Design document contribution

CSC 305 – David H. Brown

This is an *individual* assignment, though part of the overall team project!

In this context, a design document is generally written by and for developers to either document how the software has been written and/or to serve as a road map for further development.

Instructions

Prepare a UML diagram that describes the implementation of some part of your team project. Best choices are a Class or Sequence diagram. If you're describing a part that's implemented as a state machine, a State diagram is okay, too. Less likely to be appropriate is an Activity diagram. Definitely don't use a Use Case diagram: your audience is other developers working on the project (not customers/managers/etc).

Each member of the team should individually document a different aspect of the project. Coordinate who covers what at a team, but each of you needs to work alone to do the diagram and write the explanation. You do not have to be the one who wrote that section of the SRS or programmed that part of the project. It could be reasonable to document the same part of a project in different ways. For example, one team member might do a Class diagram and another a Sequence diagram for the same part of the project.

You might document code you've already written – perhaps to explain to the rest of your team how to use it? – or you might you might imagine leaving instruction/guidance for future developers carrying on with features that couldn't be implemented in the time available.

Include approximately one paragraph explaining how the part fits into the project as a whole, citing a numbered requirement from your SRS. You may expand on and explain the diagram, but consider that if the diagram itself needs too much explanation, you may have selected the wrong type of diagram for what you're trying to communicate.

These individual submissions together will form part of your project's design document to be submitted the week after these.

FAQ

Can this be the same as in my Design Pattern paper

No, but you could cover the same piece of the software. If you do, do not prepare the same type of UML diagram for the Design Pattern paper. If your design pattern paper will cover the same functionality as your design document contribution, I recommend that you prepare a class diagram for the paper and a sequence diagram, activity diagram, or state diagram for this assignment.

(The design patterns book https://learning.oreilly.com/library/view/design-patterns-elements/0201633612/ gives generic class diagrams you can adapt to your specifics, so that makes it very easy to ensure your diagram actually explains how you are implementing the pattern.)

What software should I use?

Some free software I've used is listed in the UML topic in Brightspace. It is not necessary for everyone in the team to use the same tool. You may just draw neatly and take a clear picture using an app like Microsoft Office Lens. (Just a plain photograph will look awful when embedded in the team's design document.)

How big/complex should the diagram be?

Definitely too small: a single class; a single message.

Almost surely too large: your entire application! If a class diagram, see whether you can break it down into modules/packages. Remember that an activity diagram often details a single use case; a sequence diagram might detail a single action (from an activity diagram).

More questions?

Please ask! I will be happy to extend this description if helpful, but I usually write more than people want to read anyway...

What to submit

A single PDF file that includes both the paragraph explanation and the diagram as a single document

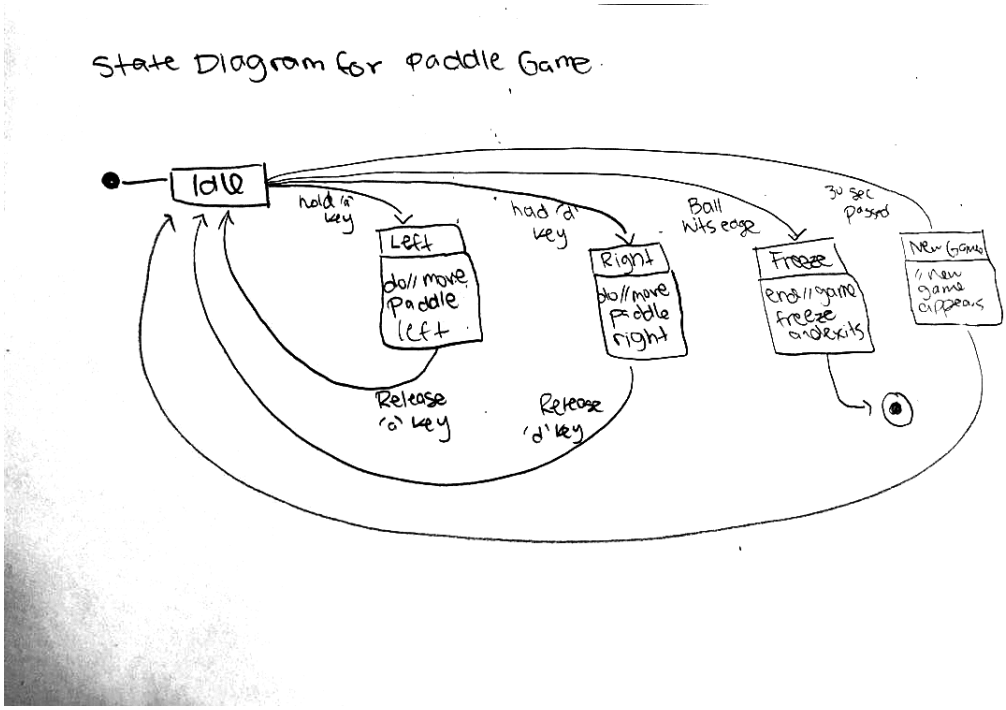
Grading breakdown (50 points total)

