Faculty of Science, Engineering and Technology

Introduction to Programming

Pass Task 3.2: Simple Menu

Overview

In this task you combine control flow with modularization to create the basis of a menu system for a text-based (terminal based) Music player.

Purpose: Learn to combine control flow and modularisation.

Task: Create a program that implements a basic menu system.

Time: This task should be completed before the start of week6.

Resources:

- **Using a case statement in Ruby**
- Chapters 6 and 8 of 'Learn to Program'

  Chapt 6: Flow Control

  Chapt 8: Writing Your Own Methods

Note: Remember to submit **all tasks** to Canvas for assessment. Also make sure you *fix and resubmit* any tasks you did not get signed off last week!

Submission Details

You must submit the following files to Canvas:

- SimpleMenu program source code.
- A screen shot of the program running

Make sure that your task has the following in your submission:

- Demonstrates use of Ruby programming convention, including indentation within selection and repetition statements.
- Demonstrates use of a **case** statement to perform selection, and a **repeat until** loop to perform repetition.

SWINBURNE UNIVERSITY OF TECHNOLOGY

Instructions

Modify a small program that will give the user the following options:

1. Enter or update an album
2. Play an existing album
3. Exit the system

1. Download and extract the resources for this task.
2. Open **simple_menu.rb** using Atom (or similar).
3. Implement a **maintain_albums** procedure with the following logic:
   - A repeat-until loop with the following:
     - Display the sub-menu options and prompt the user to select one
     - Read in the user selection and then use a case statement to call a procedure that corresponds to the selection (or leave the loop if they chose exit)
     - The called procedure should display a message to user about the option. You do not need to implement the procedure at this stage, just leave it as a 'stub' (a placeholder procedure/function) that displays a message.

```
Create a stub (a placeholder procedure/function) for main
menu option 1.

Steps:

 1: The program displays the menu options:
     'Main Menu: Enter your selection:'
     '1 To Enter or Update Album'
     '2 To Play Existing Album'
     '3 Exit'


 2: The program displays the sub-menu options:
     'Sub Menu Maintain Albums: Enter your selection:'
     '1 To Update Album Title'
     '2 To Update Album Genre'
     '3 To Enter Album
     '4 Return to Main Menu'

 3: For each sub-menu option you need to create/write a
 stub (a placeholder procedure/function).

 Each stub should display a message, wait until the user
 presses enter, then return to the calling menu or sub-
 menu.
```

NB: **Use the stub *play_existing_album()* as an example for the other stubs.**

The output of this program might look as follows:

*(NB: be careful, you should NOT be calling **main()** from the submenus!)*

```
Main Menu:
1 To Enter or Update Album
2 To Play Existing Album
3 Exit
Please enter your choice:
1
Maintain Albums Menu:
1 To Update Album Title
2 To Update Album Genre
3 To Enter Album
4 Exit
Please enter your choice:
1
You selected Update Album. Press enter to continue

Maintain Albums Menu:
1 To Update Album Title
2 To Update Album Genre
3 To Enter Album
4 Exit
Please enter your choice:
2
You selected Update Album Genre. Press enter to continue

Maintain Albums Menu:
1 To Update Album Title
2 To Update Album Genre
3 To Enter Album
4 Exit
Please enter your choice:
3
You selected Enter Album. Press enter to continue

Maintain Albums Menu:
1 To Update Album Title
2 To Update Album Genre
3 To Enter Album
4 Exit
Please enter your choice:
4
Main Menu:
1 To Enter or Update Album
2 To Play Existing Album
3 Exit
Please enter your choice:
2
You selected Play Existing Album. Press enter to continue
```

Upload to Canvas:

1. Your *simple_menu.rb* code
2. A screenshot of your code running

End of Task