1) lazy can only used for val, while lateinit can only be used for var
   Multiple initialization is possible for lateinit, but not for lazy
   lazy is allowed on properties of primitive types, but not lateinit
   lazy is thread-safe, but not lateinit
   lateinit can be initialized anywhere the object is seen from, while lazy can only be initialized by the initializer lambda
2) Extension function
   Lambda expression + inline function
   Null safety
   Data class
   String template
   Companion Object
3) Checked: are the exceptions that are checked at compile time.
   Unchecked are the exceptions that are not checked at compiled time.
   All exception in Kotlin are unchecked.
4) The Model represents a set of classes that contain the data. It also defines how the data can be changed and manipulated.

   View is a component which is directly interacts with user like XML, Activity, fragments. It does not contain any logic implemented.

   The Presenter receives the input from users via View, then process the user's data with the help of Model and passing the results back to the View. Presenter communicates with view through interface. Interface is defined in presenter class, to which it pass the required data. Activity/fragment or any other view component implement this interface and renders the data in a way they want.

   In the MVP design pattern, the presenter manipulates the model and also updates the view. In MVP View and Presenter are completely decoupled from each other and communicate to each other by an interface.
5) ViewModel is located between the View and Model layers. This is where the controls for interacting with View are housed, while binding is used to connect the UI elements in View to the controls in ViewModel.