- Introductory paragraph: 10pt
- Content is appropriate to design phase: 10pt
- Selection and implementation of diagram: 30pt

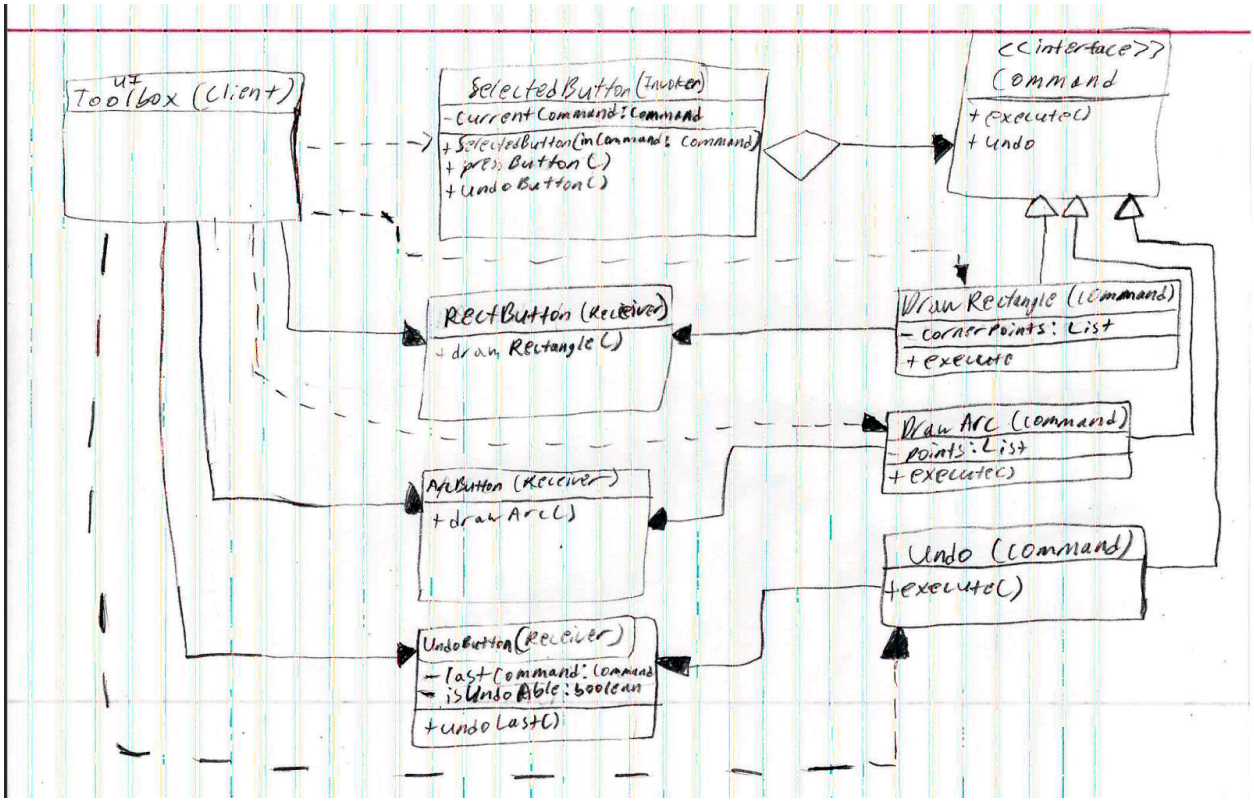
Some good examples from previous semesters follow…

