Portfolio Part 1: Component Brainstorming

- **Name**: Sophia Rakowsky
- **Dot Number**: rakowsky.12
- **Due Date**: 2/16 @11:59 pm

Assignment Overview

The overall goal of the portfolio project is to have you design and implement your own OSU component. There are no limits to what you choose to design and implement, but your component must fit within the constraints of our software sequence discipline. In other words, the component must extend from Standard and must include both a kernel and a secondary interface.

Because this is a daunting project, we will be providing you with a series of activities to aid in your design decisions. For example, the point of this assignment is to help you brainstorm a few possible components and get some feedback. For each of these components, you will need to specify the high-level design in terms of the software sequence discipline. In other words, you will describe a component, select a few kernel methods for your component, and select a few secondary methods to layer on top of your kernel methods.

You are not required to specify contracts at this time. However, you are welcome to be as detailed as you'd like. More detail means you will be able to get more detailed feedback, which may help you decide which component to ultimately implement.

Assignment Checklist

Because these documents are long and full of text, you will be supplied with a quick overview of what you need to do to get the assignment done as follows:

Getting Started Tasks

- [\ ] I have added my name to the top of this document
- [\ ] I have added my dot number to the top of this document
- [\ ] I have added the due date to the top of this document
- [\ ] I have read the assignment overview in the "Assignment Overview" section
- [\ ] I have read the assignment learning objectives in the "Assignment Learning Objectives" section
- [\ ] I have read the assignment rubric in the "Assignment Rubric" section
- [\ ] I have read this checklist

Ongoing Tasks

- [\ ] I have shared my interests in the "Pre-Assignment" section
- [\ ] I have drafted my first component idea in the "Component Designs" section
- [\ ] I have drafted my second component idea in the "Component Designs" section
- [\ ] I have dradted my third component idea in the "Component Designs" section

Submission Tasks

- [\ ] I have shared assignment feedback in the "Assignment Feedback" section
- [\ ] I have converted this document to a PDF
- [\ ] I am prepared to submit the PDF on Carmen
- [\ ] I am prepared to give my peers feedback on their ideas

Assignment Learning Objectives

Without learning objectives, there really is no clear reason why a particular assessment or activity exists. Therefore, to be completely transparent, here is what we're hoping you will learn through this particular aspect of the portfolio project. Specifically, students should be able to:

1. Integrate their areas of interest in their personal lives and/or careers with their knowledge of software design
2. Determine the achievablility of a software design given time constraints
3. Design high-level software components following the software sequence discipline

Assignment Rubric: 10 Points

Again, to be completely transparent, most of the portfolio project, except the final submission, is designed as a formative assessment. Formative assessments are meant to provide ongoing feedback in the learning process. Therefore, the rubric is designed to assess the learning objectives *directly* in a way that is low stakes—meaning you shouldn't have to worry about the grade. Just do good work.

1. (3 points) Each design must align with your personal values and long-term goals. Because the goal of this project is to help your build out a portfolio, you really ought to care about what you're designing. We'll give you a chance to share your personal values, interests, and long-term goals below.
2. (3 points) Each design must be achievable over the course of a single semester. Don't be afraid to design something very small. There is no shame in keeping it simple.
3. (4 points) Each design must fit within the software sequence discipline. In other words, your design should expect to inherit from Standard, and it should contain both kernel and secondary methods. Also, null and aliasing must be avoided, when possible. The methods themselves must also be in justifiable locations, such as kernel or secondary.

Pre-Assignment

> Before you jump in, we want you to take a moment to share your interests below. Use this space to talk about your career goals as well as your personal hobbies. These will help you clarify your values before you start brainstorming. Plus it helps us get to know you better! Feel free to share images in this section.

I love cards and card games, so I am planning on designing a card and deck that function the same way a real one would, with the intention of potentially using them for creating a game

Assignment

As previously stated, you are tasked with brainstorming 3 possible components. To aid you in this process, we have provided some example components that may help you in your brainstorming. These components are organized by CSE specializations.

