

CPSC 2720 – Assignment 3

Media Player Simulation

Overview

In this assignment, you will:

- Design a simple software system using four design patterns.
- Keep track of your progress using version control.
- Document your implementation using `doxygen`.
- Use various software engineering tools to help create quality software (code coverage, static and style analysis, memory leak checking, continuous integration).

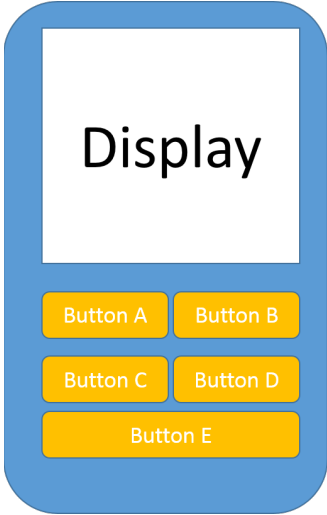
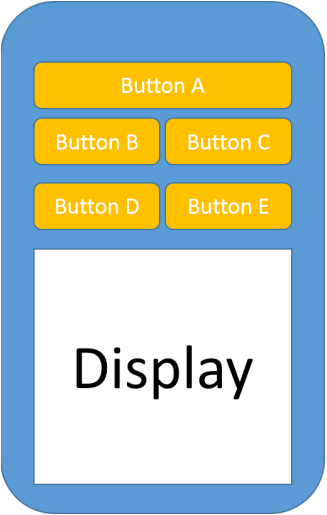
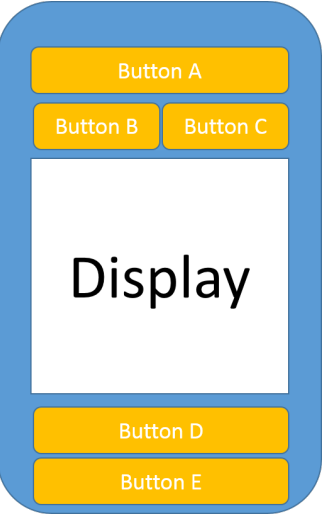
PROBLEM DESCRIPTION

You have been charged with writing the software for a new series of a portable media players that your company is creating. All players have a display and five buttons.

The player can have one or more of three different modes: `RADIO`, `MP3 Player`, and `Video Player`. The buttons can be assigned one of five different functionalities, which behave differently depending on the mode:

	RADIO	MP3	VIDEO
PLAY	Turn on radio	Play Song	Play Video
PAUSE	Turn off radio	Pause Song	Pause Video
REWIND	Tune to next radio station that can be picked up (seek backwards)	Skip to previous song	Skip to previous video
FORWARD	Tune to next radio station that can be picked up (seek forwards)	Skip to next song	Skip to next video
CHANGE MODE	Change the next mode, if there is one.		

Although each player has a display and five buttons, the actual physical configuration can vary. Examples of some of the planned media players are shown in the table below.

	<u>Model T</u>	<u>Model B</u>	<u>Model M</u>
			
<i>Button A</i>	Pause	Change Mode	Play
<i>Button B</i>	Play	Rewind	Rewind
<i>Button C</i>	Rewind	Forward	Forward
<i>Button D</i>	Forward	Play	Pause
<i>Button E</i>	Change Mode	Pause	Change Mode

Each of the models comes in one of three different varieties with different modes for what they can play. When the mode button is pressed, the player cycles through the available modes:

	<i>Radio</i>	<i>MP3</i>	<i>Video</i>
<i>Variant A</i>	Yes	Yes	Yes
<i>Variant R</i>	Yes	No	No
<i>Variant D</i>	No	Yes	Yes

As the hardware is being developed at the same time as the software, you don't have hardware to test with. Instead, you will output a message indicating the activated functionality when a button is pressed.

<i>Functionality</i>	<i>Message</i>		
Mode	Radio	MP3	Video
<i>Play</i>	Playing Radio	Playing Song	Playing Video
<i>Pause</i>	Stopping Radio	Pausing Song	Pausing Video
<i>Rewind</i>	Seeking Previous Station	Previous Song	Previous Video
<i>Forward</i>	Seeking Next Station	Next Song	Next Song
<i>Change Mode</i>	Changing Mode		

The software will keep track of the number times each button is pressed, to understand the expected lifetime of the buttons on the player, and the number of presses can be reported for each button. However, in the future other information about the button is expected to be collected.