3.2 State Diagram for Paddle Game

The diagram is a model of our SRS document section 3.2 Paddle Boarding and all of the functional requirements for the paddle board game found in 3.2.3. The following state diagram shows the movements of the paddle during the paddle game section of the multi-game feature. The paddle has two main movements after leaving its idle state. They include: moving the paddle left, which is done by pressing the ‘a’ key and moving the paddle right which is done by pressing the ‘d’ key. Another feature of this diagram is the 30-second timer that triggers a new game to appear on the screen as well. This doesn’t affect the current paddle game, hence why the diagram goes back to idle after this timmer is triggered. The game will exit when the ball hits the edge and thus the state diagram ends there.



Minimalistic CAD Undo



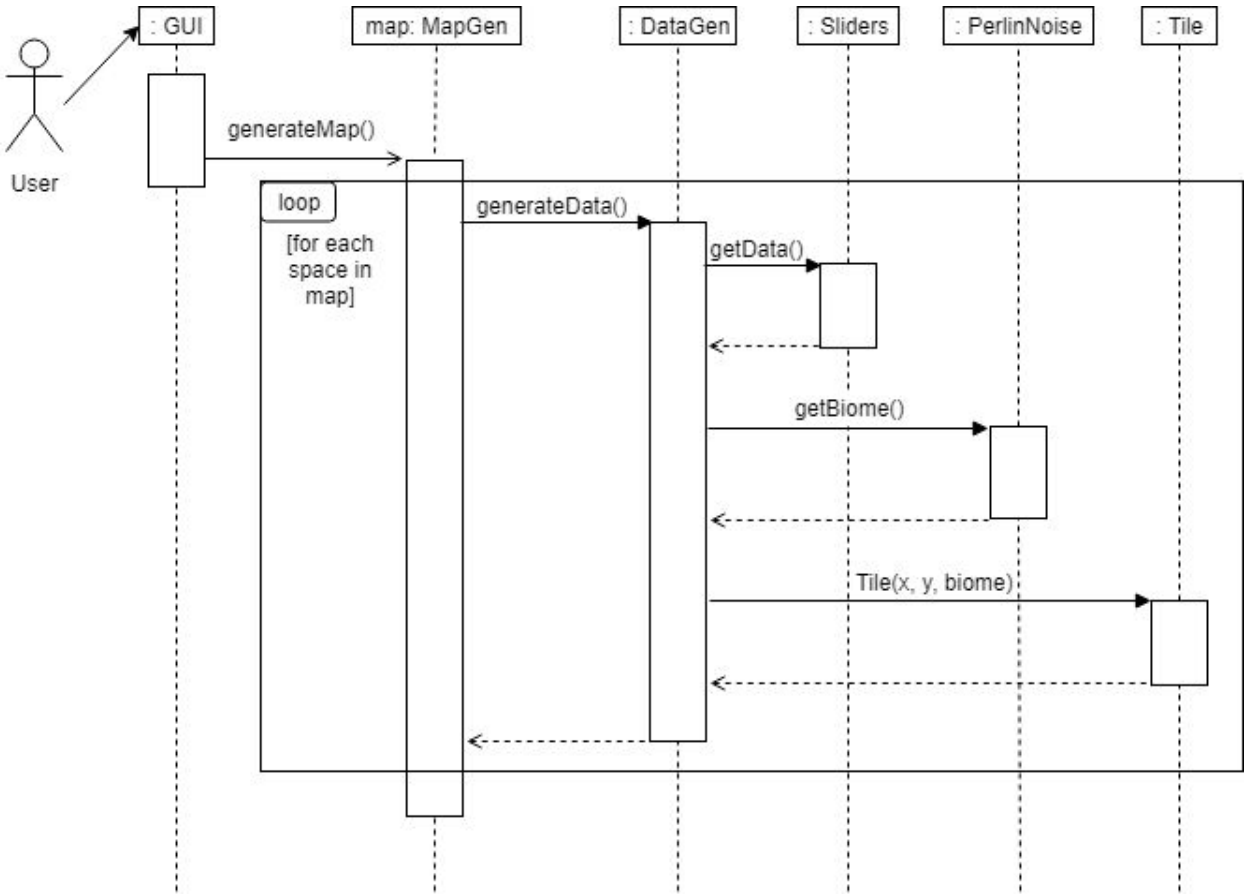
In order to represent our program's Undo Tool (SRS 3.3), I chose to use an UML class diagram. The class diagram is modelled off of the class diagram for a Command design pattern (my chosen design pattern for this feature). The UI Toolbox class is the client, so its job is to draw the buttons in our UI, which will have their associated commands. Note: for brevity I only included 3 buttons and their associated commands. Our actual program will have 16 buttons and commands. The SelectedButton class is the invoker which sends the command to be executed to