- Artificial Intelligence
  - Artificial Neuron
  - Activation Function
  - Graphs
  - Search
- Computer Graphics and Game Design
  - Pixel
  - Polygon
  - Geometry
  - Pathing
- Database Systems and Data Analytics
  - Object Relational Mapping (ORM)
  - Any Data Model
  - Scraping
  - Data Mining
- Information and Computation Assurance
  - Password Hashing
- Computer Networking
  - Packet
  - MAC Address
  - IP Address
- Computer Systems
  - Complex Logic Gate
  - Semiconductor
- Software Engineering
  - Unit Testing
  - Performance Testing
- Individualized Option (e.g., X + CS)
  - Mathematical Matrix
  - Music Playlist

There is no requirement that you make use of any of the subdisciplines above. If you want to model something from the real-world, go for it! Very common early object projects usually attempt to model real-world systems like banks, cars, etc. Make of this whatever seems interesting to you, and keep in mind that you're just brainstorming right now. You do not have to commit to anything.

Example Component

With all that in mind, here's an example component you can use for reference. Feel free to use this example as the groundwork for your own components below.

- Example Component: `Point3D`
  - **Description**: The purpose of this component is to model a 3-dimensional point. Our intent with this design was to keep a simple kernel that provides getters and setters for the three underyling coordinates. Then, we provide more complex mathematical operations in the secondary interface. It might be possible to create an immutable version of the class by removing the setters and instead having all secondary methods return new points. However, this design seems more inline with NaturalNumber.
  - **Kernel Methods**:
    - `double getX()`: gets the x-coordinate of `this`
    - `double getY()`: gets the y-coordinate of `this`
    - `double getZ()`: gets the z-coordinate of `this`
    - `double setX(double x)`: sets the x-coordinate of `this`
    - `double setY(double y)`: sets the y-coordinate of `this`
    - `double setZ(double z)`: sets the z-coordinate of `this`
  - **Secondary Methods**:
    - `void translate(double x, double y, double z)`: moves `this` by translating each coordinate by `x`, `y`, and `z`, respectively
    - `void translate(Point3D p)`: moves `this` by treating `p` as a vector
    - `void move(double x, double y, double z)`: moves `this` by overwriting each coordinate with `x`, `y`, and `z`, respectively
    - `void move(Point3D p)`: moves `this` to `p`
    - `double distance(Point3D p)`: gives the distance between `this` and `p`
    - `Point3D vectorTo(Point3D p)`: gives the vector from `this` to `p`
    - `boolean isOrigin()`: returns `true` if all three coordinates are zero

Keep in mind that the general idea when putting together these layered designs is to put the minimal implementation in the kernel. In this case, the kernel is only responsible for giving back the coordinates of the point. The secondary methods use these getters and setters to perform more complex operations like translating the point or computing the distance between two points.

Also, keep in mind that we don't know the underlying implementation. It would be completely reasonable to create a `Point3D1L` class which layers the kernel on top of the existing `Point` class in Java. It would also be reasonable to implement `Point3D2` on top of three doubles and `Point3D3` on top of an array. Do not worry about your implementations at this time.

On top of everything above, there is no expectation that you have a perfect design. Part of the goal of this project is to have you actually use your component once it's implemented to do something interesting. At which point, you will likely refine your design to make your implementation easier to use.

Component Designs

> Please use this section to share your designs.

- Component Design #1: `Deck`

- **Description**: The purpose of this component is to model a playing card. My intent with this design was to keep a simple kernel that provides adding to and removing removing from the deck, and the size. Then, I provide more complex shuffling, distributing, and viewing operations in the secondary interface. It is possible to create an version which only allows the deck length to go up to 52 (the number of cards in a deck) and to mix decks to create a bigger set of cards to work with. This will have an interface Deck.Card that has variables val, color, suit, maybe vis(ibility). Still deciding if I want index to be 0 to length - 1 (easy implementation) or 1 to length (easy for user), will likely do easy for user
- **Kernel Methods**:
  - `void addCard(int index, Deck.Card card)`: adds `card` to `index` of `this`
  - `Deck.Card removeCard(int index)`: gets and removes the `Deck.Card` at `index` of `this`
  - `int length()`: gets the number of Deck.Cards in `this`
  - `boolean hasCard(Deck.Card card)`: returns true if `card` is in `this`
  - `Deck.Card seeCard(int index)`: gets the card at `index` of `this`
  - `int position(Deck.Card card)`: gets the index of `card` in `this` //should this have the suit,color,val as param or deck.card //implement on sequence
- **Secondary Methods**:
  - `void moveCard(int start, int end)`: changes location of card at index `start` to index `end` in `this`
  - `void addDeck(Deck deck)`: appends `deck` to `this`
  - `Deck cut(int index)`: gets and removes all Deck.Cards in `this` before `index`
  - `void deal(int amount, Deck[] hands)`: deals `amount` to each `hand` from `this`
  - `void shuffle()`: randomizes the order of Deck.Cards in `this` // - `void sort(Comparator<Deck.Card> order)`: orders the Deck.Cards in `this` based on SortingMachine
  - //clear?
  - //rotate?
  - //reverse? //implement on DeckKernal

