

Fundamentals of Java Programming

2/21/2002

Target Audience:

Academy Students

Prerequisites:

Participants should have a Reading Age Level (RAL) of 13, basic computer literacy, including the use of application software such as word-processors, spreadsheets and or databases and understanding of the Internet. Prior familiarity with elementary programming concepts such as storing of data in variables and control logic is desirable but not required.

Course Description:

The Fundamentals of Java Programming Language course provides a conceptual understanding of Object Oriented programming. The course also teaches students how to use the JAVA™ language's object oriented technologies to solve business problems. Students will learn how to create classes, objects, and applications using the language. Topics also include the language fundamentals, the Java language API (application programming interface). Additionally, the course will address the demand for training and preparation for the Sun Certified Programmer for Java [tm] 2 Platform.

Course Objectives:

Upon completion of this course, students will:

- Understand the history and current use of Objected oriented programming and the Java language to solve business problems.
- Understand concepts of Object, class, instance, member data/fields, member attributes/methods, and local variables.
- Understand the Java Environment and use of the Java Development kit for the creation and execution of java programs from java source files.
- Apply java language keywords, and syntax to create statements for declaring and storing java data types.
- Understand the result of operations and decision-making on java data types, using any operator or method.
- Create statements for data operations, decision-making, class definition, object constructors, method definitions and method invocation.
- Use the online documentation for the Java 2 Platform, API Specification.
- Implement the Javadoc features in their source files to document their classes.
- Write code that implements OO principles and design patterns of encapsulation, composition and inheritance.
- Write code to invoke overridden or overloaded methods and parental or overloaded constructors, and be able to describe the effect of invoking these methods.
- Design and construct instances of any concrete class including normal top-level classes, inner classes, static inner classes, and anonymous inner classes.
- Write code to define, instantiate classes of the java.lang, java.util, java.awt, java.io packages.

Lab Requirements:

Software—Download the BlueJ java development tool. BlueJ is an integrated Java environment specifically designed for the teaching of java programming. BlueJ is a free download available at <http://www.bluej.org>. The current version of BlueJ is 1.5.1. It requires Java 2 SDK v 1.3 or 1.3.1 available at <http://java.sun.com/products/archive/j2se/1.3.1/index.html>.

Certification Alignment:

The Fundamentals of Java Language Programming Course will be aligned with and prepare a student for the Sun Certified Programmer for Java 2 Platform. The alignment matrix is available in a separate document.

Course Overview

Course Outline

What is Java?

- Computer basics
- What is a programming language
- What is Java
- The Java Environment Basics
- A Java program
- Entering data at runtime for the java program
- Understanding programming error messages
- Understanding Java Development environments – IDEs
- Case Study : Banking Application

Object Oriented Programming

- Terminology used in object oriented programming
- What are objects?
- Describing Objects
- Object terminology applied to a Java Program
- Introduction to java.lang.System class
- Case Study: Banking Application

Java language elements

- Documentation
- Java language elements
- Where and when of data storage
- Concept of data types
- Syntax
- Object creation, mutability and destruction
- Case Study: Banking application

Java language operators and control structures

- Object operations
- Numeric data and operations
- Concepts of Casting and conversion
- Character and String data
- Control structures - Decision Making and repetition
- Exploring the java.lang.System class member in and out
- Dissecting sample code
- Lab activities

Basics of Defining and using classes

- A review of class object definitions
- Four steps to creating objects
- Encapsulation
- Attributes (class and instance)
- Constructors
- Method types and syntax
- The variable this
- Data available in a method
- Implementing encapsulation using methods

- Overloading
- Java type lifespan – creation, mutability and destruction
- Case Study: Banking Application

System, Strings , String Buffer, Math and Wrapper classes

- System class
- String class
- StringBuffer class
- Input selection and repetition using System, String and String buffer classes
- Wrapper classes
- Math class
- Math package
- Working with dates
- Case Study: Banking Application

Midterm test

Classes and Inheritance

- Inheritance and Object Oriented Programming
- Java language support for inheritance
- Access modifiers and inheritance
- Overriding
- Use of this and super
- Inheritance and constructors
- Extending classes
- Interfaces
- Polymorphism and dynamic binding
- Case Study: Banking Application

Arrays

- Arrays
- Declaring and initialize an Array object
- Initializing Arrays
- Using Arrays
- Multidimensional arrays
- Case Study: Banking Application

Understanding packages

- Exploring the API packages
- Packages and names in your programs
- Accessing packages
- Packaging your classes
- Case Study: Banking Application
- **Using the AWT**
- Case Study: Banking Application

Creating GUI applications using AWT

- Reviewing the AWT
- Steps to create a GUI
- GUI functionality
- Case Study: Banking Application

