Lesson 3.XX1 — Programming Project (FracCalc Alternative)

Overview

Objectives — *Students will be able to...*

- **Conduct user-centred research** to identify specific functions for a specialized calculator application
- **Plan and create** a calculator that perform specialized operations for an end-user
- **Test, evaluate, and share** the end product

Assessments — *Students will...*

- **Apply** advanced data and control concepts covered in Unit 3
- **Submit** a complete, functional program

Pacing Recommendation

- This project is intended to be the same length as the FracCalc Project.
- The duration of this project is at the discretion of the teacher. About 4-5 classes are recommended.
- Project involves conducting research work (survey or interviews), and communicating with end-user, outside of the classroom.

Implementation Details

Complexity and Creativity

This project is an alternative to the FracCalc project in the existing AP Computer Science course. Students should come up with the idea themselves, based on user-centred research, and ideate a calculator application to address the needs of a specific user group. The "calculator application" could possibly involve several steps, such as solving the quadratic formula, or sharing the cost of a party.

Students will follow applied design process to implement the idea. You should talk to your teacher often to ensure that your progress is in-line with expectations.

Documentation and Style

As with all projects, your program must be well-written, well-documented, and readable. Writing code with good style is always good idea. This will help you debug, pick up where you left off each day, and keep track of progress.

STEP 1 - UNDERSTANDING CONTEXT

Conduct user-centred research to find design opportunities and barriers.

Select an end-user for whom you will design and create this program (this can be a friend, classmate, relative, etc). Create interview questions that will allow you to understand the end-user's interests and likes/dislikes. Since we are creating a cacluator application, here are some possible questions: * when was the last time you used a calculator? * what do you use it for? * what are your most common uses for the calculator?

As the interview progresses, you may prompt them with ideas, but also give them time to think. Some possible uses for the calculator may be: calculate how to share costs for meal; calculate the cost of something with a special discount (eg, "buy one, and get one half price"); or calculate cost of something when travelling in foreign country (and compare with the cost of same item back home!).

You may also ask the user about more complex problems that requires a formula (or several steps) to solve. Examples: * solving the quadratic formula * factoring polynomials * cost sharing (eg: 5 people all bought something for the party, how much does each person "owe", or "gets paid back") * difference in cost for filling up gas in the US vs in Canada (involves metric/imperial units, and currency conversion)

At this point, you will need to ask more specific questions: * "what type of conversions do you do the most?" * "give me an example of some calculation related to [something they mentioned]" * "what would really be handy in this situation?

STEP 2 - DEFINING AND IDEATING

Choose a design opportunity and point of view, make inferences about limitations and boundaries. Take creative risks to indentify gaps to explore, generate a range of possibilities, prioritize ideas for prototyping.

Using the responses from your end-user interview, begin to develop a plan for your custom calculator. Given the duration of the project, you should limit your calculator to do *an interesting calcuation* for a *specific end-user*. At this point you will only be be to implement a text-based user interface (ie, no graphical elements, like buttons or scrollbars). Keep it simple and easy to use.

List: * the types of calculator functions it will do: * what will the user input be? * what output will be calculated? * what are example data for testing? * what user interaction with the program will the user have? * a message to prompt for input? * a mesage to report on the caluclated output? * type a word ("quit") to exit?

Share these ideas with your end user. Record their comments, suggestions and feedback and note any changes that you may make as a result of this interview.

Identify any issues or problems that might arise as you begin to program (what areas might require more information or programming solutions? Where can you find this information?)

STEP 3 – PROTOTYPING AND TESTING

Construct prototypes, making changes to code as needed.

Program your calculator. Be sure to review course notes and activities to make sure that you effectively implement object oriented programming design for your appliction.

When you are ready, create a short test plan and test your program. Include expected output and actual output and include details related to any fixes that needed to be made. A good motto to remember is **code a little, test a little**

Have your end-user try a working version of your program. Note their suggestions, comments and feedback. Indicate any changes that you made to the program, based on the user's feedback.

STEP 4 – SHARING, TESTING AND FINAL ITERATION

Gather feedback from users over time to critically evaluate your design and make changes to product design or processes Identify new design issues

Share your work with other classmates, friends or family. Record any feedback, suggestions and comments and use this information to make the final iteration of your program.

Grading Scheme/Rubric	
Implementation	
Project is appropriately complex and creative	4 points
Program is well-documented and shows good style	10 points
Final product meets all requirements and goals laid out in checkpoint specifications	8 points
Program uses programming concepts effectively, including all required elements with an appropriate level of complexity	4 points
Object Oriented Programming concepts are effectively applied with an appropriate level of complexity	6 points