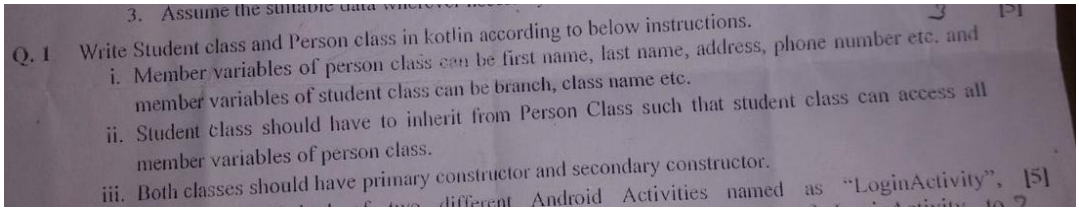


3. Assume the suitable data

Q. 1 Write Student class and Person class in kotlin according to below instructions. [5]

i. Member variables of person class can be first name, last name, address, phone number etc. and member variables of student class can be branch, class name etc.

ii. Student class should have to inherit from Person Class such that student class can access all member variables of person class.

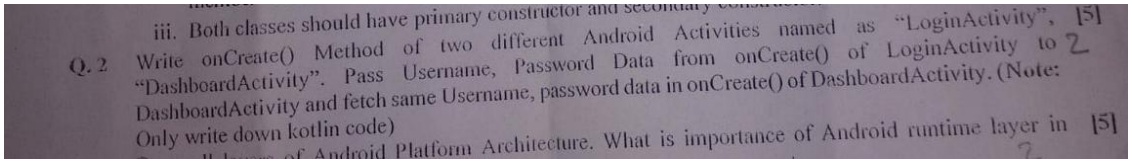
iii. Both classes should have primary constructor and secondary constructor.

different Android Activities named as "LoginActivity",

```
import org.jetbrains.annotations.NotNull
import kotlin.properties.Delegates

fun main()
{
    var s1 = Student("ABC","XYZ",1234567890,"Mehsana","IT(D2D)", "CEIT-C")
    s1.show_info()
    println("\n")
    var s2 = Student("ABC","XYZ ",12345678,"Mehsana","IT(D2D)")
    s2.show_info()
}
open class Person(var fname:String, var lname:String, var mobile_no:Long){
    lateinit var addr:String
    constructor(fname:String, lname:String, mobile_no:Long,
addr:String):this(fname,lname,mobile_no)
    {
        this.mobile_no=mobile_no
        this.addr=addr
    }
}
class Student(fname:String, lname:String, mobile_no:Long, addr:String, var
bname:String):Person(fname,lname,mobile_no,addr)
{
    lateinit var cls:String
    constructor(fname:String, lname:String, mobile_no:Long, addr:String,
bname:String, cls:String):this(fname,lname,mobile_no,addr,bname)
    {
        this.cls=cls
        println("Class Name is: ${cls}")
    }
    fun show_info()
    {
        println("First Name is: ${fname}")
        println("Last Name is: ${lname}")
        println("Mobile Number is: ${mobile_no}")
        println("Address is: ${addr}")
        println("Branch Name is: ${bname}")
    }
}
```

Output:

```
Class Name is: CEIT-C
First Name is: ABC
Last Name is: XYZ
Mobile Number is: 1234567890
Address is: Mehsana
Branch Name is: IT(D2D)


First Name is: ABC
Last Name is: XYZ
Mobile Number is: 12345678
Address is: Mehsana
Branch Name is: IT(D2D)
```



iii. Both classes should have primary constructor and secondary [illegible]

Q. 2 Write onCreate() Method of two different Android Activities named as "LoginActivity", "DashboardActivity". Pass Username, Password Data from onCreate() of LoginActivity to DashboardActivity and fetch same Username, password data in onCreate() of DashboardActivity. (Note: Only write down kotlin code) [5]

[illegible] of Android Platform Architecture. What is importance of Android runtime layer in [5]

MainActivity.kt

```
package com.example.mid1_q2

import android.content.Intent
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle

class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        Intent(this,MainActivity2::class.java).apply {
            putExtra("Uname","XYZ")
            putExtra("Password","123")
            startActivity(this)
        }
    }
}
```

