Cardgame

For the third installment in this magnificent streak of assignments, you will be creating a card game. The goal of this is to help you get familiar with creating a GUI (Graphical User Interface) in Java. To accomplish this you will be using the Model-View-Controller (MVC) pattern. There is a section in the reader about this; please read through it before you begin this assignment.

The GUI will be created using Java Swing. We do not use this because it creates the prettiest applications (although it definitely can), but rather because it nicely teaches you the basics of creating GUIs. There are lots of big applications written using Swing (IntelliJ for example) and it is still widely used.

For this assignment, you will have two weeks and you are allowed to make any card game you want! There is also an optional demo halfway through so that you can receive some intermediate feedback on your program.

1 Workflow

Since you are allowed to pick any card game you want, we cannot give you a very detailed description of how to do this assignment. To provide you at least with something, we will try to help you by explaining how to go about tackling such a larger project. It will save you quite some time if you tackle this problem properly.

The structure explained below will allow you to check if your model works as intended before you put a lot of time and effort into creating the `View`. This allows you to fully focus on figuring out the `View` rather than spending time on your `Model` yet again.

You will find out that most problems arise when you start working on your `View`, so therefore we would highly recommend you follow the following workflow (lot's of flows there):

1.1 Brainstorm

Before you begin, try to think about what kind of game you want to make. Think about what it should be able to do and how you will design it. Then draw a simple diagram showing the connections between the different classes. This is a nice way to have an overview of what you are about to create. At this point it is also a good idea to create a backlog (see next section).

1.2 Write the Model

Start with writing the actual core of your card game. Essentially you want the functionality of the game to already be done before you start working on any sort of UI. It is easier to change some details in the UI compared to changing the Model and having to make a lot more changes to the UI.

1.3 Create a simple GUI

Create a very simple GUI (Just a JFrame with a JPanel) that displays some text messages on the screen so that you can verify that it updates correctly. The interaction at this point can still be through the command line (In this case that will be your `Controller`). This should update the `Model` and the `Model` should, via the `Observer` pattern, update the `View`. The purpose of this step is to make sure that the `Observer` pattern works properly and that your `View` is being updated as your model updates.

1.4 Write the Controller

Think about what type of actions you want to allow your users to do. Then for each of these actions, create a corresponding `Action` (a class that extends `AbstractAction`; see demo program) and attach this to a button. Clicking the buttons should do something with the `Model`. This, in turn, will then update the `View`. It is OK to have all the user interaction through buttons for this assignment, but you can also let the user use the mouse to select things.

1.5 Verify that it works so far

At this point you can still have some simple text `View`. This means that right now, you should be able to verify that your program works. Only if this is the case would we recommend you move on to the next stage.

1.6 Create a proper view

Now you can move on to creating a proper View. After this step, your card game should really look like a card game. Again, make sure to verify that it works!

1.7 Be creative!

Now that the basic card game works, you can go completely mental and add all sorts of awesome things to your game! This is obviously the most fun part. Keep in mind that, while these extras are nice, we are mainly judging your code and design. So a properly designed game without any extra features will likely get a higher grade than a badly designed program with a bunch of extras. Also, a properly designed program will make adding new features much easier!

2 Working with larger projects

As the amount of required code for the assignments increases, so does the difficulty of properly managing such a project. We will provide you with some tips about how to deal with this. Note this part is not mandatory, it will simply make things a lot easier to manage.

2.1 Organising Tasks

Before you start, it is a good idea to have a brainstorm session. Write down all the things you want to do for this project. Now you should have a good idea of what you want to implement. Try to split these things into different tasks and sort them by priority. You now have a list of things you need to do and the order you need to do them in. Awesome! This is called a backlog. Since you are working together, it is a good idea to assign a person to each task (unless you do everything together). The simplest form this takes is a table that looks something like this:

What	**Who**
The user should be able to draw a card	Niels
The user should be able to flip a card	Erblin
World Domination (つ◕ᴥ◕)つ━☆ﾟ.*･｡ﾟ	~~Niels~~ ~~Erblin~~ Niels

It is not necessary to create the entire backlog in one go. You can keep adding new tasks to it throughout the development process. Whenever a task is done, you can marked it as such or remove it. If you decide to

remove it, it is a good idea to save this somewhere so that you have a good idea of what has been done already.

As you can see backlogs are a nice way to keep track of what still needs to be done. You can add a lot more detail to this backlog such as estimations of how long tasks will take, but also splitting up tasks even more. Try not to make your tasks too big, because that will make it difficult to properly distribute tasks and see progress.

There are a few different ways you can make such a backlog. An easy one would be Google Sheets, but a more dedicated tool that you can also go for is Trello, which is what we would recommend.

3 Demo Program

Since Java Swing is better shown than explained, we have provided a demo program. This demo program is located in the `cardgameDemo` directory. Make sure that you do your work in the `assignment_2` directory and not in the `cardgameDemo` directory!

4 Requirements

The requirements for the program are as follows:

- you should have a correctly working card game.
- it should have a GUI;
- it should have a way for the user to interact with the game via the GUI. This does not need to be incredibly complicated. A simple menu with some buttons is enough;
- it should have a correct Model-View-Controller structure. This means:
    - having an Observable Model;
    - having an View that observes the Model. The View should only be changed through the Observer pattern;
    - having a Model should only be changed through the Controller.

As you can see, most of it is up to you and there are many opportunities to add your own spin and all sorts of extras to it. Make sure to show these during the demos!

There is one restriction here: you are not allowed to use `Threads`. These are covered in Advanced OOP and are quite hard to debug.

5 Handing in

You know the drill by now. Create a pull request from `development` into `master`. Due to some issues with CircleCI, you can ignore that output for now; just make sure the code compiles and runs.