**2.1 Task 1 : Design pattern and its Use Case Scenario Description. (4 points)**

Provide a description of the application use case scenario that you have selected for your final project. Describe what are your main software design concepts regarding this application. For example describe:

### What are the design goals in your project?

* + Flexibility
    - I will have 10 years of military service in June, 8 of those years has been with TUSAB. The one constant over that timeframe has been a continual change in requirements, especially regarding processing missions. Generally, these changes are minute, however, the way ServeMuse is designed is with generally broad "strokes" that capture concepts that have been around since before I was born: job sheets, transportation requests, etc. However, some ensembles and come and gone since I've been in the band and with each new 2-Star MDW general (they turn over every 3 years or so and are our direct "leadership") comes new requirements to mission processing. Most recently, we have to distinguish between the way we process missions that fall outside the National Capitol Region to those from within. This was not addressed directly by ServeMuse in this implementation but could easily be handled through the evaluation of the location field in a Request objects Event attribute.
  + Reusability
    - I feel the design is abstract and broad enough to be easily adapted to
      * Other military bands
      * Non-musical military units
      * Civilian music contractors
      * Potentially any corporate task delegation system
  + Sufficiency
    - It effectively accomplishes the following requirements:
      * Creating a Request
      * Submitting a Request
      * Delegating a Request
      * Generating mission details from a Request
      * Distributing mission details
      * Updating the status of a Request
      * Distributing the updated status in the deliverables containing mission details informing subscribing parties of the update
  + Understandability
    - See below for more information
  + Efficiency
    - As noted above this system essentially automates a series of processes (listed under *Sufficiency* above) that are currently manual processes
    - Consequently, this system would improve efficiency by many orders of magnitude by automating these processes
  + High Cohesion/Low Coupling
    - By focusing on the overlapping design patters noted in the UML Class Diagram High Cohesion and Low Coupling have been achieved to the extent it was possible
  + Reliability
    - More testing is needed however, seems to not fail when used properly at an acceptable rate.
  + Information Hiding
    - By adhering to the design patterns utilized, modularization is achieved and information hiding is achieved
  + Areas to Improve
    - Security
      * Need to build out defense against bad actors
    - Robustness
      * Although it is designed to be flexible generally, it is additionally designed to be interacted with in a certain way
        + This could be improved by developing out the capacity to handle different types of inputs

Essentially the only input to the system is a Google Calendar Event by design.

An easy way to build out Robustness would be to create a system agnostic Calendar API with the ability to interact with any Calendar API

### How is the flexibility, of your implementation, e.g., how you add or remove in future new types?

* + The Abstract Factory Pattern implemented is essentially designed to be able to support easy extension of the product and creator abstract classes.
  + The Observer Pattern implemented easily supports adding Subscriber base classes
    - In our case here we just need to be mindful that these new Subscriber base classes are folded into our Abstract Factory creator and producer classes
  + The Delegation Pattern implemented also easily supports implementation of more concrete delegators as needed.
  + Lastly, here the State Pattern implementation easily supports through implementing the Status Interface as many states as needed if more are needed.

### How is the simplicity and understandability of your implementation?

* + Understandability
    - It would be quite easy to explain and relate what the system is doing to a non-technical supervisor
      * The system reflects quite well "the way things are" at the office, how this process works in the real world
        + It is also clear how the automation of these processes improves the efficiency by many orders of magnitud

### How you avoided duplicated code?

* + By adhering to the above-mentioned design patterns quite strictly.
    - The producers package is an example where only Singletons are utilized and on the surface there may appear to be code duplicated
    - I have attempted to show that this is not the case in the BluesProducer, ChorusProducer, and DownrangeProducer classes findJobInCalendar methods all have deceivingly similar signatures however these methods are implemented with slightly varying algorithms to justify show why they are needed as distinct classes.

**2.2 Task 2 – UML Class Diagram. (5 Points)**

. See PDF CS665 Final Project Class Diagram or online [here](https://lucid.app/lucidchart/aac2db52-9d4f-4308-85e6-7340fab17501/edit?page=0_0&invitationId=inv_a8ae3f3c-9761-49ea-bb37-28c7c8c46bbc%23) to scale up

Diagram, map

Description automatically generated

**2.2.1 Task 2.1 – UML State Transition Diagram.**

2. requests package Status State Transitions also available [here](https://lucid.app/lucidchart/06d2539f-b7ba-4178-85ea-703f0d3c141a/edit?beaconFlowId=4A70AF925967D6B6&invitationId=inv_50785b53-ae29-4d45-8a5a-261450f6b659&page=0_0%23)

Diagram

Description automatically generated

**2.3 Task 3 – Implement your solution in Java. (5 Points)**

<https://github.com/metcs/met-cs665-assignment-project-MichaelKramerGuitar>

**2.4 Task 4 – Create a presentation about your project (5 Points)**

See ***mgkramer\_cs665classproject\_ServeMuse.pptx*** in project root folder

**2.5 Task 5 – Record your presentation and upload a MP4 file. (5 Points)**

See upload to blackboard.