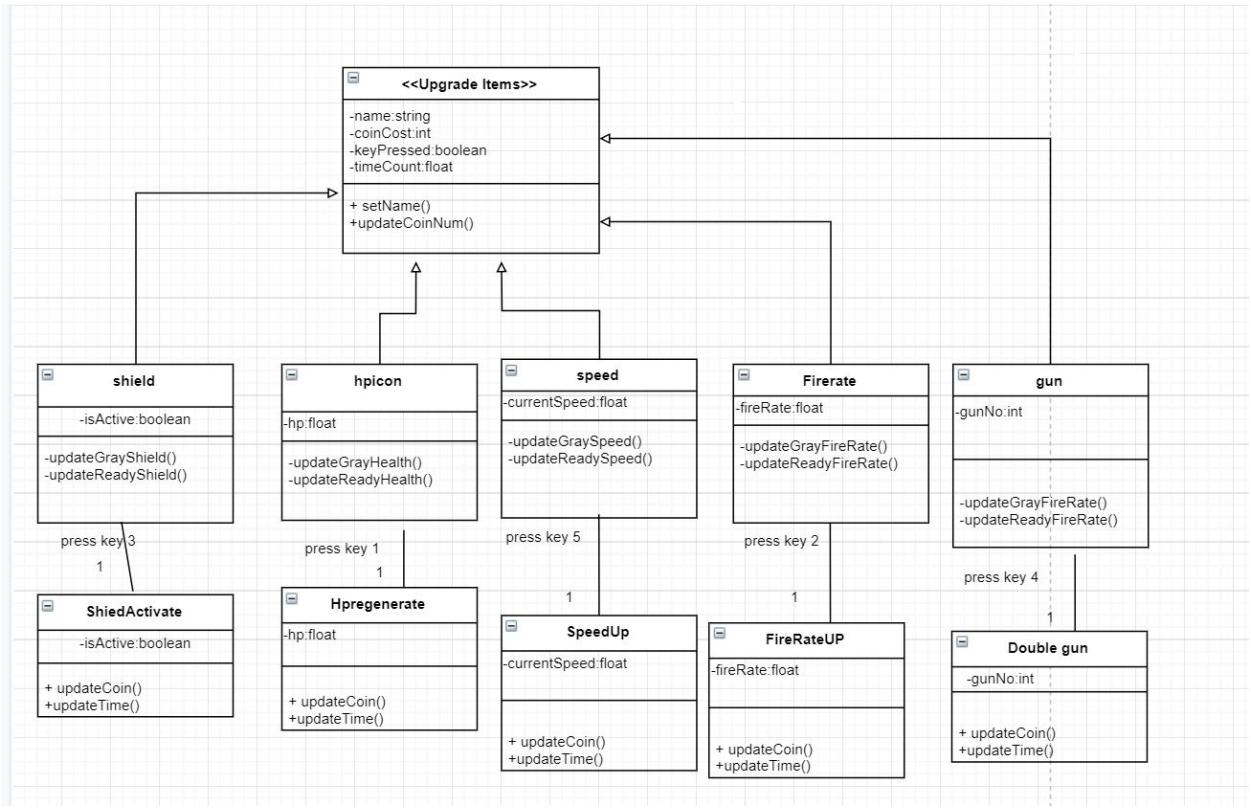
one of our buttons (the receivers). The Command interface calls a specific receiver's method and is used to abstract out the button's commands from the button classes themselves. Each inherited command: DrawRectangle, DrawArc, and Undo, calls an execute command which invokes the corresponding action from the associated receiver. For the Undo command, its execute method will call the inverse of the last reversible command. For example, if a rectangle is drawn based on its corner points, the Undo command will delete the rectangle. The receivers (buttons), when invoked, will execute their action/method. Finally, the diagram fulfills the Undo Tool's functional requirements: 3.3.3.1 Undo UI Button and 3.3.3.2 Keeping Track of the Last Command. 3.3.3.1 is satisfied by the UI Toolbox class (client), which will draw the button for the UndoButton receiver. The diagram completes requirement 3.3.3.2 by having the UndoButton receiver store the last command the user executed in order to reverse that command.