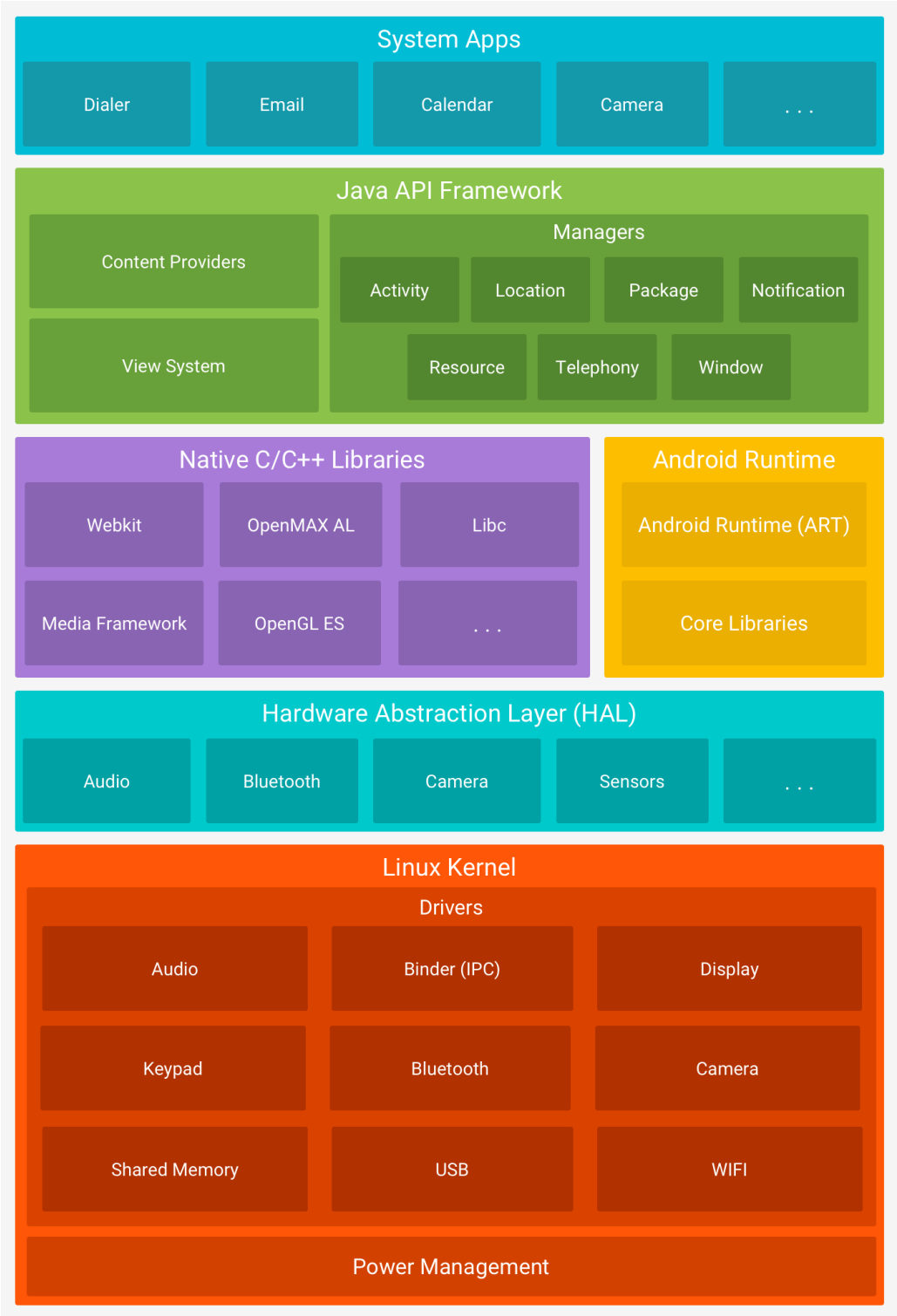
MainActivity2.kt

```
package com.example.mid1_q2

import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle

class MainActivity2 : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main2)

        var user_name=intent.getStringExtra("Uname").toString()
        var password=intent.getStringExtra("Password").toString()
    }
}
```

Only write down kotlin code)

