**CRITERION C – DEVELOPMENT**

**🡪 Suggested Structure:**

Technique / Link to success criterion or criteria / Justify said link 🡨 (How the technique helps meet the success criteria / Explain the implementation of your technique [in your code, solution] / Code sample

**🡪 Examples of Possible Techniques:**

1. Menus/menu driven interface 🡪 (SWITCH/CASE)
2. Inheritance (is-a)
3. Dependency (uses-a)
4. Association (has-a): Aggregation / Composition
5. Encapsulation (relates to abstraction as well; you may talk about the choice of methods included in your class[es] –excluding constructors, accessors and mutators)
6. Data hiding (related to encapsulation as well – *private* attributes, accessors and mutators)
7. Polymorphism: Overriding 🡪 subclass overrides method from its superclass ( e.g. toString() )
8. Data structures (Arrays, ArrayLists, Linked Lists, etc.)
9. Files 🡪 permanent storage (load, save methods)
10. Traversals 🡪Loops (FOR, WHILE)
11. Validation 🡪 Loops & Conditionals (WHILE, IF)
12. Java built-in classes (SCANNER, DATE, etc.)
13. Searching (sequential or binary search)
14. Sorting (bubble/selection/insertion sort)
15. File I/O: Plain text, CSV, Serialisation
16. Polymorphism: Overloading 🡪 methods with same name, but with different return types & parameters/arguments
17. Exception handling 🡪 (throws+) try…catch
18. GUI (not recommended): Use of Swing/AWT/JavaFX classes; Event handling; Layout management; Use of a visual GUI builder (such as Netbeans)

**[TURN OVER]**

**🡪 Sample Paragraph & format for criterion C:**

**Technique:** Aggregation

**Link to success criterion/criteria:**

Store data in an organised and structure way

Structure data so that is it easy to process

**Justification:** Manage data of expenses and a budget which are both linked to each user of the system. *+ Justify the link to one or more success criteria*.

**Explanation:**

The ***User*** class has a ***Budget*** object as one of its attributes (fields). The ***User*** class also has an array of ***Expense*** objects (size 999, as a home user would rarely have more expenses in one month) to keep track of money spent and budget for each user.

**Code sample:**

[ Include the lines of code showing the class name and attributes of the ***User*** class, clearly showing how it contains ***Budget*** and ***Expense*** objects inside; show ***User***‘s constructor(s) if all such code is not too long. ]

*Useful links:*

* <https://nirajrules.wordpress.com/2011/07/15/association-vs-dependency-vs-aggregation-vs-composition/>
* <https://beginnersbook.com/2013/03/polymorphism-in-java/>

**[TURN OVER]**

**CRITERION D – DEVELOPMENT**

**🡪 Modification and expansion of the system** (post-completion of the development stage)

This is part of criterion D of your IA.

***Suggestions:***

* Add your final UML class diagram (made with EasyUML or a similar plugin of functionality of your advanced IDE) to the end of the document for criterion B (Solution Overview/Design)
* Explain any modifications you made to your original class design (describe the modifications and justify each of them)
* Create a table including the most important classes in your system and their responsibilities / purposes; include comments on how these classes may be modified by future developers in order to keep up with increasing demands or changes (see example on next page)

|  |  |  |
| --- | --- | --- |
| Class | Responsibility | Modification/expansion |
| User | Defines the attributes and methods related to each family member who is given a budget and is allowed to have expenses. | The expenses array may be replaced by an *ArrayList<Expense> expenses = new ArrayList<>();* in the User class, after importing java.util.ArrayList. Class constructors and toString methods in *User* will need to be modified accordingly, plus any array references in the *UserManager* class: add, remove, edit expense, plus save and load methods. |
| Budget | Defines the attributes and methods related to the monthly budget of each user. | The *calculateGST* method uses GST, a class constant (final), type double declared and initialised on line 3 of the Budget class to 0.07, the prevailing Goods and Services Tax where the client resides and at the time when this IA was developed. Modify this constant and re-compile the code if there are changes in the GST. |
| … | … | … |