CSC 305

Individual Design Document Contribution

For my UML diagram contribution, I created a sequence diagram for the Map Generation feature (3.1). This diagram explains the sequence of steps required in the map generation process. The user shown in the diagram is not a significant part of the diagram - it is just to signify that the generation must be triggered by the user's interaction with the GUI. The GUI is included because of 3.1.2.1, which specifies that the generation is triggered by a button in the GUI. The implementation of 3.1.3.2 is demonstrated in the interactions between MapGen, DataGen, and Sliders. Through these interactions, MapGen indirectly has access to slider data through methods in these classes, as required in the SRS document. MapGen functions by repeatedly interacting with the DataGen class. The method calls in the diagram may be slightly simplified - there may be more steps involved (for example, in how the biome is determined) - but since this diagram is only meant to represent 3.1 and not any other feature, I left the diagram slightly simplified in terms of the interactions between some of the classes which do not directly interact with the map. The DataGen class creates a Tile with the appropriate biome and resources, and then returns this tile to MapGen, to be placed in the map. This loop must continue until the map is completely filled with Tiles. Once the map is full, map generation is complete.


: GUI
map: MapGen
: DataGen
: Sliders
: PerlinNoise
: Tile
User
generateMap()
loop
[for each space in map]
generateData()
getData()
getBiome()
Tile(x, y, biome)



