



Java Programming 2

Course Summary:

The student will learn essential object-oriented programming concepts, exception handling, recursion, generics, and important data structures in the Java Collections Framework. Advanced topics include algorithm analysis using Big *O* notation, a comparison of major sorting algorithms, and creation and traversal of a binary search tree. Lessons are accompanied by frequent programming exercises.

Course Outline

- 1. Object-Oriented Thinking: Part 1 (L)**
 - Understand class abstraction and encapsulation
 - Explain the object-oriented paradigm
 - Identify class relationships
 - Translate a UML class diagram into Java code
- 2. Object-Oriented Thinking: Part 2 (L)**
 - List the Java wrapper classes associated with each primitive data type
 - Demonstrate the use of common methods and fields in numeric wrapper classes
 - Explain autoboxing and autounboxing
 - Perform calculations on very large numbers using BigInteger
- 3. Inheritance and Polymorphism: Part 1 (L)**
 - Understand inheritance
 - Use the super keyword to call a superclass constructor or method
 - Override methods inherited from a superclass
 - Explain overriding and overloading
- 4. Inheritance and Polymorphism: Part 2 (L)**
 - Explain polymorphism, dynamic binding, declared types, and actual types
 - Describe the difference between the comparison operator and the equals() method
 - Store and retrieve data from an ArrayList
 - Understand the protected modifier
 - Explain the effect of the final keyword on classes and methods
- 5. Exception Handling and Text I/O: Part 1 (L)**
 - Understand the basics of exceptions and exception handling
 - Identify different types of exceptions
 - Declare, throw, and catch exceptions
 - Explain the use of the finally clause
 - Decide when to use exceptions
- 6. Exception Handling and Text I/O: Part 2 (L)**
 - Demonstrate how to rethrow an exception
 - Understand the File class
 - Handle basic keyboard input and file I/O
- 7. Abstract Classes and Interfaces: Part 1 (L)**
 - Explain the need for abstract classes and methods

- Apply the rules for abstract classes and methods
- Identify abstract classes and methods in the Java API
- 8. Abstract Classes and Interfaces: Part 2 (L)**
 - Discuss how interfaces are used
 - Demonstrate the implementation of the Comparable interface
 - Discuss the differences between interfaces and abstract classes
 - Explain cohesion, consistency, encapsulation, clarity, and completeness
 - Contrast instance vs. static and inheritance vs. aggregation
- 9. Recursion (L)**
 - Understand the basics of recursion
 - Write recursive methods
 - Explain the need for recursive helper methods
 - Apply recursion to a file system
 - Contrast recursion and iteration
- 10. Generics (L)**
 - Explain the syntax and use of generic types
 - Define a generic class
 - Understand generic methods and bounded types
 - Understand the syntax and implications of wildcard generic types
- 11. Lists, Stacks, Queues, and Priority Queues: Part 1 (L)**
 - Contrast important data structures in the Java Collections Framework
 - Describe methods of the Collection interface
 - Traverse a data structure using an Iterator and a foreach loop
 - Store and retrieve information using a List
- 12. Lists, Stacks, Queues, and Priority Queues: Part 2 (L)**
 - Define a class that implements the Comparator interface
 - Use static methods of the Collections class to manipulate data in a list
 - Explain the operation of stacks and queues
- 13. Sets and Maps (L)**
 - Describe the differences between HashSet, LinkedHashSet, and TreeSet
 - Add and retrieve data from a set
 - Choose the best data structure based on a program's requirements
- 14. Developing Efficient Algorithms (L)**
 - Explain Big O notation
 - Calculate the time complexity for an algorithm or block of code
 - Compare and order functions of time complexity
- 15. Sorting (L)**
 - Describe the strategy behind common sorting algorithms
 - Determine the time complexity of common sorting algorithms
- 16. Binary Search Trees (L)**
 - Describe the structure of a binary search tree
 - Implement a binary search tree
 - Traverse a binary search tree using recursion
- 17. Java Programming II Course Review (L)**
 - Review lesson objectives and key terms
 - Review textbook readings
 - Utilize pretests and pretest summaries
 - Explore potential careers in the field of Java programming
- 18. Java Programming II Final Assignment (L)**