# Task ##P/C – Spike: Renting app

## Goals:

*This goal of the assignment is to create a simple renting app. This app need to include the implement of opening activity 2 and passing back the data from the activity 2 using intent and parcelable object. Also, this app is a great app to write a simple ui and unit test. Moreover, the app UI also being develop using android widget and validate the user input from the second activity*

*List if knowledge gap:*

* Pass back data
* Validation user input
* Toast
* Unit and UI test
* UI element
* Style in app
* Sketch
* Intent and Parcelable object
* Command of IDE

This is the link to my github classroom:

<https://github.com/SoftDevMobDev-2023-Classrooms/core2-bbi3mn4u69/tree/main>

## Tools and Resources Used

*This section lists related software, tools, libraries, API's, and other resources used for this knowledge gap.*

* Android studio
* Thakkar, R 2021, *UI Testing with Espresso in Android*, Mindful Engineering.
* *RatingBar* n.d., Android Developers, viewed 30 September 2023, <https://developer.android.com/reference/android/widget/RatingBar>.
* SeekBar n.d., Android Developers, viewed 30 September 2023, <https://developer.android.com/reference/android/widget/SeekBar?hl=en>.

‌

## Knowledge Gaps and Solutions

*This section presents the listed knowledge gaps and their solutions with supporting images, screenshots and captions where appropriate/required.*

### Gap 1: Pass Back Data

Since we are able to create a second activity, pussing data in and passing data back is important. In order to get the data back, I m using ActivityResultContract first we need to define a late initialization variable “result laucher”. This variable will hold an object of the ActivityResultLauncher type, which is a part of Android's Activity Result API. The <Intent> part indicates that this ActivityResultLauncher will be used for launching intents



Figure 1 result launcher

The result laucher will first used to lauch the second activity with passing the parcelable object (game) as an intent. The object here will be initialize at the beginning and being pushed in a list of game object (Information list).

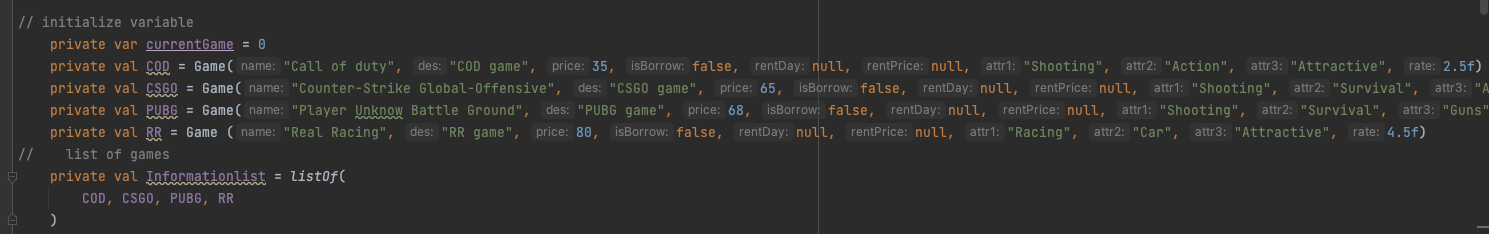


Figure 2 list of object

Here we are pushing the current game of that list aka the object at index number of that list of game. Pushing the current game (object) that the screen is showing and launching the second activity :



Figure 3 lauch the second activity

Second activity pass back the data:



Figure 4 pass back the data from the second activity

Initialize the result laucher for result from the second activity

After the make sure the result ok, the second activity will pass the days, price and isborrow back to the main activity. These variable now will replace the object rentday, price and isborrow for further display to the screen. A toast will show if the user confirm that they want to borrow the game later on.

A screen shot of a computer

Description automatically generated

Figure 5 result launcher

### Gap 2: Validation of User input

For validation the user input, I have set up the code: the save button can only pressed if the user make change on the seek bar, if the back button is pressed, that consider to be the user cancel it:

A screenshot of a computer program

Description automatically generated

Figure 6 validation of user input

### Gap 3: Toast

I have implement the toast in this code, the toast will show up when the user press the save button without making any change to the seek bar. Also, when the user is make change to the seek bar to set the days they borrow the game and press the save button to go back, another toast will appear saying “good choice”. If they press the back button, the program will not save the borrow information and display a toast.