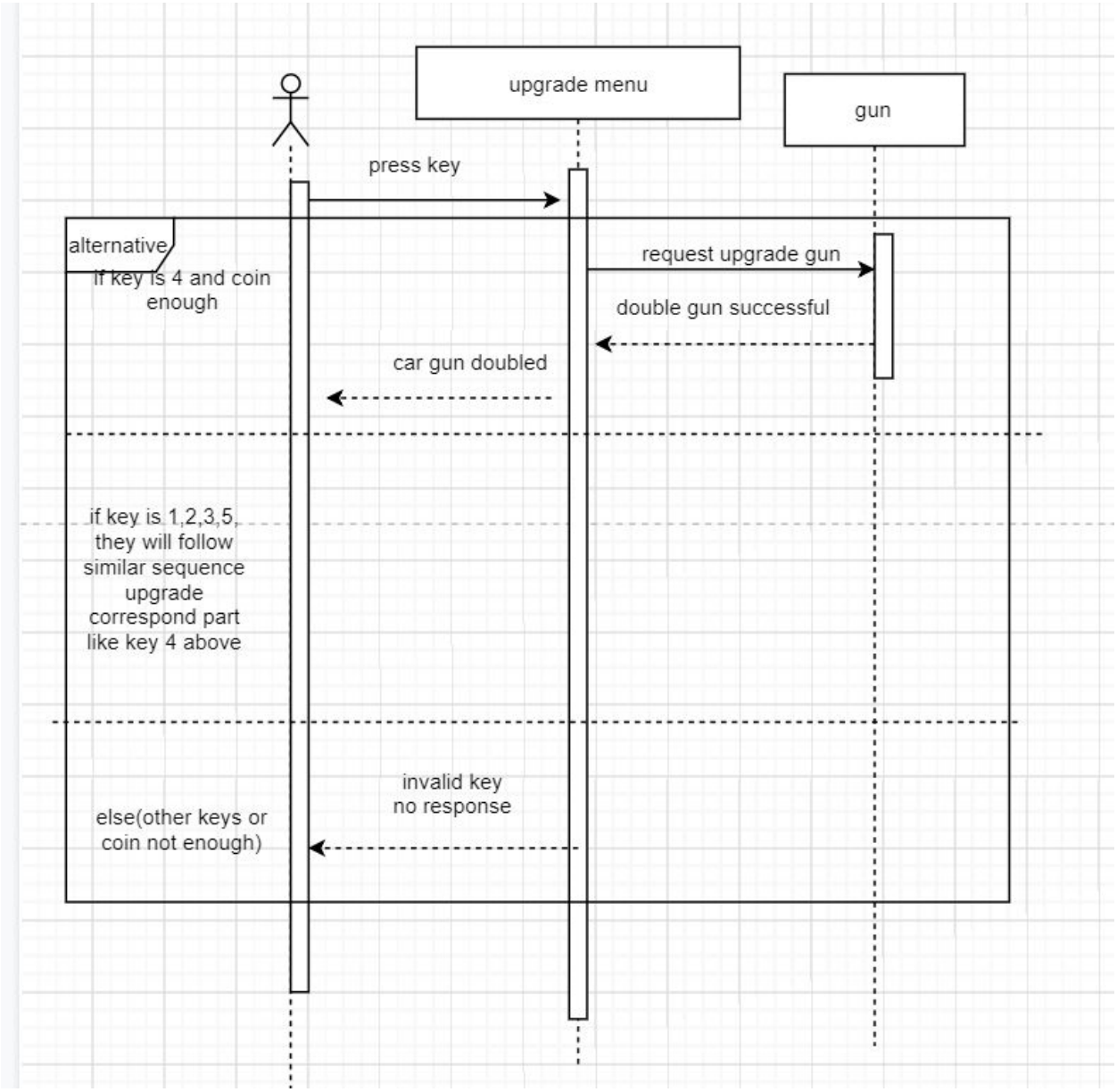
My team asks me to draw the class diagram for what happens when you press keys (1,2,3,4,5) for upgrades.

Upgrade items is an abstract class, it has 5 different parts could be upgraded.The above five class inherit all from the abstract class. Method updateCoinNum is used as checking the coin whether reach the satisfied condition to upgrade parts or not, if yes, then apply the upgrades by pressing number keys (1,2,3,4,5).

In my opinion, I think using class diagram for upgrading selections is suitable, but ‘press keys and responses’ should use sequence diagram. But as I tried to communicate with my team members through discord, they haven’t reply yet. To compromise between my team requirements and my thoughts, I decided to use association relation. Number ‘1’ represent each upgrades could only exist one time,for example, you can’t get triple gun if press key 4 three times.

Each upgrades has time limit, so I’m using updateTime() method to count down the time. I thought composition is an optional way, but it couldn’t express clearly which key for which upgrades.

In the bottom sequence diagram, it shows how the press key reacts. User press keyboards, if press key number four and user have enough coin, the upgrade menu will upgrade gun for user. This is just a single action, we could add press key 1,2,3,5 by adding more objects: hp, fire rate, etc. Draw.io is frustrating. Else (not 1,2,3,4,5 key or not enough coin), upgrade menu would do no action.


upgrade menu
gun
press key
alternative
if key is 4 and coin enough
request upgrade gun
double gun successful
car gun doubled
if key is 1,2,3,5, they will follow similar sequence upgrade correspond part like key 4 above
invalid key
no response
else(other keys or coin not enough)