	University of Aberdeen **School of Natural and Computing Sciences** **Department of Computing Science** **BSc in Computing Science** **2024 – 2025**

Assessment 1 – Student Budgeting Tool

Title: CS2020 – Software Programming	Note: This assessment accounts for 35% of your total mark of the course.

Learning Outcomes

On successful completion of this component a student will have demonstrated competence in the following aspects:

- Ability to write and run basic Java applications
- Ability to judge how Java applications should be structured
- Understanding and ability to apply, techniques to support correct code in a Java application
- The ability to analyse simple problems and design a program solution
- The understanding of software libraries and their application in order to classify them for usage
- The ability to apply programming concepts for solving problems in order to create a new program
- Knowledge and understanding of basic programming concepts and their application
- Ability to create a graphical user interface using Swing

Information for Plagiarism: Your report and test cases may be submitted for plagiarism check (e.g., Turnitin). Please refer to the slides available at MyAberdeen for more information about avoiding plagiarism before you start working on the assessment. Please also read the following information provided by the university: https://www.abdn.ac.uk/sls/online-resources/avoiding-plagiarism/

If you decide to use any Generative AI to aid you in your work, then declare this in the comments. As you're still learning Java, we'd advice against this, as it will slow down your learning of the language and its constructs.

Use Git to Save Your Work

While we're not marking your use of Git in this assignment, you are strongly encouraged to do this work inside a Git repository. By doing that you gain the benefit of rolling back your edits, or doing work in branches, and then merging it to the main branch as you accomplish tasks along the way.

Perhaps most importantly, by using Git and pushing your work to a remote repository on GitHub, you provide a safe backup of your work. You'd be surprised by how regularly fellow students discover failed hard disk drives, or that the laptop they're using needs to be returned to its owner, or be repaired due to some liquid being spilt on the keyboard.

Assessment Details

In this assessment, you will build a simple budgeting tool for students, which is displayed in a graphical user interface. We will provide a simple starter system which displays a UI with a two input text fields for Wages and Loans; the sum of these is presented in a Total Income output text field when a Calculate button is pressed. Using the starter is optional, you can start from scratch if you prefer.

Functionality and marking are as described below.

Basic system (30 pts)

- 10 pts: user enters three (or more) income fields (wages, loan, other) and the system computes total income when a Calculate button is pressed.
- 10 pts: user enters three (or more) spending fields (food, rent, other) and the system computes total income when a Calculate button is pressed.
- 5 pts: When Calculate is pressed, the system also shows surplus/deficit (income minus spending). This is black if positive or zero, and red if negative.
- 5 pts: System checks user input (numbers for validity), and produces an appropriate error message if input is not valid. Empty fields are treated as 0 (no error message)

Extensions (20 pts)

- 10 pts: allow users to specify income/expenditure numbers per week, per month, or per year, for each input field. This requires both adding appropriate choice widgets (such as combo boxes), and also modifying the way you calculate totals. You can assume that there are 52 weeks in a year, 12 months in a year, and 4.3333333 weeks in a month.
- 10 pts: implement "spreadsheet" behaviour, that is totals are updated whenever the user changes a number or time-period, with no need to press a Calculate button. You should update whenever the focus shifts, as well as whenever an action is performed.

Undo (30 pts)

- 10 pts: implement a single-level of Undo, so the user can "undo" his or her most recent action. This require saving the state of the system (ie, all numbers) when a change is made.
- 10 pts: implement multiple Undo, so the user can undo multiple actions. We recommend that you create a class to hold state information numbers, and then maintain a stack of states.
- 10 pts: Good collection of JUnit tests for Undo

Programming style (20 pts)

- 10 pts: Java code follows good coding style, including good code comments and variable names
- 10 pts: Java code is well structured and decomposed into methods; for example win/loss detection is done using methods that check for a win in a specified row or column.

Please note that no marks are given for visual appearance or visual design of the user interface. We are marking purely on the basis of functionality and programming style.

Submission Instructions

This assessment is due at **1700PM** on **Thursday, 5 December 2024**

Please submit in MyAberdeen

- Coversheet for the assessment
- A zip file containing your code
- A README file explaining what you have implemented (text or PDF)

It is your responsibility to ensure that your code runs in VSCode. If you develop it in another IDE such as IntelliJ, you need to ensure it runs in VSCode, and does not require special IntelliJ libraries,