COM S 362: Object-Oriented Analysis and Design
Midterm Practice Exam
Spring 2022

Cover Sheet

Student Name: _____

Format:

- Time: 75 mins
- Points: 100
- Question Types: matching, true/false, short answer and diagram

Instructions:

- You may use 1 (one) letter sized sheet of paper (front and back), that you have prepared yourself with notes before the exam, as a "cheat sheet" during the exam.
- You may not consult classmates, electronic devices or resources other than the cheat sheet during the exam.
- Questions of clarification should be asked directly to an instructor or TA.

Question	**Points**
1	/24
2	/26
3	/10
4	/4
5	/4
6	/4
7	/8
8	/10
9	/10
Total	/100

1. (24 pts, 3 pts each) For each description on the left, select the best matching term on the right, each term is used only once but some will not be used.

____ an entity which initiates a particular use case

____ a code smell

____ a language independent graphical modeling language

____ a technique for identifying candidate domain concepts

____ the external binding force acting between modules which makes it difficult to change one module independently of another

____ indicates the number of instances participating in an association

____ indicated by a line between two classes

____ type of diagram describing interactivity between objects

a. UML
b. domain model
c. coupling
d. actor
e. use case
f. association
g. noun-phrase analysis
h. creator
i. long method
j. scenario
k. multiplicity
l. communication

2. (26 pts, 2 pts each) Which of the following statements are true? Write T or F for true or false.

____ During refactoring new features are added to the code.

____ Encapsulation is used to hide the secrets of abstractions.

____ UP and Scrum are incompatible methodologies.

____ Module boundaries are a good place to apply the SOLID principle of dependency inversion.

____ The code smell primitive obsession is often the sign of one of more missing abstractions.

____ Fred Brooks, the author of "No Silver Bullet – Essence and Accident in Software Engineering", was alarmed by the growing amount of accidental complexity in software.

____ The agile manifesto lays out a framework for organizing team roles and meetings.

____ Communication and use case diagrams are similar in that they can both show message passing.

____ A sequence diagram shows communication between classes and interfaces.

____ Following the principle of Separation of Concerns leads to low cohesion.

____ The code smell "large class" is often a sign that the principle of Separation of Concerns has been violated.

____ Both communication diagrams and CRC cards are useful when simulating scenarios.

____ A domain model shows how software classes are related to each other.

Answer the following questions using tear out page A.

3. (10 pts, 2 pts each) Which of the following statements are true? Write T or F for true or false.

____ Board is a subclass of Game.

____ The likely intention of the design is that WindowDecorator contains an instance variable of type Window.

____ The arrows between Controller, RepositionCmd and Window demonstrate the dependency inversion principle.

____ The lifecycle of Board is intended to be controlled by Game.

____ The likely intention of the design is that Game has a dependency on Controller.

4. (4 pts) The use of Controller in the design can best be described as an example of ____ (select one).

a. a violation of encapsulation
b. an example of high coupling
c. an example of low cohesion
d. a violation of separation of concerns
e. a violation of modularity

5. (4 pts) Suppose we want to assign Controller the creator responsibility for a SimpleWindow object. It does not keep a reference to the new object, it only passes it to Game. The best way to represent the change between Game and SimpleWindow is with ____ (select one).

a. generalization
b. association
c. aggregation
d. dependency
e. composition

6. (4 pts) The relationship between RepositionCmd, Window and SimpleWindow can best be described as ____ (select one).

a. high coupling
b. creator
c. dependency inversion
d. Liskov substitution
e. low cohesion

7. (8 pts, 4 pts each) Provide 2 to 4 sentence answers to each of the following.

a) Evaluate HorizontalScrollBar and VerticalScrollBar with respect to Liskov Substitution Principle.

b) Evaluate the design with respect to Interface Segregation Principle.

8. (10 pts, 5 pts each)

a) With respect to the Open-Closed Principle, describe any negative consequences of the design of the group: Controller, MinimizeCmd, ResizeCmd and RepositionCmd. (3 to 5 sentences)