Q. 3 Draw all layers of Android Platform Architecture. What is importance of Android runtime layer in Android OS? [5]

Draw flow chart of Activity Life Cycle and describe all functions of Activity Life Cycle. [5]



### 2.4 Android Runtime

Android Runtime (ART) is an application runtime environment used by the Android OS. ***A runtime environment is a state in which program can send instructions to the computer's processor and access the computer's RAM. Android apps programmed in (let's say) Java, will be first converted to byte code during compilation packaged as an APK and run-on runtime.***

Android uses a Virtual Machine to execute any application so as to isolate the execution of the program from OS and protecting malicious code from the affecting system.

*Before Android 4.4, that runtime used to be DVM (Dalvik Virtual Machine) which has since been replaced by Android Runtime (ART).*

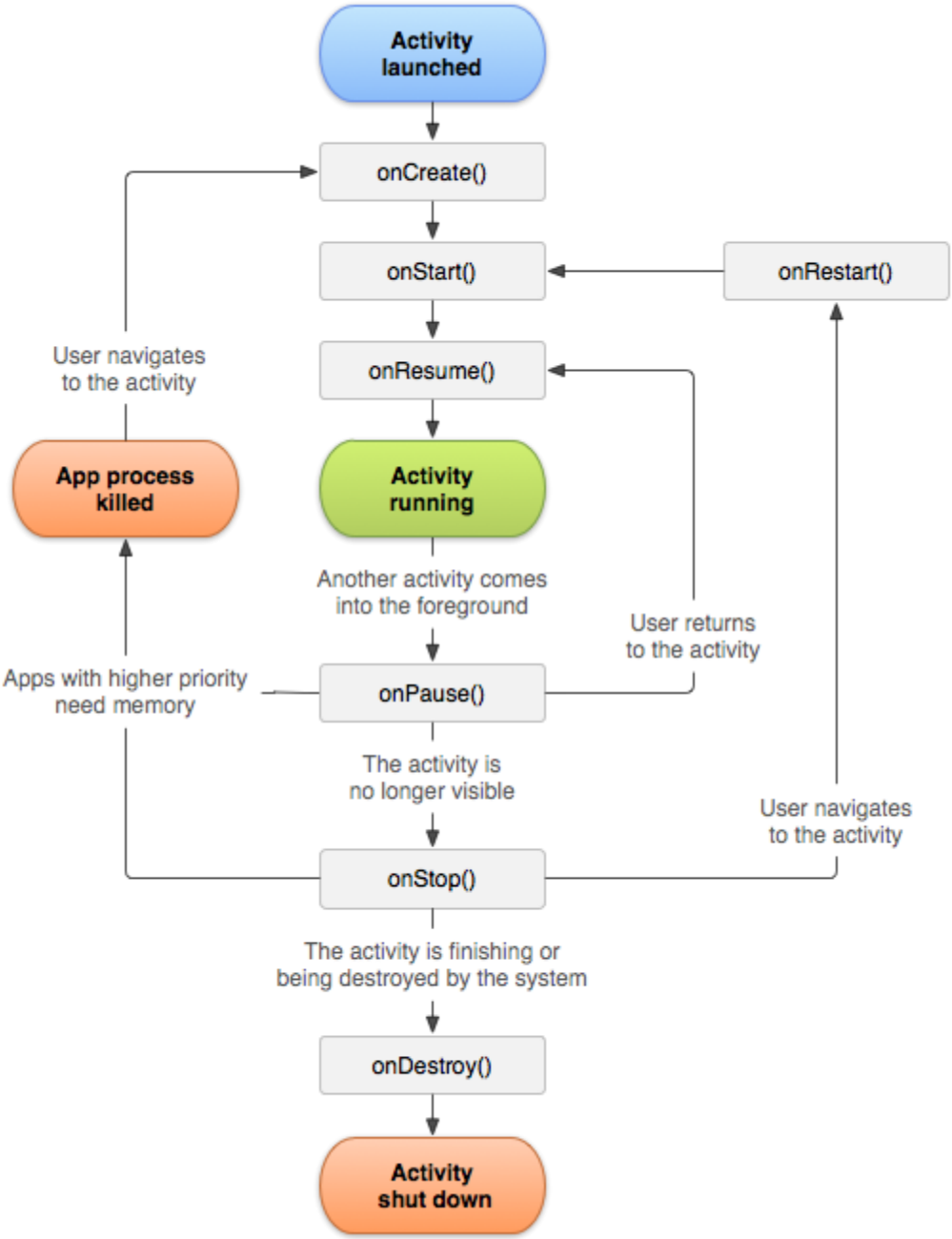
DVM used JIT (Just in time) compilation which worked by taking application code, analyzing it and actively translating it during runtime. ART uses AOT (ahead of time) compilation that compiles *.dex files (Dalvik Bytecode) before they are even needed. Usually, it is done during installation and stored in phone storage.

