FRANKLIN UNIVERSITY PROFICIENCY EXAM (FUPE) STUDY GUIDE

Course Title:	***Introduction to CS and OOP (COMP 111)***
Recommended Textbook(s):	*Big Java*, 4th Edition, Cay Horstmann, Wiley (ISBN: 0-471-69703-6)

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 111 FUPE addresses the following weekly course outcomes:

Week 1

1. Use an Integrated Development Environment (IDE) to edit, compile, and run a Java program.
2. Employ the feedback from an automatic grading system to correct program deficiencies.
3. Distinguish between and correct syntax and logic errors.
4. Describe the inputs, activities, and outputs of each step in the compilation process.

Week 2

5. Distinguish between a data type, an identifier, and a variable.
6. Distinguish between objects, classes, and methods.
7. Distinguish between inputs and outputs of methods.
8. Identify appropriate number types for various kinds of data.
9. Use objects, write test cases, and identify and correct program errors.

Week 3

10. Instantiate and use an object from a class.
11. Examine how accessor and mutator methods affect objects.
12. Create and execute unit tests for pre-written objects.
13. Use API documentation to explore object interfaces.
14. Explain the differences between a reference and the object it references.

Week 4

15. Define encapsulation and abstraction in the context of object-oriented programming.
16. Apply the principles of encapsulation and abstraction to define the public interface of a class.
17. Implement simple classes and methods.
18. Create API documentation for object interfaces.
19. Employ a testing framework to validate implemented classes.

Week 5

20. Distinguish between instance fields, local variables, and parameter variables in terms of their use, scope and lifetime.
21. Distinguish between implicit and explicit parameters.
22. List and describe primitive data types in Java.
23. Utilize modifiers **static** and **final** to define constants.
24. Apply operators and Java library methods to compute arithmetic results.
25. Implement classes and methods, write test cases, use **String** objects, and perform simple I/O.

Week 6

26. List and use common **String** operations and methods.
27. Use the **Scanner** class for formatted input and the **Sytem.out.printf** method for formatted output.

Week 7

28. Use **if/else** statements to implement decisions.
29. Apply relational operators to compare numeric data.
30. Apply methods to compare strings and objects.
31. Write nested **if/else** statements to implement complex logic.
32. Combine logical expressions using Boolean operators.
33. Predict the output of code snippets that contain conditionals.
34. Use Boolean expressions and operators, selection control structures, and repetition control structures.
35. Select necessary and sufficient test cases.
36. Explain black- and white-box testing, regression testing, and test coverage.

Week 8

37. List and describe the four components of all loops.
38. Use **for, while,** and **do** loops to solve simple problems.
39. Identify and correct common loop errors such as off-by-one errors, infinite loops, and non-executing loops.
40. Predict the output of code snippets that contain loops.
41. Implement algorithms requiring nested loops.
42. Differentiate between various loop termination conditions such as sentinels, symmetric and asymmetric bounds, and counting.

Week 9

43. Use the basic methods of the **Random** class to generate random numbers.
44. Use a debugger to examine a running program.
45. Describe the properties and uses of arrays and **ArrayList** data structures.
46. Use arrays and **ArrayLists** and advanced loop processing.

Week 10

47. Compare and contrast the properties and uses of primitive data types versus wrapper classes.
48. Apply the generalized **for** loop to simple array algorithms.
49. Declare, instantiate, initialize, and use multi-dimensional arrays.
50. Write code snippets to copy, insert, and delete elements of arrays.

Week 11

51. Identify and defend good candidates for classes and methods.
52. Design classes to reduce coupling, increase cohesion, and minimize side effects.
53. Apply advanced array processing algorithms, good class design principles, and problem solving techniques.

Week 12

54. Compare, contrast, and demonstrate the execution of algorithms for selection, insertion, and bubble sorts.
55. Analyze the performance of selection, insertion, and bubble sorts using profile data and primitive operation counts.
56. Compare, contrast, and demonstrate the execution of the algorithms for linear and binary search.
57. Analyze the performance of linear and binary search using profile data and primitive operation counts.

Week 13

58. Specify the pre- and post-conditions for a method.
59. Differentiate between the uses of **static** and **non-static** fields and methods in a class.
60. Determine the scope of a variable.
61. Use packages to group related classes.

Week 14

62. Apply previous learning outcomes to design a problem solution involving multiple dependent classes.
63. Create an object-oriented design of several classes, complete with UML diagram, attributes (description of instance data), and behaviors (methods).