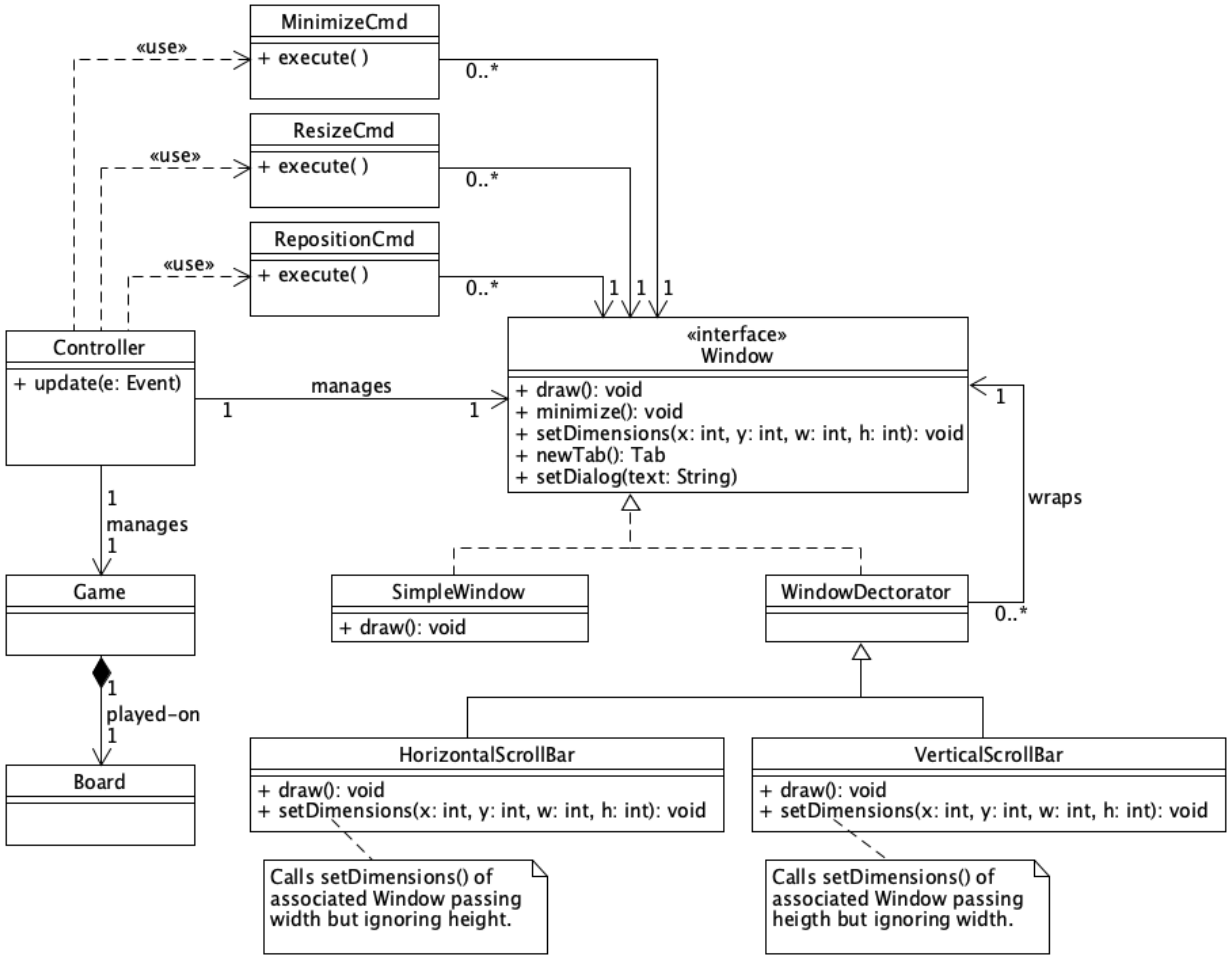
b) Draw a class diagram the describes a refactoring of the design to achieve better open-closed principle. Don't redraw the entire class diagram, only include parts of the diagram that are required to make the design improvement clear.

9. (10 pts, 5 pts each) Use the code shown on tear out page B to complete the following problems.

a) Create a communication diagram using the call to startServer() as the first message.

b) Create a sequence diagram using the call to startServer() as the initiating found message.

Tear out page A, remove this page to use as a reference.



Tear out page B, remove this page to use as a reference.

```
interface Logger {
    // ...
    public void log(Level level, String msg);
    public void close();
}

class FileLogger implements Logger {
    // ...
    public FileLogger(FileWriter fw, Level minLevel, Level maxLevel) {
        // ...
    }
    public void log(Level level, String msg) {
        // ...
    }
    public void close() {
        // ...
    }
}

class MultiLogger implements Logger {
    List<Logger> loggers = new LinkedList<Logger>();

    public void addLogger(Logger logger) {
        loggers.add(logger);
    }
    public void log(Level level, String msg) {
        for (Logger logger: loggers) {
            logger.log(level, msg);
        }
    }
    public void close() {
        for (Logger logger: loggers) {
            logger.close();
        }
    }
}

public class LoggingServer {
    public static Logger logger;

    public void initLogger() {
        // ...
        MultiLogger ml = new MultiLogger();
        ml.addLogger(new FileLogger(logFw, Logger.CONFIG, Logger.WARNING));
        ml.addLogger(new FileLogger(errorFw, Logger.SEVERE, Logger.SEVERE));
        logger = ml;
    }
    public void startServer() {
        logger.log(Logger.INFO, "server started");
    }
    public static void main(String[] args) {
        LoggingServer server = new LoggingServer();
        server.initLogger();
        server.startServer();
        logger.close();
    }
}
```