by the application developer. Implicit intents specify the action which should be performed and optionally data which provides content for the action.

1. What is the difference between ListView and RecyclerView?  
   A: ListView adapters do not require the use of the ViewHolder pattern to improve performance. In contrast, implementing an adapter for RecyclerView requires the use of the ViewHolder pattern for which it uses RecyclerView.Viewholder.   
   ListView can only layout items in a vertical linear arrangement and this cannot be customized. In contrast, the RecyclerView has a RecyclerView.LayoutManager that allows any item layouts including horizontal lists or staggered grids.   
   ListView contains no special provisions through which one can animate the addition or deletion of items. In contrast, the RecyclerView has the RecyclerView.ItemAnimator class for handling item animations.  
   istView had adapters for different sources such as ArrayAdapter and CursorAdapter for arrays and database results respectively. In contrast, the RecyclerView.Adapter requires a custom implementation to supply the data to the adapter.  
   ListView has the android:divider property for easy dividers between items in the list. In contrast, RecyclerView requires the use of a RecyclerView.ItemDecoration object to setup much more manual divider decorations.  
   ListView has a AdapterView.OnItemClickListener interface for binding to the click events for individual items in the list. In contrast, RecyclerView only has support for RecyclerView.OnItemTouchListener which manages individual touch events but has no built-in click handling.
2. What is the ViewHolder pattern? Why should we use it?  
   A ViewHolder object stores each of the component views inside the tag field of the Layout, so you can immediately access them without the need to look them up repeatedly. Code might call findViewById() frequently during the scrolling of ListView, which can slow down performance. Even when the Adapter returns an inflated view for recycling, you still need to look up the elements and update them. A way around repeated use of findViewById() is to use the "view holder" design pattern.
3. Explain all the major Android components (e.g activities, services, broadcast receivers, content providers) and how they each are used in apps. Explain the difference between Service, IntentService, and AsyncTask? When would you use one over another?  
   A: Activities- They dictate the UI and handle the user interaction to the phone screen.  
   Services - They handle background processing associated with an application.  
   Broadcast Receivers - They handle communication between Android OS and applications.  
   Content Providers - They handle data and database management issues.  
   IntentService is used for long tasks usually with no communication to main thread. If communication is required, one can use main thread handler or broadcast intents and also used, when callbacks are needed (Intent triggered tasks).  
   Services are used for tasks with no UI, but running shouldn't be too long. Usually used for Long task in general.  
   AsyncTask is mainly used for small tasks having to communicate with main thread or for tasks in parallel use multiple instances OR Executor or for disk-bound tasks that might take more than a few milliseconds
4. Explain what Looper, Handler and HandlerThread are and how they are used.  
   A: A Looper is a message handling loop: it reads and processes items from a MessageQueue. The Looper class is usually used in conjunction with a HandlerThread (a subclass of Thread). A Handler is a utility class that facilitates interacting with a Looper—mainly by posting messages and Runnable objects to the thread's MessageQueue. When a Handler is created, it is bound to a specific Looper (and associated thread and message queue).
5. Explain the difference between services and threads in Android? How and why are they used?  
   A: Service : is a component of android which performs long running operation in background, mostly with out having UI.  
   Thread : is a O.S level feature that allow you to do some operation in the background. If app needs to perform heavy background functionality then it is always better to have a service with thread. If the background functionality should be alive as long as the activity is alive, then use Threads.
6. Why should you avoid running non-ui code on the main thread?  
   A: To prevent from “Application Not Responding" (ANR) dialog thrown by Android anytime a app blocks main UI thread by performing lengthy operations on main UI thread which causes android application from responding to user inputs.
7. What are the options for persisting data in an Android app? When would you use the various different options available?  
   Shared Preferences - Easily save basic data as key-value pairs in a private persisted dictionary.  
   Local Files - Save arbitrary files to internal or external device storage.  
   SQLite Database - Persist data in tables within an application specific database.  
   ORM - Describe and persist model objects using a higher level query/update syntax.