To be noted: After Android N, JIT compilation was reintroduced along with AOT and an interpreter in ARt making it hybrid to tackle against problems like installation time and memory.

Android OS? [5]

Q. 4 Draw Flow chart of Activity Life Cycle and describe all functions of Activity Life Cycle.

*____*____*



**Activity Created: onCreate(Bundle savedInstanceState):**

onCreate() method is called when activity gets memory in the OS. **To use create state we need to override onCreate(Bundle savedInstanceState) method.** Now there will be question in mind what is Bundle here, so Bundle is a data repository object that can store any kind of primitive data and this object will be null until some data isn't saved in that.

- When an Activity first call or launched then onCreate(Bundle savedInstanceState) method is responsible to create the activity.
- When ever orientation(i.e. from horizontal to vertical or vertical to horizontal) of activity gets changed or when an Activity gets forcefully terminated by any Operating System then savedInstanceState i.e. object of Bundle Class will save the state of an Activity.
- It is best place to put initialization code.

**Activity Started: onStart():**

onStart() method is called just after it's creation. In other case Activity can also be started by calling restart method i.e after activity stop. So this means onStart() gets called by Android OS when user switch between applications. For example, if a user was using Application A and then a notification comes and user clicked on notification and moved to Application B, in this case Application A will be paused. And again if a user again click on app icon of Application A then Application A which was stopped will again gets started.

**Activity Resumed:.onResume():**

Activity resumed is that situation when it is actually visible to user means the data displayed in the activity is visible to user. In lifecycle it always gets called after activity start and in most use case after activity paused (**onPause**).

**Activity Paused: onPause():**

Activity is called paused when it's content is not visible to user, in most case onPause() method called by Android OS when user press Home button (Center Button on Device) to make hide.

Activity also gets paused before stop called in case user press the back navigation button. The activity will go in paused state for these reasons also if a notification or some other dialog is overlaying any part (top or bottom) of the activity (screen). Similarly, if the other screen or dialog is transparent then user can see the screen but cannot interact with it. For example, if a call or notification comes in, the user will get the opportunity to take the call or ignore it.

**Activity Stopped: onStop():**

Activity is called stopped when it's not visible to user. Any activity gets stopped in case some other activity takes place of it. For example, if a user was on screen 1 and click on some button and moves to screen 2. In this case Activity displaying content for screen 1 will be stopped.

Every activity gets stopped before destroy in case of when user press back navigation button. So Activity will be in stopped state when hidden or replaced by other activities that have been launched or switched by user. In this case application will not present anything useful to the user directly as it's going to stop.

**Activity Restarted: onRestart():**

Activity is called in restart state after stop state. So activity's onRestart() function gets called when user comes on screen or resume the activity which was stopped. In other words, when Operating System starts the activity for the first time onRestart() never gets called. It gets called only in case when activity is resumes after stopped state.

**Activity Destroyed: onDestroy():**

Any activity is known as in destroyed state when it's not in background. There can different cases at what time activity get destroyed.

First is if user pressed the back navigation button then activity will be destroyed after completing the lifecycle of pause and stop.

In case if user press the home button and app moves to background. User is not using it no more and it's being shown in recent apps list. So in this case if system required resources need to use somewhere else then OS can destroy the Activity.

After the Activity is destroyed if user again click the app icon, in this case activity will be recreated and follow the same lifecycle again. Another use case is with Splash Screens if there is call to finish() method from onCreate() of an activity then OS can directly call onDestroy() with calling onPause() and onStop().