- Component Design #2: `Matrix`
  - **Description**: The purpose of this component is to model a matrix. My intent with this design is to create the basic structure of a matrix in the Kernal, including setting and getting the values at each position. In the Secondary methods, I plan to include more complex mathematical operations.
  - **Kernel Methods**:
    - `int rows()`: gets the number of rows in `this`
    - `int columns()`: gets the number of columns in `this`
    - `void change(int column, int row, int value)`: sets `value` to position [`column`, `row`]
    - `int get(int column, int row)`: gets value at [`row`, `column`] of `this`

    - **Secondary Methods**:
        - `void add(Matrix matrix)`: adds `matrix` to `this`
        - `void subtract(Matrix matrix)`: subtracts `matrix` to `this`
        - `void multiply(Matrix matrix)`: multiplies `this` by `matrix`
        - `void transpose()`: transposes `this`
        - `void inverse()`: changes `this` to inverse of `this`

- Component Design #3:`TikTakToe`

    - **Description**: The purpose of this component is to model a TikTakToe game. My intent is to have the ability to make a move in the kernal, and the check for if a game has been won in the Secondary Methods.
    - **Kernel Methods**:
        - `boolean canMove(int row, int column)`: determines if the (`row`, `column`) in `this` is empty
        - `int move(int row, int column, char player)`: sets `player` to (`row`, `column`) in `this`
    - **Secondary Methods**:
        - `boolean movesLeft()`: determines if there are any empty spots left in `this`
        - `boolean win()`: determines if there are of the same moves in a row in `this`

Post-Assignment

The following sections detail everything that you should do once you've completed the assignment.

Submission

If you have completed the assignment using this template, we recommend that you convert it to a PDF before submission. If you're not sure how, check out this Markdown to PDF guide. However, PDFs should be created for you automatically every time you save, so just double check that all your work is there before submitting.

Peer Review

Following the completion of this assignment, you will be assigned three students' component brainstorming assignments for review. Your job during the peer review process is to help your peers flesh out their designs. Specifically, you should be helping them determine which of their designs would be most practical to complete this semester. When reviewing your peers' assignments, please treat them with respect. Note also that we can see your comments, which could help your case if you're looking to become a grader. Ultimately, we recommend using the following feedback rubric to ensure that your feedback is both helpful and respectful (you may want to render the markdown as HTML or a PDF to read this rubric as a table).

Criteria of Constructive Feedback	**Missing**	**Developing**	**Meeting**

Criteria of Constructive Feedback	Missing	Developing	Meeting
Specific	All feedback is general (not specific)	Some (but not all) feedback is specific and some examples may be provided.	All feedback is specific, with examples provided where possible
Actionable	None of the feedback provides actionable items or suggestions for improvement	Some feedback provides suggestions for improvement, while some do not	All (or nearly all) feedback is actionable; most criticisms are followed by suggestions for improvement
Prioritized	Feedback provides only major or minor concerns, but not both. Major and minar concerns are not labeled or feedback is unorganized	Feedback provides both major and minor concerns, but it is not clear which is which and/or the feedback is not as well organized as it could be	Feedback clearly labels major and minor concerns. Feedback is organized in a way that allows the reader to easily understand which points to prioritize in a revision
Balanced	Feedback describes either strengths or areas of improvement, but not both	Feedback describes both strengths and areas for improvement, but it is more heavily weighted towards one or the other, and/or descusses both but does not clearly identify which part of the feedback is a strength/area for improvement	Feedback provides balanced discussion of the document's strengths and areas for improvement. It is clear which piece of feedback is which
Tactful	Overall tone and language are not appropriate (e.g., not considerate, could be interpreted as personal criticism or attack)	Overall feedback tone and language are general positive, tactul, and non-threatening, but one or more feedback comments could be interpretted as not tactful and/or feedback leans toward personal criticism, not focused on the document	Feedback tone and language are positive, tactful, and non-threatening. Feedback addesses the document, not the writer

Assignment Feedback

> Now that you've had a chance to complete the assignment, is there anything you would like to say about the assignment? For example, are there any resources that could help you complete this assignment? Feel free to use the feedback rubric above when reviewing this assignment.

my advice was send a list of past ideas but that already happened