INSTRUCTIONS

Setup

1. Fork the repository at <http://gitlab.cs.uleth.ca/cpsc2720/DesignPatterns/MediaPlayer>. As it is a CS department server, you will only be able to do this on the campus network (or via VPN).
2. Set your notification settings for this repository to "Watch" so you will receive email notification if there are any changes to repository (e.g. clarifications are added to the instructions).
3. Fork the repository so you have your own copy.
4. Set the project visibility for your forked repository to "Private".
5. Add the marker and instructor as a member of your project with the permission "Reporter". You will be provided with their CS department user name in the lab and/or on Moodle. This is needed so the marker can grade your assignment.
6. Setup your GitLab repository for running continuous integration for your project.
 - a. Set the *Git Strategy* to "git clone"
 - b. Set the Timeout to 5 (i.e. 5 minutes). Your CI job will be small, so this should be lots of time and will prevent any infinite loops from tying up the CI server
7. Create a local clone of your forked repository.

Design

1. Create a UML static structure (i.e. class) diagram using `Dia` that shows the design of your software.
2. Export your diagram to `PNG` format.
3. Put the image of your design in a `design` folder at the root of your repository.

You are expected to use the following design patterns in your design.

<i>Design Pattern</i>	<i>Functionality</i>
Factory Method	For creating the nine possible players. You can refer to each one by their model and variant using the pattern <code>M<model>V<variation></code> . For example, a Variant A of the Model T hardware would be referred to as “MTVA” and the Variant R of the Model B hardware would be “MBVR”.
State	For handling the different modes that a player can have and the different behavior of the buttons depending on the mode.
Command	For assigning the actions to the different buttons.
Observer	For tracking the number of times a button is pushed, reporting this number, and allowing for other information to be collected in the future.

Implementation

Implement your design. It is suggested that you proceed in an incremental manner as follows:

1. Implement a Factory Method that creates a player with one mode.
2. Implement the State pattern by adding more modes to the player, and update the Factory Method to support the different variations for the model you are using for testing.
3. Implement the Command Pattern and add support for the different models to the Factory Method.
4. Implement the Observer Pattern.

A `Makefile` and CI configuration file (`.gitlab-ci.yml`) have been provided for you to help you run and test and your code.

NOTES

- A `Makefile` is provided which:
 - Builds and runs a testing executable (`make tests`).
 - Checks for memory leaks (`make memcheck`)
 - Runs static analysis (`make static`)
 - Runs style checking (`make style`)
 - Runs code coverage (`make coverage`)

- Runs all of the checks (`make all`)
- A continuous integration configuration file (`.gitlab-ci.yml`) is provided for you. It is not expected that you will need to change this file.

GRADING

You will be graded based on your demonstrated understanding of the use of version control, good software engineering design practices, and design patterns. Examples of items the grader will be looking for include (but are not limited to):

- Design and implementation shows use of requested design patterns.
- Version control history shows an iterative progression in completing the assignment. It is expected that you will have at least 10 new commits in your repository (roughly one commit for each design pattern and accompanying test fixture).
- Version control repository contains no files that are generated by tools (e.g. object files, binary files, documentation files)
- Implementation shows understanding of software engineering design principles, specifically for the four design patterns.
- Source code is appropriately documented using the principles discussed in class (e.g. all classes and methods) so that `doxygen` can extract the documentation.
- Code coverage of 90% or better.
- Memory leak checking, static analysis and style analysis show no errors.
- The build on GitLab passes as of the assignment deadline.

SUBMISSION

There is no need to submit anything, as GitLab tracks links to forks of the assignment repository.

- Creating **an independent copy of the repository (i.e. not a fork)** will result in an **automatic 0 (zero)** as the marker will not be able to find it.
- Make sure that the permissions are correctly set for your repository on GitLab so the marker has access. **You will receive an automatic 0 (zero) for the assignment if the marker cannot access your repository.**