5COSC023W - Tutorial 11 Exercises

As part of the tutorial for this week, you should complete **ALL** the tasks described in the following specifications: (**make sure that you ask questions to your tutor for anything that you do not understand or if you are stuck at any point**).

> Tutorial sessions are practical sessions that you need to work towards the exercises set. They will give you the chance to practice the material learned in the lectures and learn new things as well.
>
> **You should not use these sessions to work towards the assessed assignments!** If you decide to work towards your assessed work instead, then you are not considered as part of the tutorial session. You will not get any help on the code of the assessed work by your tutor but you can ask your tutor ONLY about any clarifications you might need regarding the specification of the coursework.

Like all other modules, you are expected to study towards you module outside the lecture and dedicated tutorial slots for a number of hours. If you do not finish all of the exercises in the tutorial session, make sure that you finish them on your own time and by the end of the week. This is a normal process and part of your university learning.

For all the tasks you should use Jetpack Compose and NOT Views!

The Cocktails App - Another Web Services Example

Develop an Android application which the user interacts with to find and prepare coctails. The application will using Kotlin and the `HttpURLConnection` based on what we covered in the lecture. If you have not implemented the Book Finder app that wass covered in the lecture, you should implement that first and make sure that you understand the full code.

The application will be using the `https://www.thecocktaildb.com/api.php` Web service.

1. Implement the application so that the user can search for the names of all coctails containing an ingredient, e.g. Vodka, by using the following url: `https://www.thecocktaildb.com/api/json/v1/1/filter.php?i=Vodka` The user should be see all the names of the coctails containg vodka.

2. Extend the application so that the user can enter the name of a coctail and receive the instructions for the preparation of the coctail. This should be done by displaying the content of the field strInstructions after a request to the `https://www.thecocktaildb.com/api/json/v1/1/search.php?s=margarita` url.

3. Extend the application so that the it displays a picture for the coctail (hint: this is the url corresponding to the strDrinkThumb key of the JSON response in the last subquestion).

4. Extend your application so that user presses a button and he/she is presented with the full details of a random coctail (including the name, the recipe and its picture).

   Hints: you need to study the API of the website to make the relevant requests.

   You can use the following function for retrieving the bitmap picture from a URL and then use a composable `Image` to display the `ImageBitmap` returned from the function:

```
// retrieve a bitmap image from the URL
fun getBitmapPicture(): ImageBitmap {
    var bitmap: Bitmap? = null

    val url = URL(picture_url)
    val con = url.openConnection() as HttpURLConnection
    val bfstream = BufferedInputStream(con.inputStream)

    bitmap = BitmapFactory.decodeStream(bfstream)

    return bitmap.asImageBitmap()
}
```

The Memory Game (Homework)

If you do not have the time, to start/finish this task during your tutorial session, work towards it at your free time as a homework. Sample solution code is provided but do not look at it, until you spend a number of hours for this task.

You will be developing a game that help users to improve their memory. You might want to follow the steps indicated in the pseudocode shown at the end of this tutorial or simply follow your own steps.

The game is based on a grid. A grid of empty squares is presented to the user for 5 seconds. Following this, some random cells of the grid will be highlighted as green for 5 seconds. The aim of the game is that the user memorises these green cells and be able to recall them.

1. A random empty grid is presented to the user. The size of the grid will be selected randomly and its size will be 3×3, 3×4, 4×3, 5×5, 4×5, or 5×4.

   After 3 seconds, a random number of cells (4, 5, or 6 cells or for simplicity always choose 6 cells) will be presented to the user highlighted as green. These should be displayed for 3 seconds and after this time the grid will display as empty again.

   Hint: Represent each cell with a button. To display the green cells for 3 seconds and then make them become gray again, use the `CountDownTimer` abstract class and implement its methods `onTick` and `onFinish`. Study the relevant documentation from the Android API. For example, the following code shows a 30 second countdown in a text field:

```
object : CountDownTimer(30000, 1000) {
    override fun onTick(millisUntilFinished: Long) {
        mTextField.setText("seconds remaining: " + millisUntilFinished / 1000)
    }
    override fun onFinish() {
```

```
            mTextField.setText("done!")
        }
    }.start()
```

In your case, you can implement the `onTick` method with a blank body doing nothing, as nothing happens during every clock tick.

2. Extend the app, so that the user clicks on each of the buttons that were presented to them as green earlier. Each correct guess (click) will highlight the cell as green. Each incorrect guess will highlight the cell as red with an `X` letter displayed on the cell itself.

3. Extend the app so that the user can click only for a limited number of times. The limit of how many clicks the user can do, is the same as the number of green cells presented earlier. After the maximum number of tries is attempted, the game restarts itself and it displays a new set of 4, 5 or 6 green cells (or simply choose always 6 cells) that the user needs to memorise. The whole process is repeated again.

4. Extend the app so that it keeps a score of how many correct cells were guessed by the user and it displays the score in the top right corner of the activity. For example, if during the app, the user was presented in total with 27 green cells that were supposed to be memorised and the user guessed 21, then the score `21/27` should be displayed on the top right corner of the activity.

5. Extend the app so that it updates the score of the game after each click by the user. You might have already implemented this.

Pseudocode Steps

1. Decide on the number of rows and columns of the grid.

2. Choose some (6) random cells to become green.

3. Draw the grid using rows of buttons

4. Present some cells to the user with green colour - for 3 secs

5. Add a `Text` which displays the total score.