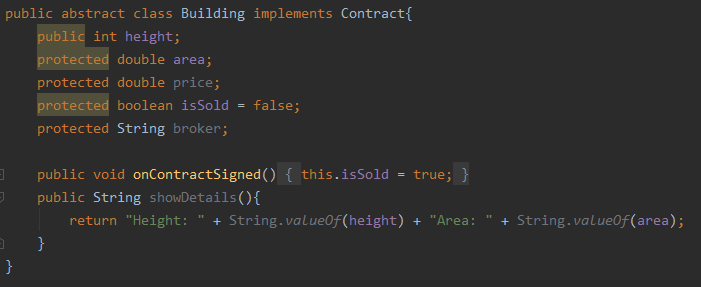
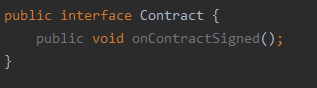
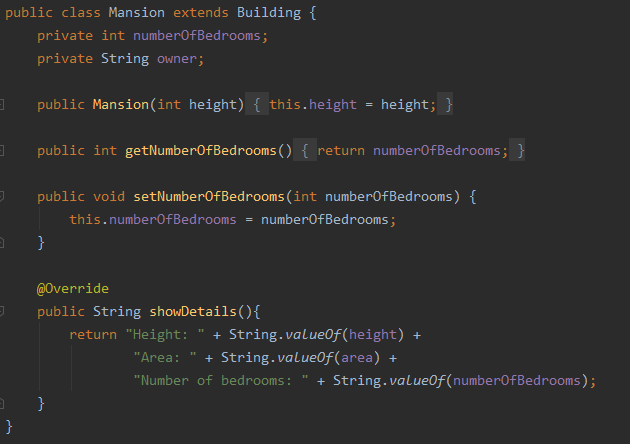
**Ex 1: Object-orient Development Main Concepts**

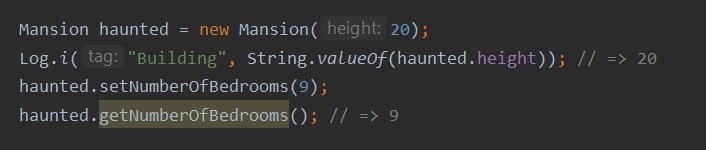
1. **Object**: is a data type whose instance contains one or several key-value pairs called data members. These pairs are predefined by creators of the programming language, but can also be defined by developers
2. **Class**: can be understood as templates to create objects. Objects created directly from the same class have share the same data members
3. **Instantiation of object** (creating an object): to create a representative of a class, which has all the data members defined in that class
4. **Visibility** (public / private / protected): these are called access modifiers, which precede the declaration of a data member and control the access to its value.
   1. “public” means that the value of this data member can be accessed anywhere (for example, create an object of this class in some random function and call to this data member of the object using the “.” Operator).
   2. “private” means the property/method can only be accessed/called within the class they are declared in.
   3. “protected” means the property/method can only be accessed/called within the class they are declared in plus all the children classes.
5. **Member datas / methods:** are pairs of key-value inside an object or a class. The values can be of any data type if it is a property or a block of code if it is a method
6. **Inheritance:** a mechanism in which an object or a class possesses all data members of other objects or classes
7. **Interface:** an abstract class whose methods can only be implemented by its subclasses.
8. **Polymorphism:** a phenomenon in which a class can be a superclass of other classes but can also be a subclass of other classes
9. **Overriding:** inherited methods is modified in the subclass. It still has the same name but might behave in a different way than its super.
10. **Abstract classes:** classes which cannot be instantiated.