A screen shot of a computer program

Description automatically generated

Figure 7 Toast

In this app, I opted for using toast messages instead of snackbars because the messages displayed here don't affect or require any action from the user in response. Additionally, snackbars persist until the user takes an action or a timeout is set programmatically, which can result in an inconvenient user experience and unnecessary, redundant code. In contrast, toast messages simply appear for a brief 2-3 seconds, making them a more efficient choice.

### Gap 4: Unit and UI test

I have write a UI test for the program, tested if the text and the button display on the activity correctly: for example the save button will be appear if we press the borrow button and go to the second activity, etc

A screen shot of a computer code

Description automatically generated

Figure 8 UI Test

Also, a unit test is being implement. This unit test is make sure that the individual component behaviour in the way that we expected

For example: this test is when press the borrow button and set the values for the seek bar, then press save, the app return to the main activity and the borrow button is expected to be disable

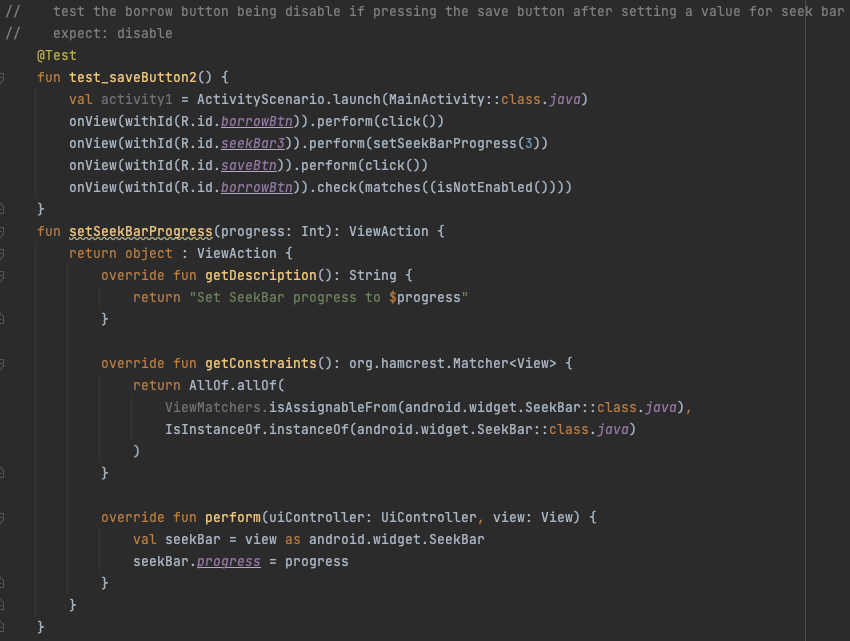


Figure 9 Unit Test

All the test is passed:

