1. GRASP (ch. 17 AND 25)
   a. Create a sequence diagram for a specific main scenario and mark all the GRASP responsibilities fulfilled by methods in the SD.
   b. Design methods using GRASP responsibilities: creator, expert, controller, polymorphism, indirection, etc.
   c. Use GRASP principles of polymorphism, indirection, pure fabrication, low coupling, and high cohesion to improve the design in a sequence diagram.
   d. Does the given code follow the Liskov Substitution Principle (LSP)? Why? Why not?
   e. Rewrite the code to apply LSP
   f. What is the difference between abstraction and encapsulation in Java?
   g. What is the difference between information hiding and encapsulation in Java?
   h. Explain which GRASP principles are implemented in the following:
      i. Code
      ii. Sequence diagram
      iii. Class diagram

2. Design for visibility
   a. What are the differences between different types of visibility?
   b. Why do we have different types of visibility?
   c. Recognize the different types of visibility in a code snippet.

3. Mapping designs to code
   a. Write the Java code that corresponds to the given class diagram.
   b. Write the Java code that corresponds to the given sequence diagram.
   c. Write the Java code that corresponds to the given collaboration diagram.

4. Test driven development
   a. What are the different types of testing?
   b. Why use multiple types of testing methods?
   c. What test cases would you write for the following code?
   d. How would you test the input of the following code to improve its security?
   e. Which are the three test cases that you should consider during unit testing?

5. Gang of Four (GoF) design patterns
   a. What makes a pattern?
   b. Create a sequence / collaboration diagram and apply GoF patterns to all appropriate methods that identify object responsibilities.
   c. Which GoF pattern is depicted in the class diagram below?
   d. How is GRASP connected to GoF?
   e. When should you use adapter, factory, singleton, strategy, composite, façade, or observer pattern in your Java software?
f. Why should we use design patterns?
g. What is the MVC pattern? What are its advantages and disadvantages?
h. Why should we use layered architecture model? What are the advantages? What are the disadvantages?
i. In what scenarios we should not use design patterns?
j. What is the importance of a design pattern?
k. What is the difference between the factory and singleton pattern?

6. Activity Diagrams
   a. Draw the activity diagram for a given scenario.
   b. When do we draw activity diagrams?
   c. Why do we need activity diagrams?
   d. What is the difference between activity diagrams and interaction diagrams?
   e. Explain the fork/join notation in an activity diagram.

7. State Machine Diagrams
   a. What is the difference between activity and state diagrams?
   b. When and why do we use state diagrams?
   c. Draw a state diagram for a given scenario.

8. Secure Software Development
   a. Find the parts in the code that may cause buffer overflow.
   b. List the top ten OWASP security weaknesses that you will need to address when you design X software package (where X may be web application server, video game, health monitor, etc.)
   c. How should you handle passwords with your software and why?
   d. How should you handle sensitive personal data in your software and why?
   e. Explain the secure software lifecycle.
   f. How can encapsulation and information hiding be used for secure software development?
   g. How can exception handling help in secure software development?