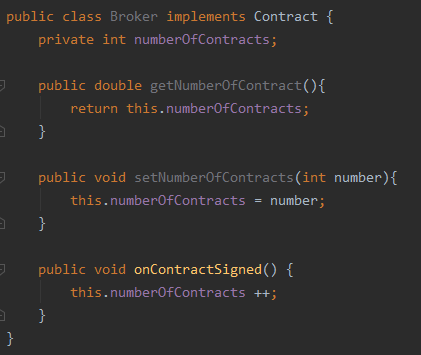
**Code examples:**

****

****





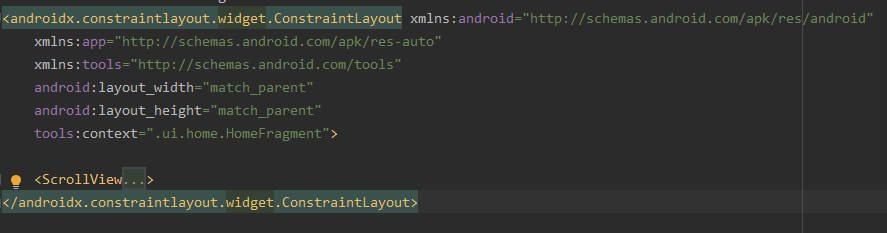


*Building* is an abstract class which holds some properties that all buildings have in common such as *height* and *area*. These are the data members of class *Building*. *Mansion* class inherits from this class, so it has all the protected properties from *Building* such as *height* and *area* and also have one private *property* of its own: *numberOfBedrooms*. Object *haunted* is created from class *Mansion. height* is a public member so its value can be accessed by *haunted*.*height*; while *numberOfBedrooms* is a private member, so its value can only be changed/read using public getter and setter methods. *Contract* is an interface class providing a method which is implemented in both *Mansion* class and *Broker* class but in different ways. Public method *showDetails()* of class *Building* is overridden in *Mansion* class.

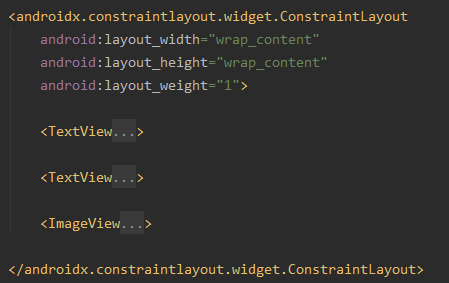
**Ex2:**

Examples of different types of layouts:

* Linear Layout: can be vertical or horizontal, set by android:orientation = “layout orientation here”



* Constraint layout:



* CoordinatorLayout: is basically FrameLayout in which interactions among components are also defined. In FrameLayout, each component is considered as a block of the screen. CoordinatorLayout “empowers” this by adding Behaviors, which are usually visual effects triggered when changes occur on the screen.

**Ex3:**

* Android apps can be written in Java, Kotlin or C++
* apk stands for Android package, which contains all the content of the app and is used by Android devices to install the app.
* Android OS encapsulate apps in separate virtual machines which have the following features:
  + Each app is considered a unique user inside Android OS and is assigned with unique ID. The OS sets permissions for certain IDs, so only apps with specific IDs can access files, and only to certain files which have been granted access to.
  + Code of an app is run only inside that VM
  + Every app runs in its own Linux process, which is started when any components of the app are executed and stopped when the process is no longer needed or the OS needs to make room in memory for other apps.
* 4 types of Android components
  + Activity is an entry point which has a single screen with UI, which facilitates the following interactions between system and app:
    - Keep track of required running processes to keep the activity run smoothly
    - Prioritize recently stopped processes which contain things user might return to
    - Help the app saving its state before the process is killed to user can return to the activity without losing information
    - Provide ways for apps to implement flows between each other
  + Service is a general-purpose entry point for keeping the app running in background to perform long-running operations or to perform work without disturbing user’s interactions with other activities. Service doesn’t provide an UI. There are two types of services: started service and bound service. Started service tells the system to keep it running until the work is done. There are 2 types of started service: one whose operation is known by user (such as playing music) and one whose operation is not (such as syncing data). Bound service is the one which runs because other app or the system depends on it.
  + Broadcast receiver is a component which receive and allow the app to respond to system-wide announcements. This is an entry point to the app, which allows system to deliver events to apps which is not even running. Broadcast receiver doesn’t display UI, but may create status bar notification to alert user about the event.
  + Content provider manages a shared set of data which is kept in persistent storage such as file system, SQLite database or on the web. Through content provider, an app can query or modify data if allowed by content provider. Content provider is best understood as an entry point between system and app for system to publish data via URI scheme and among apps to share those URIs (or the data produced by the app themselves).
* Manifest file is a file in the root directory of an app project, which declares all the components of that app. It also declares the following:
  + User permissions which the app needs
  + Minimum API level required by the app
  + Hardware and software features required by the app
  + API libraries which the app needs to be link against
* Resources are anything relating to visual presentation of the app such as images, screen layout in .xml files. They help enriching characteristics of the app without requiring changing the source code. Resources provided in different configuration helps improving UX in devices of multiple hardware specifications.

UI Hierarchies

The .xml file just declares the components in this activity, which are a horizontal linear layout containing 3 buttons, an EditText component to receive user input and a Listview to display a list of country names. These 3 major components are put inside a vertical LinearLayout container.

Inside class MainActivity, the inherited method onCreate was overridden. First, the previously saved state of this activity was passed as an argument. Next, setContentView() open the .xml file named “activity\_main”, which contains the layout of this activity. A list of strings named COUNTRIES was initiated and set to be “final”, which means its values can no longer be changed. The ListView component of this activity is assigned to a variable named myListView. Its content was set from an array adapter, which converts a list of country names mentioned above into a list of components, which are compatible to be nested inside the ListView.