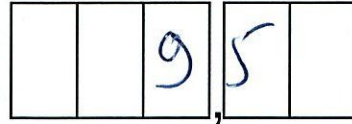


NOM Poupa

Prénom Adrien

Promo D1 2018

Date 12/04/17



POUPA Adrien  
M1 - 2016

## MATIÈRE Mobile Development

### 1) Code Analysis

1. The following code is an activity that contains a button that creates a notification when it is clicked.

The notification has a title, a description, sends an intent and makes the phone vibrate.

2. We will use an intent.

```
2) 3. <Linear Layout xmlns:android="http://..."
    android:layout_height="match_parent"
    android:orientation="horizontal"
    android:gravity="center"
    android:layout_width="match_parent" >
    {
    <Button android:text="1"
        android:id="@+id/button01"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
    </Button >
    }
</Linear Layout >
```

duplicate  
for buttons  
2 & 3  
changing  
text and id

U. By default, the buttons should be disabled because we don't know yet if the service is available.

S. U: String containing the button ID to load (eventually the URL which could contain the button ID)

V. Integer, stage of completion of the task

W. String, data returned from the ~~server~~ server

G. protected void onPostExecute (String string) {  
// I can't remember exactly how to extract a char from a string using its position

String firstButtonString = string.substr(0,1);

String secondButtonString = string.substr(2,3);

String thirdButtonString = string.substr(4,5);

if (firstButtonString == 0) {

myFirstButton.set Enabled (false);

}

else {

myFirstButton.set Enabled (true);

}

// do the same check for mySecondButton and myThirdButton

}

(button disabled)

7. This should be done in the function onPreExecute such as myButton.set Enabled (false); and the web service should be checked in the doInBackground() function.

### 3) List

```
8. <LinearLayout xmlns:android="http://..."  
    android:orientation="vertical"  
    android:layout_width="match_parent"  
    android:layout_height="match_parent"  
    android:gravity="center"
```

>

```
<TextView
```

```
    android:id="@+id/name"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:text="@string/name"  
    android:layout_gravity="center"
```

>

```
<TextView android:id="@+id/episode"  
    android:text="My Episode" />
```

```
<TextView android:id="@+id/season"  
    android:text="my season" />
```

```
</LinearLayout>
```

9.

10. In the activity class:

```
ListView  
Button b = (Button) findViewById(R.id.myButton);  
b.setOnClickListener(new OnClickListener() {  
    public void onClick (View v) {
```

listViewID

```
        Intent broadcast = new Intent("my.episode");  
        //broadcast.putExtra(position); //position of the current item  
        sendBroadcast(broadcast);
```

```
    }  
};
```

NOM POUA

Prénom Adrien

Promo 71 2018

Date 12/04/17

7/1/16

## MATIÈRE Mobile development

### 1) Simple code analysis

1) UINavigationController class is a parent class in charge of implementing controllers used in views such as the Split view, the Tab Bar or Navigation Controller. These are UINavigationController properties <sup>it is currently in /</sup> Tab Bar Controller, SplitView Controller and Navigation Controller.

In general, in iOS, one should use multiple MVCs, one for each functionality. We can use a segue to communicate between MVCs.

2) UIKit is a framework used to create user interfaces on iOS. ✓

3) UILabel! type is UILabel or exception.

In general, optionals are just ~~enum~~ enums in Swift. The question mark is used as a "else" statement so that `String? = nil` means that it should be null if there is no String. ✓

To simplify the code even more, Optionals can be chained.

4) @IBAction is used in front of a functions triggered by the user.

@IBOutlet is used at the end of an instruction

Interaction between UI and Controller is done through `UIViewController`.

5) `displayValue` is a ~~mutable~~ <sup>Computed</sup> property.

Read only:

```
var displayValue: Double {  
    get {  
        return Double(displayLabel.text!)  
    }  
}
```

II) Note Swift and UIKit

1. a) A dictionary is an associative array in Swift. It is declared like this:

```
var dict = [String: Int]()
```

Assigned like this

```
dict = [ "MyString": 1, "My other string": 2 ]
```

Accessed like this:

```
let first = dict["MyString"]
```

Enumerated like this


```
for (key, value) in map dict {  
    print (" \ (key) = \ (value) ")  
}
```

b) var dict = [Int: String]()

```
dict = [ 1 : "Monday",  
        2 : "Tuesday",  
        3 : "Wednesday",  
        4 : "Thursday",  
        5 : "Friday",  
        6 : "Saturday",  
        7 : "Sunday" ]
```

```
c) for (key, value) in dict {  
    print (" \ (key) = \ (value) ")  
}
```

2) a) To create a custom UIView, we create a UIView subclass and we override drawRect:

Override func drawRect(region to be drawn: CGRect) 

using a C-like (non OO) API (Core Graphics) or using the UIBezierPath.

b)