8. What is the difference between a fragment and an activity? Explain the relationship between the two.  
   A: A fragment is a reusable class implementing a portion of an activity. A Fragment typically defines a part of a user interface. Fragments must be embedded in activities; they cannot run independently of activities.
9. How would you communicate between two Fragments? Describe the best practices for communication.  
   A: Fragments should generally only communicate with their direct parent activity. Fragments communicate through their parent activity allowing the activity to manage the inputs and outputs of data from that fragment coordinating with other fragments. Activity acts as the controller managing all interaction with each of the fragments. Fragment and an activity can communicate through   
   Bundle - Activity can construct a fragment and set arguments  
   Methods - Activity can call methods on a fragment instance  
   Listener - Fragment can fire listener events on an activity via an interface
10. What are "launch modes"? What are some common ways you might use launch modes.  
    A: Launch mode is an instruction for Android OS which specifies how the activity should be launched. It instructs how any new activity should be associated with the current task. These are common ways to set them:   
    standard - The default behavior. Creates a new instance of the activity in the task. singleTop - Reuse an activity instance if already at the top of the stack; otherwise create new instance.   
    singleTask - Reuse an activity instance if exists in an existing stack; otherwise create in a new task stack.   
    singleInstance - Same as singleTask but no other activities are ever inserted into the created task stack.
11. What is a BroadcastReceiver? What is a LocalBroadcastManager? What situations might you commonly use these?  
    A: A broadcast receiver (receiver) is an Android component which allows one to register for system or application events. All registered receivers for an event are notified by the Android runtime once this event happens.
12. What is a ContentProvider and what is it typically used for? Would you commonly use them in an app, and if not, why not?  
    A: Content providers are Android’s central mechanism that enables you to access data of other applications – mostly information stored in databases or flat files. Content providers support the four basic operations, normally called CRUD-operations. CRUD is the acronym for create, read, update and delete
13. How do you handle Bitmaps in Android and what are common issues associated with them? How do you address these issues?  
    A: Bitmaps have large memory demand which can immediately use up all the memory available to the app. Also, Loading bitmaps on the UI thread can degrade your app's performance, causing slow responsiveness or even ANR messages. It is therefore important to manage threading appropriately when working with bitmaps. Few ways to address this issue are to choose the most appropriate decode method based on your image data source, once image dimensions are known, they can be used to decide if the full image should be loaded into memory or if a subsampled version should be loaded instead.
14. What is the function of an intent filter? When would you use them in an app?  
    A: When you use an implicit intent, the Android system finds the appropriate component to start by comparing the contents of the intent to the intent filters declared in the manifest file of other apps on the device. If the intent matches an intent filter, the system starts that component and delivers it the Intent object. If multiple intent filters are compatible, the system displays a dialog so the user can pick which app to use.
15. What is the difference JVM, DVM and ART?
16. Have you used ConstraintLayout and what do you think of it?
17. What is the architecture you used for your last app? Describe how the files and components were organized and what worked well or could be improved.
18. What is Dependency Injection? Why is it useful? Can you name few libraries?
19. What do you think of RxJava? What are the benefits of using it? Are there any cons.
20. What is Android Data Binding? Do you like using this in your apps, why or why not?
21. What is an ORM? What ORM do you use? What is Room and how do you think this compares with other ORMs?
22. What is a SurfaceView? Why would you use one?
23. What is overdraw? Why is this important and how do you fix this?
24. Have you used Kotlin and what do you think of it?
25. How to avoid memory leaks in Android? How to discover them?
26. Name 3 libraries that you like to use in apps and why?
27. What’s a new Android SDK feature or API that was released recently that looked interesting or are playing with?
28. What is a JobScheduler? How does this differ from other alternatives?
29. How do you play sounds in Android?
30. Describe any experience or knowledge high level of unit testing. What is Espresso used for? What is Mockito used for? What is the difference between unit and instrumented tests?
31. What is obfuscation? What is it used for? What common tools enables this?