College of Engineering, Construction and Living Sciences
Bachelor of Information Technology
IN721: Mobile Application Development
Level 7, Credits 15

Practical 02: Calculator

Assessment Overview

In this assessment, you will design, develop & UI test a calculator application. Also, you will research & implement a menu using the provided resource. This assessment contributes **4%** towards your final mark in **IN721: Mobile Application Development**.

Learning Outcomes

At the successful completion of this course, learners will be able to:

1. Implement & publish complete, non-trivial, industry-standard mobile applications following sound architectural & code-quality standards.
2. Identify relevant use cases for a mobile computing scenario & incorporate them into an effective user experience design.
3. Follow industry standard software engineering practice in the design of mobile applications.

Assessment Table

Assessment Activity	**Weighting**	**Learning Outcomes**	**Assessment Grading Scheme**	**Completion Requirements**
Practical	20%	2, 3	CRA	Cumulative
Project	80%	1, 2, 3	CRA	Cumulative

Conditions of Assessment

You will complete this assessment during your learner managed time, however, there will be availability during the teaching sessions to discuss the requirements & your progress of this assessment. This assessment will need to be completed by **Friday, 19 March 2021 at 5:00 PM**.

Pass Criteria

This assessment is criterion-referenced (CRA) with a cumulative pass mark of **50%** over all assessments in **IN721: Mobile Application Development**.

Authenticity

All parts of your submitted assessment must be completely your work & any references must be cited appropriately including, externally-sourced graphic elements. Provide your references in a **README.md** file. All media must be royalty free (or legally purchased) for educational use. Failure to do this will result in a mark of **zero** for this assessment.

Policy on Submissions, Extensions, Resubmissions & Resits

The school's process concerning submissions, extensions, resubmissions & resits complies with **Otago Polytechnic** policies. Learners can view policies on the **Otago Polytechnic** website located at https://www.op.ac.nz/about-us/governance-and-management/policies.

Submissions

You must submit all program files via **GitHub Classroom**. Here is the URL to the repository you will use for your submission – https://classroom.github.com/a/VJIq7Ae0. Create a new branch called **02-calculator** from the **main** branch by running the command - **git checkout -b 02-calculator**. This branch will be your development branch for this assessment. Once you have completed this assessment, create a pull request & assign the **GitHub** user **grayson-orr** to a reviewer. **Do not** merge your own pull request. Late submissions will incur a **10% penalty per day**, rolling over at **5:00 PM**.

Extensions

Familiarise yourself with the assessment due date. If you need an extension, contact the course lecturer before the due date. If you require more than a week's extension, a medical certificate or support letter from your manager may be needed.

Resubmissions

Learners may be requested to resubmit an assessment following a rework of part/s of the original assessment. Resubmissions are to be completed within a negotiable short time frame & usually must be completed within the timing of the course to which the assessment relates. Resubmissions will be available to learners who have made a genuine attempt at the first assessment opportunity & achieved a **D grade (40-49%)**. The maximum grade awarded for resubmission will be **C-**.

Resits

Resits & reassessments are not applicable in **IN721: Mobile Application Development**.

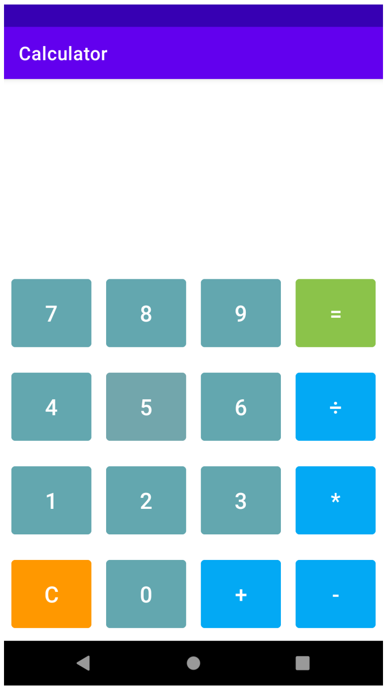
Instructions - Learning Outcomes 2, 3

Task One (0.5%):

Create a new project with the following configurations:

- Template - Empty activity
- Name - Calculator
- Package name - op.mobile.app.dev.calculator
- Save location - /path to your practical GitHub repository/02-calculator
- Language - Kotlin
- Minimum SDK - API 28: Android 9.0 (Pie)

In **activity_main.xml**, create a layout which resembles a calculator. For example:



Note: each number & operator in the example above is a **Button** widget.

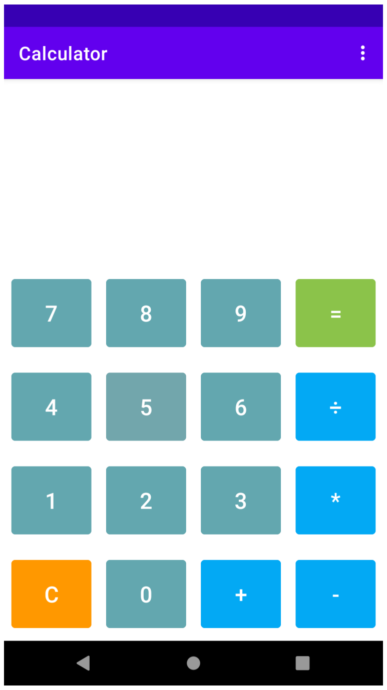
Task Two (0.5%):

