FRANKLIN UNIVERSITY PROFICIENCY EXAM (FUPE) STUDY GUIDE

Course Title: Object-Oriented Data Structures and Algorithms 1 (COMP 121)


Number & Type of Questions: Approximately 50 questions. The test contains a variety of types of questions: multiple choice, true/false, predict the output of given code, write code segments to accomplish specific tasks, fill in the blank, trace an algorithm, short answer.

Permitted Materials: Pencil or pen, calculator

Time Limit: 4 hours

Minimum Passing Score: 80%

Knowledge & Skills Required:

The COMP 121 FUPE addresses the following weekly course outcomes:

Week 1
1. Review the fundamental object-oriented principles of encapsulation, composition, and abstraction.
2. Review the algorithmic building blocks of methods, selection, and repetition.
3. Review the properties, operations, and use of primitive data types, strings, and arrays.
4. Use an integrated development environment to edit, compile, run, submit, and correct a Java program.

Week 2
5. Explain how interfaces reduce coupling while increasing code reuse.
6. Define polymorphism and late binding.
7. Write generic algorithms that use polymorphism to act on covariant data types.
8. Explain what design patterns are and how they are used.
9. Recognize and apply the Strategy design pattern to solve a given problem.
Week 3
10. Explain the purpose, uses, and scope of inner classes.
11. Describe inheritance and the Java mechanisms that implement it.
12. Distinguish between overloaded and overridden methods.
13. Organize a set of related classes into an inheritance hierarchy.
14. Determine when and how to call superclass methods from within a subclass.
15. Apply the rules of implicit and explicit object conversion.

Week 4
16. Distinguish between abstract and concrete classes and methods.
17. Describe the uses and effects of the keywords final, public, protected, and private as they apply to classes, methods, and instance fields.
18. Supply appropriate equals, toString, and clone methods for base, derived, and composed classes.
19. Apply appropriate access modifiers within class design.
20. Recognize and apply the Template Method design pattern to solve a given problem.

Week 5
21. Read and write text files.
22. Explain the purpose and use of exception handling for error detection and correction.
23. Differentiate between checked and unchecked exceptions.
24. Use the keywords throws, try, throw, catch, and finally to implement exception handling.
25. Define and use a domain-specific exception hierarchy.

Week 6
26. Read and write binary files.
27. Read and write serialized object files.
28. Distinguish between random and sequential access files.

Week 7
29. Compare and contrast iterative versus sequential software lifecycle approaches.
30. Use CRC cards to capture the responsibilities of each class and the relationships between classes.
31. Use UML class diagrams to illustrate relationships between classes.
32. Use primitive operation counts and deductive reasoning to determine the efficiency of algorithms.
33. Given a set of initial conditions, predict how the runtime of an algorithm is affected by increased input size.

Week 8
34. Describe how generics increase code reuse and type-safety.
35. List the common operations and properties of all collections.
36. Implement the Collection interface in an AbstractCollection and ArrayCollection and ArrayList implementations and justify design decisions.
37. Recognize and apply the Iterator design pattern to solve a given problem.

Week 9
38. List the common operations and properties of all lists as distinct from collections.
39. Extend the AbstractCollection implementation into AbstractList and ArrayList implementations and justify design decisions.
40. Analyze the ArrayList implementation to determine algorithmic efficiency.
41. Use an ArrayList data structure to solve a problem.

Week 10
42. Extend the AbstractCollection implementation into a LinkedCollection and justify design decisions.
43. Apply object-oriented design principles to maximize code-reuse in LinkedCollection.
44. Compare and contrast array-based versus linked collections.

Week 11
45. Extend the AbstractList implementation into a LinkedList implementation and justify design decisions.
46. Analyze the LinkedList implementation to determine algorithmic efficiency.
47. Compare and contrast array-based lists versus linked lists.
48. Use a LinkedList data structure to solve a problem.

Week 12
49. Recognize and apply the Decorator design pattern to solve a given problem.
50. Recognize and apply the Adapter design pattern to solve a given problem.

Week 13
51. List the typical operations and properties of stacks as distinct from other collections.
52. Implement operations on stacks and justify design decisions.
53. Analyze the stack implementation to determine algorithmic efficiency.
54. Use a stack data structure to solve a problem.

Week 14
55. List the typical operations and properties of queues as distinct from other collections.
56. Implement operations on queues and justify design decisions.
57. Analyze the queue implementation to determine algorithmic efficiency.
58. Use a queue data structure to solve a problem.