Task Management System:
Design Summary

Mikaella Ziade

Marianne Moukashar

Adam Fawzi

Design Summary**Object Oriented Structure**The program is designed using object-oriented programming. Key components:

1. **Classes:**

• A general task takes the role of a base class.

• Used subclasses for specific classes (e.g. One-on-one Meeting and General Meeting are subclasses of the Event Class) 2. **Encapsulation:**

• Every class contains private variables with public getter and setter methods.3. **Inheritance:**• Enables general use of any task while allowing customization for specific cases.4. **Polymorphism:**• The displayinformation() method is overridden in derived class for displaying specific task information.

Data Management

- Tasks are stored in a dynamic vector that enables one to add, remove, and loop through the list as desired. [SEP]

- The existence of two different vectors allows for the implementation of undo and task reversal operations. [SEP]**User Interactions** [SEP]

- The main menu interface uses numbered options for easy access. [SEP]

- Input validation confirms user input correct data (eg. Dates, times, titles that are not [SEP]blank). [SEP]**Flexibility and Extensibility** [SEP]

- New Tasks are easily added by extending the Task class with minimal changes. [SEP]

- Features such as reminders and conflict checking are in the pipeline for future [SEP]implementation. [SEP]**System Architecture** [SEP]

- Core Classes: Task, Time, Location, Event, Meeting. [SEP]

- Workflow: [SEP]

1. Data entered by users to formulate, update, or query tasks. [SEP]

2. Tasks operate on location and time parameters to arrange or select outcomes. [SEP]

Lessons Learned

- Encapsulation is necessary to trace different responsibilities and tasks in programming.
- Subclass and inheritance complicate maintenance; composition may be considered in future iterations.