The application's **action bar** displays the activity's **title** on the top left-hand side & an overflow menu on the top right-hand side. Create a new resource directory called **menu**. To do this, right-click on **res** > **Android Resource Directory**. In the **New Resource Directory** window, change the **Directory name** & **Resource type** to **menu**. In the **menu** directory, create a new **XML** file called **menu**. Use the following resource to define a menu - https://developer.android.com/guide/topics/ui/menus

To specify the options menu for an activity, i.e., **MainActivity.kt**, override the **onCreateOptionsMenu()** method. In this method, inflate your **menu** resource defined in **menu.xml** into the **Menu** provided in the callback. For example:

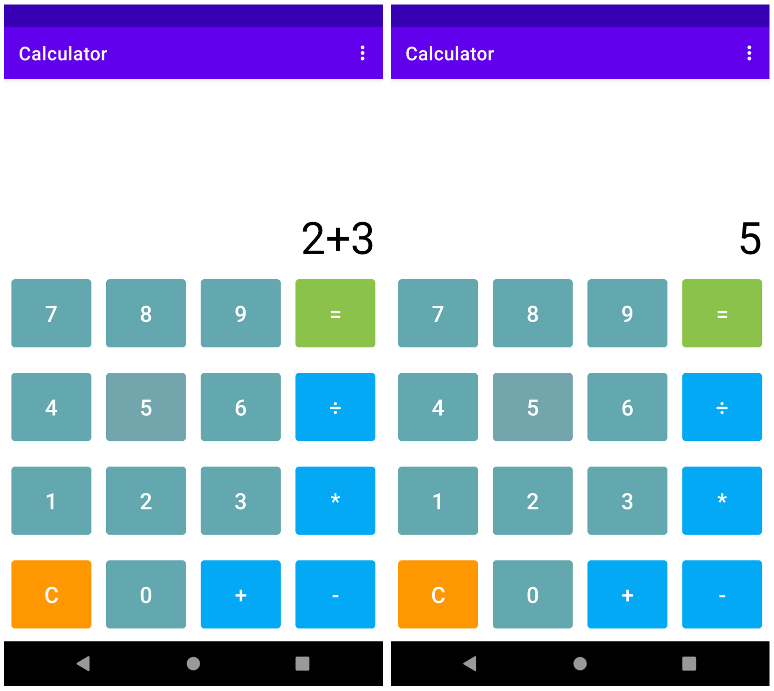
```
    override fun onCreateOptionsMenu(menu: Menu): Boolean {
        val inflater = menuInflater
        inflater.inflate(R.menu.menu, menu)
        return true
    }
```

Run your application on either an **Android Emulator** or **connect device**. You should see a vertical ellipsis on the top right-hand side. This is your inflated **menu** resource mentioned above.

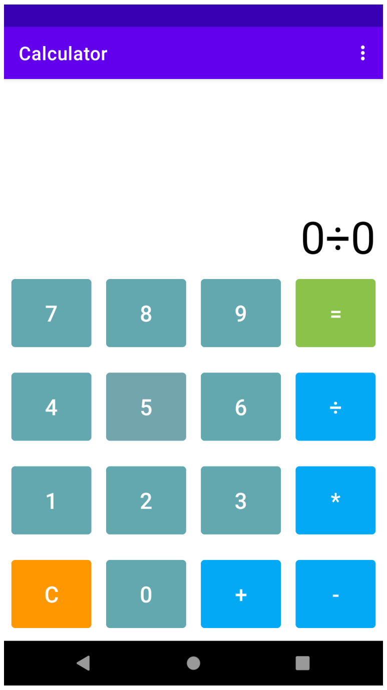


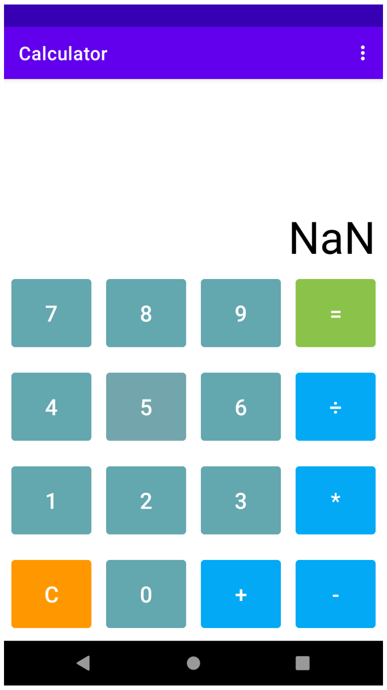
Task Three (2%):

In **MainActivity.kt**, write code that allows your application to function as a calculator. If you wish, you can use third-party libraries. However, it is encouraged that you write the functionality from scratch.



Where possible, write code that checks for potential calculation errors. For example, dividing by zero should return **NaN** or **undefined**.





Task Four (1%):

One method to UI testing is to have a person perform a set of tasks on a target application & verify that it is behaving correctly. However, this manual method is time-consuming & error-prone. A more efficient method is to write UI tests that perform such tasks in an automated way.

Create a new test file called **CalculatorTest**. To do this, right-click on **op.mobile.app.dev.calculator (androidTest)** > **Kotlin Class/File**. In **CalculatorTest.kt**, write UI tests for the following cases:

- Calculate two numbers. **Note:** any operator will be sufficient.
- Output of the calculation. For example, if you add 2 & 4, you should expect the output to be 6.
- Calculate two numbers using the divide operator so that the output is **NaN** or **undefined**.

To run your test file, right-click **CalculatorTest.kt** > **'Run CalculatorTest'**.