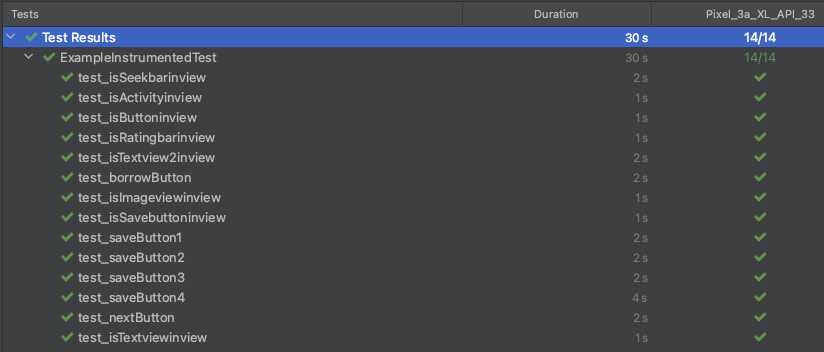
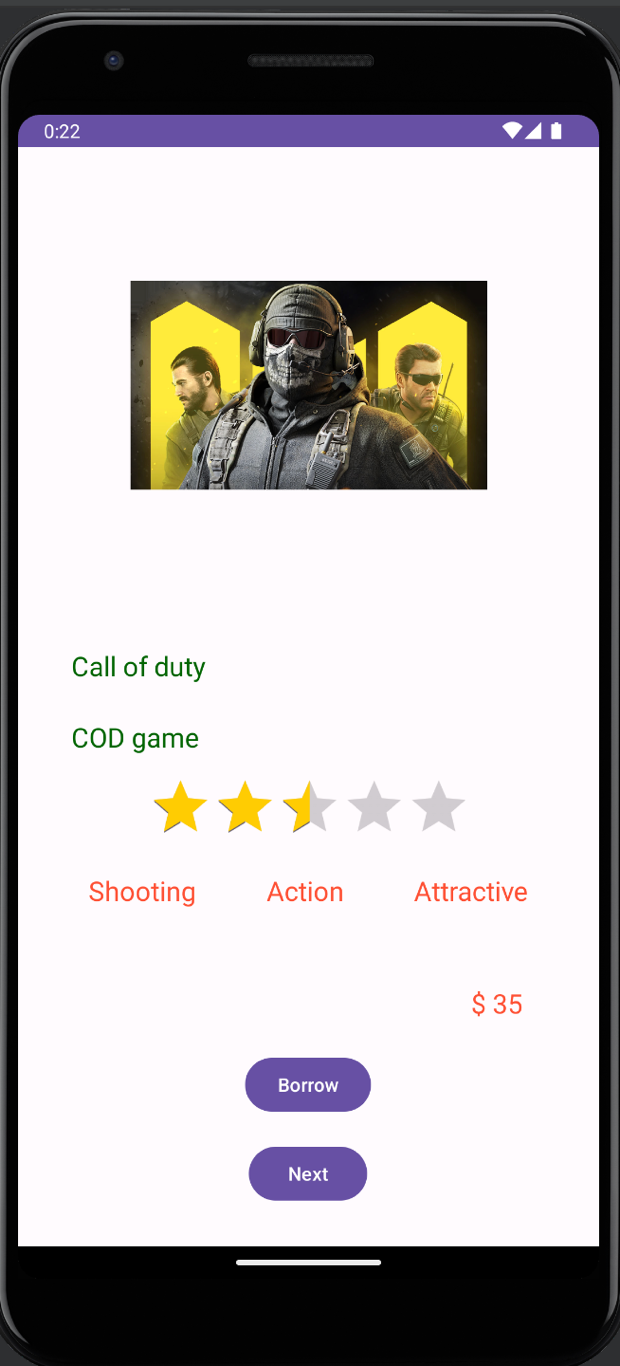


Figure 10 Test passed

### Gap 5: UI element

In this app, I have use a wide range of UI element, most typical is the seek bar and the rating bar the rating bar is used for the user to see the rating of the games that they will choose to borrow. Meanwhile the seek bar is used to set the days that the user borrow the game.

A screen shot of a phone

Description automatically generated

### Gap 6: Work with resource

The image resource is included in a list of drawable resources. The "current game" variable corresponds to the index number of a list of games. The resource is accessed and displayed based on the value of the "current game" variable. For instance, if the current game is set to 0, the first resource (representing "COD")

will be drawn and displayed.

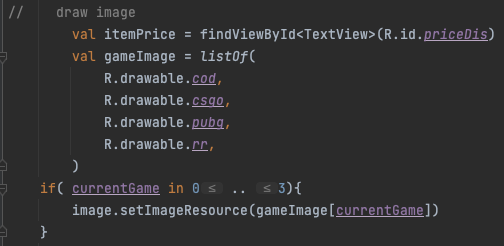


Figure 11 handle image resource

In this approach, the code will be more clean and less redundant code rather than using when.

### Gap 7: Style in app

For this app, im using a wide range of style for different component. Including the attribute style, the text view style, the rating bar

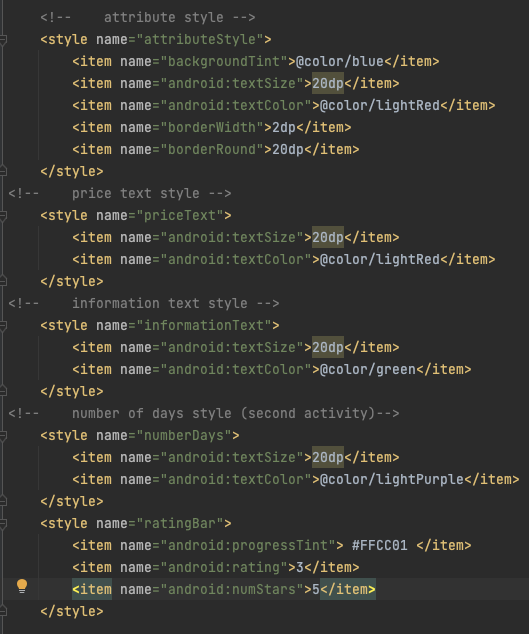


Figure 12 style in app

### Gap 8: Sketch

For this app, I have sketch it on iPad, the interface will look like this:

First possible layout for the main activity and the second activity:

A screen shot of a black screen

Description automatically generated

Figure 13 initial desgin

A screen shot of a cellphone

Description automatically generated

Figure 14 initial desgin

The second possible layout

A screen shot of a phone

Description automatically generated

Figure 15 second possible layout

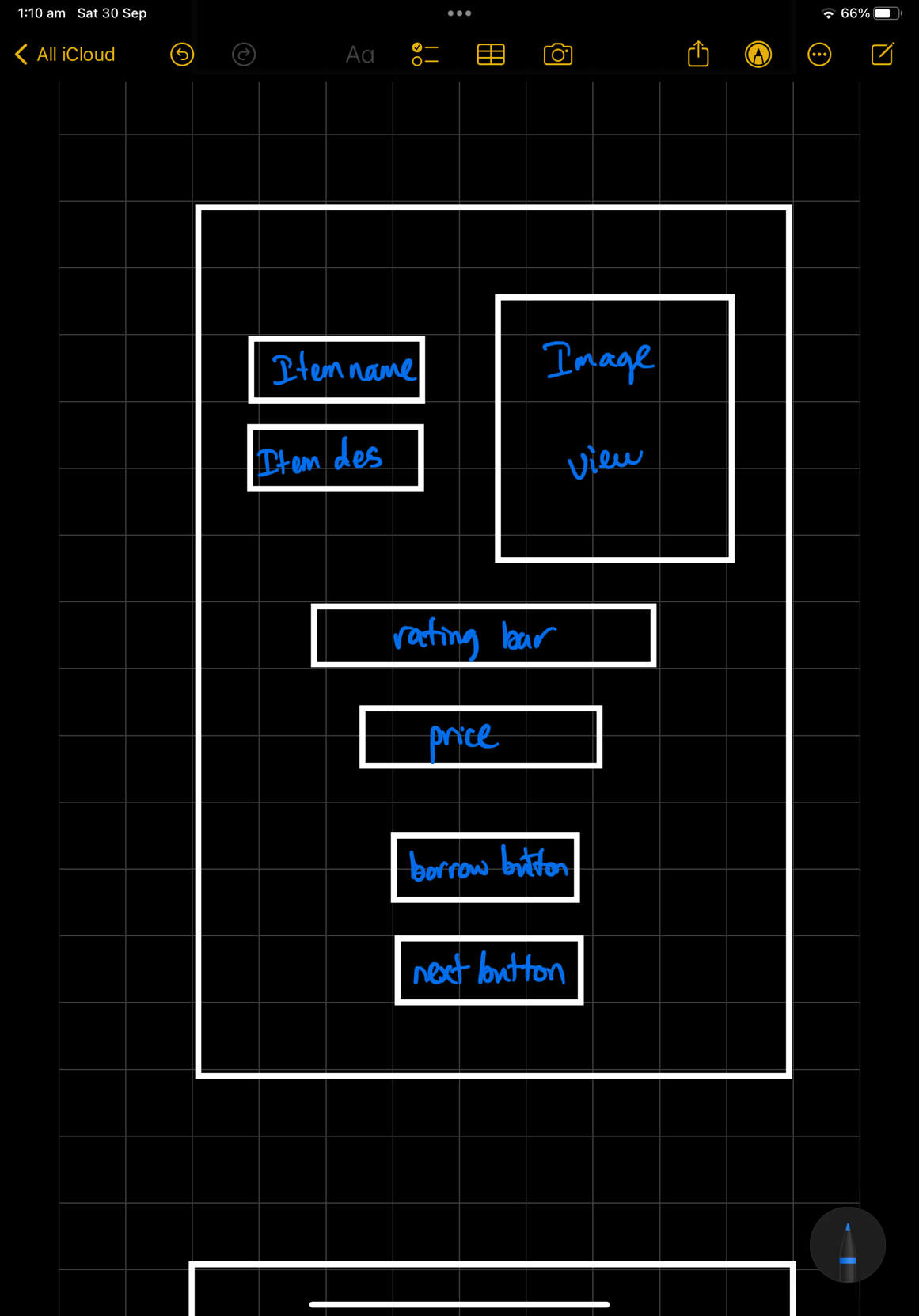


Figure 16 second possible layout

Considering both layout, the first layout seem to provide to the user more information than the sencod possible layout. Moreover, with a bigger space for image, that layout will fit both square or rectangle image while not being crop of. On the other hand, the space for image in layout two a little bit small, if it a rectangle image, the detail on the image will be small, provide a inconvenient for the user experience. After all, the first layout is the best choice for the app.

