5COSC023W - Tutorial 2 Exercises

As part of this 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**):

The Lost Dog Application

1. Implement the "Lost Dog" application described in the lecture (watch the lecture video if you did not attend the live lecture). The image of the dog used can be downloaded from `https://github.com/udacity/dog-project/blob/master/images/Brittany_02625.jpg`

Displaying Drawables Dynamically

As we have seen previously when we would like to display drawable resources like images we could use the `@drawable/resource` in XML.

Similarly with other resources, drawable resources can be used to update widgets dynamically during run-time (in Kotlin code) depending on user choices.

The following, shows two different ways of setting the image content of your views dynamically. Try both of them in your "Lost Dog" implementation (choose the appropriate variable names, the code below is just an example that you should understand, not simply copy and paste it!)

1. You can delete the `android:srcCompat="@drawable/brittany_02625"` line from your XML layout file and modify the Activity's code so that the `onCreate()` method looks like the following:

```
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_main)

    // retrieve the image view
    val imgView1 = findViewById<ImageView>(R.id.iv1)

    // set the content of the image view
    imgView1.setImageResource(R.drawable.brittany_02625)
}
```

2. Alternatively, the resource id can also be retrieved programmatically, if you know the name of the actual resource, e.g. the image filename and use it without its extension. Modify the `onCreate()` method to look as the following:

```
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_main)

    // retrieve the image view
    val imgView1 = findViewById<ImageView>(R.id.iv1)

    // set the content of the image view
    val resource: String = "brittany_02625"
    val resource_id = resources.getIdentifier(
                                        resource,
                                        "drawable",
                                        "uk.ac.westminster.dogbreeds")
    imgView1.setImageResource(resource_id)
}
```

The `getIdentifier()` method accepts 3 arguments: the first is the resource name, the second the type of the resource (it is a drawable) and the third is the package name of the application.

Identify the Dog Breed App

The task is to develop an application that the user will be using to gain knowledge and be able to identify dog breeds.

1. When the application starts it displays to the user 3 different unique random different breed dog images. The images should be clickable. It is not allowed to display the same image more than once, i.e. the 3 images should be unique, as it should also be the breed.

   You can use images of different breeds from the following website: `http://vision.stanford.edu/aditya86/ImageNetDogs/`

   The screen should also display the name of a breed corresponding to one of the displayed images and a button labelled `Submit`.

   The user's aim is to click on the dog image corresponding to the displayed breed name, after which (a single attempt is only allowed) the message *CORRECT!* (in green colour) or the message *WRONG!* (in red colour) appears, depending on whether the answer given is correct or incorrect respectively.

   Following this, the user should click the `Next` button so that 3 new random images are displayed, giving the chance to play again. Every time that this option is chosen different images should be displayed.

2. Extend the application so that the first screen contains a second button with the title `Finish`. As soon as the user clicks on the button, a new activity starts. The new activity displays how many correct guesses and how many incorrect guesses the user has made since the start of the application (no saving required — as soon as the application terminates the score is reset).