1. What is Encapsulation? Try to list out the benefits of Encapsulation.

封裝（英語：Encapsulation）是指，一種將抽象性函式介面的實作細節部份包裝、隱藏起來的方法。同時，它也是一種防止外界呼叫端，去存取物件內部實作細節的手段，這個手段是由程式語言本身來提供的。封裝被視為是物件導向的四項原則之一。

適當的封裝，可以將物件使用介面的程式實作部份隱藏起來，不讓使用者看到，同時確保使用者無法任意更改物件內部的重要資料，若想接觸資料只能通過公開接入方法（Publicly accessible methods）的方式（ 如："getters" 和"setters"）。它可以讓程式碼更容易理解與維護，也加強了程式碼的安全性。

2. Try to explain what enum is and why we use them.

Enum 是在 TypeScript 中增加的新語法，也被稱做「列舉」或「枚舉」，實務面會用它來管理多個同系列的常數（不可修改的變數），做為狀態的判斷所使用。

列舉型別是Java 5中新增特性的一部分，它是一種特殊的資料型別，之所以特殊是因為它既是一種類(class)型別卻又比類型別多了些特殊的約束，但是這些約束的存在也造就了列舉型別的簡潔性、安全性以及便捷性

3. How to use enum in Kotlin? Attach a sample code that uses enum in Kotlin.

enum requestStatusCodes {error = 400,success = 200,}

enum requestWrongCodes {

missingParameter = 'A',

wrongParameterType = 'B',

invalidToken = 'C',

}

const handleResponseStatus = (status: number): void => {

switch (status) {

case requestStatusCodes.success:

// Do something...

break;

case requestStatusCodes.error:

// Do something...

break;

default:

throw (new Error('No have status code!'));

}

};

const handleWrongStatus = (status: string): void => {

const { missingParameter, wrongParameterType, invalidToken, } = requestWrongCodes;

switch (status) {

case missingParameter:

// Do something...

break;

case wrongParameterType:

// Do something...

break;

case invalidToken:

// Do something...

break;

default:

throw (new Error('No have wrong code!'));

}

};

4. What are the differences between LinearLayout and RelativeLayout? Try to explain in detail.

兩個佈局方式的相同，都是繼承自View.Group，是一種容器控件，LinearLayout的佈局層次比較簡單，方向和位置都比較容易控制，而相對佈局RelativeLayout 對於其內子控件提供了更加豐富的佈局層次和佈局位置。

5. Try to explain the benefits of ConstraintLayout. Why should you use it?

所有的佈局可以在界面上通過拖動和調整來完成，這點相對於 RelativeLayout 要方便很多。

不需要嵌套 linearLayout，直接添加控件，將控件之間的約束調整好使用。