### Gap 9: Intent and parceable object

In android development, intent is a fundamental component of the android operating system. Intent is also a mechanisim for different action happen on the program. Its can be used to pass the data and also lauch an activity:

For example in the code below, the intent is being created in the current activity (activity 1) and the target component is the activity2

The intent here will being taking with the Extra data. In this extra data, game is the key name of the data, which used to identify the data, Informationlist[currentGame]: This is the value associated with the key. It is an element from a list called Informationlist at the index current Game



Figure 17 intent and parcelable object

Im also using a parcelable object. An object that implements the android.os.Parcelable interface is known as a Parcelable object. It enables efficient serialisation and deserialization of custom objects for use in passing them across various Android components and processes.

As here, the game data class will be implemented to be used as parcelable, so all the object create based on this data class will be consider as parcelable object.

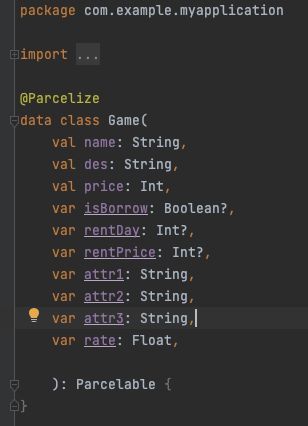


Figure 18 data class

The advantage of Parcelable object: When transferring data through intents on Android, using Parcelable objects has many benefits. By contrast to conventional serialisation techniques, Parcelable objects are designed for Android's inter-component communication, enhancing performance and efficiency in the first place. Complex data structures may be efficiently transferred between tasks, services, or even between separate apps. Additionally, Parcelable objects are memory management optimised for Android, which is important for mobile devices with limited resources. Additionally, because the Parcelable mechanism is designed specifically for Android, it can be seamlessly integrated with system functions and services, making it a crucial option for effective data transfer and communication in Android app development.

### Gap 10: Command of IDE

For is app to build successfully, we need to add more plugin dependencies

To be more specific, kotlin-parcelize is being added, this is a Kotlin plugin specifically for enabling the @Parcelize annotation. The @Parcelize annotation is used to generate Parcelable implementations for Kotlin classes automatically

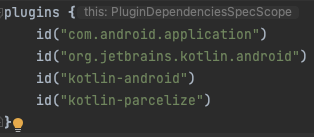


Figure 19 dependencies plugin

Moreover, this task require to write the test so this is the path to the test file

A screenshot of a computer

Description automatically generated

Figure 20 file path

Inside the test file:

The @RunWith annotation in Android is used to specify the test runner class that should be used when running Android instrumentation tests. In code, @RunWith(AndroidJUnit4::class) is specifying the test runner class as AndroidJUnit4, which is a commonly used test runner for Android instrumented tests.

A black background with white text

Description automatically generated

Figure 21 annotation

The @Test annotation used to mark a method as a test method. This annotation indicates that the method contains test logic that should be executed when running tests.

A screenshot of a computer program

Description automatically generated

Figure 22 annotation

We can see the test result here:

A screenshot of a computer

Description automatically generated

Figure 23 result test

To create another activity, we will follow this path:

A screenshot of a computer

Description automatically generated

To create the new data class, we will follow this path:

A screenshot of a computer

Description automatically generated

Also choose the data class and rename it:

A screenshot of a computer

Description automatically generated

Also to set up the style, first I will set up the color in this file path:



After that set up the color:

A screenshot of a computer code

Description automatically generated

And then used it in a file style follow this path:



And setup the style:

A screen shot of a computer program

Description automatically generated

## Open Issues and Recommendations

*The only open issue it that I can not passing the data from second acitivity back as a form of parcelable object, I have tried but its keep getting error, so instead, I choose to pass like normal data though intent. This will create redundant code.*

*In the future, I will address this problem.*