Creating an applets and graphics

- Applets
- Creating an applet
- Applet and AWT class heirarchies
- Applet methods
- Applet Components
- Applets and Event-driven programming
- Case Study: Banking Application
- **Graphic objects**
- Creating Graphic objects
- Graphics methods
- Create simple animation

Exceptions

- Idea behind exceptions
- Types of exceptions
- How exception handling works
- Exception objects
- Dealing with exceptions
- Defining and using your own Exception
- Structuring a method the execution sequence
- Overriding exceptions
- Case Study: Banking Application

Streams, files and Stream output

- Understanding files
- Input and Ouptu classes
- A how-to for input and output operations
- Storing Objects in a file
- Case Study: Banking Application

Utility classes

- The java.util package
- Collections
- The Collections framework
- Generating Random Numbers
- Case Study: Banking Application

Threads

- Threads and Multithreading
- The Thread class and Runnable interface
- The lifecycle of a thread
- Creating and running threads
- Managing threads
- Lab exercise

Final exam

Fundamentals of Java Language Programming Course Outline

1 What is Java?

- 1.1 Computer basics
 - 1.1.1 Computers
 - 1.1.2 Software
 - 1.1.3 Applications
 - 1.1.4 Operating Systems
- 1.2 What is a programming language?
 - 1.2.1 What is a programming language
 - 1.2.2 Evolution of programming language
 - 1.2.3 Basic language elements
 - 1.2.4 Differences between languages
- 1.3 What is Java?
 - 1.3.1 Historical background of Java
 - 1.3.2 Use of Java as a programming language
- 1.4 The Java Environment Basics
 - 1.4.1 Overview
 - 1.4.2 Understanding the Java Virtual Machine
 - 1.4.3 How Java Programs work on your computer
- 1.5 Your First Java Program
 - 1.5.1 Organizing resources to create a Java program
Lab Activity – Locating resources, managing editors and console windows
 - 1.5.2 Understanding the Java 2 Software Development Kit (J2SDK)
 - 1.5.3 Elements of a simple Java program
 - 1.5.4 Three steps to create and execute a java program
Lab Activity – Creating and running the HelloStudent1 program
- 1.6 Entering Data at Runtime for the Java Program
 - 1.6.1 Modifying the Hello Student program to accept input at Runtime
Lab Activity – Modifying HelloStudent1 program to accept input from user at Runtime
- 1.7 Understanding programming error messages
 - 1.7.1 Types of errors in programs
 - 1.7.2 Editing a java program for compiler errors
Lab Activity – Debugging and correcting errors in pre-defined classes
 - 1.7.3 Editing a java program for runtime errors
- 1.8 Understanding Java Development environments – IDEs
 - 1.8.1 What is an IDE?
 - 1.8.2 Basic editing with the BlueJ IDE
Lab Activity – Creating HelloStudent3 in BlueJ
 - 1.8.3 The BlueJ tutorial
Lab Activity – Exploring Bluej tutorial
 - 1.8.4 CaseStudy: JBANK a banking application
Lab Activity – Create and run the Teller pogram
Lab Activity – Modifying the Teller program to accept runtime data

2 Object Oriented Programming

- 2.1 Terminology used in object oriented programming
 - 2.1.1 Object oriented introduction
 - 2.1.2 Object speak
- 2.2 What are objects?
 - 2.2.1 Introduction to objects
 - 2.2.2 A Programmers Point of view
 - 2.2.3 Identifying objects
 - 2.2.4 Defining classes of objects
 - 2.2.5 Creating objects – Object construction
 - 2.2.6 Operating on objects
Lab Activity - Teacher class calls Student class

- 2.2.7 Encapsulation
- 2.2.8 Object relationships
- 2.2.9 Inheritance
- 2.2.10 Object mutability and destruction
- 2.3 Describing Objects
 - 2.3.1 Modeling languages and symbols
 - 2.3.2 Basic Class Symbol
- 2.4 Object terminology applied to a Java Program
 - 2.4.1 Class definition
 - 2.4.2 Creation of objects – object data
 - 2.4.3 Operating on Objects – object methods
- 2.5 Introduction to java.lang.System class
 - 2.5.1 SayHello source code
- 2.6 Case Study: Banking Application
 - 2.6.1 Defining and modeling the JBANK application
 - Lab Activity - Designing and describing classes using UML
 - Lab Activity – Develop the Banking classes for Phase I

3 Java language elements

- 3.1 Documentation
 - 3.1.1 Why and for whom – class uses
 - 3.1.2 Guidelines for documenting classes
 - 3.1.3 The javadoc parameters
 - 3.1.4 Java language API online documentation
 - 3.1.5 Generating API docs for your classes using the javadoc tool
 - Lab Activity – Reviewing the API documentation
 - 3.1.6 Case Study: JBANK banking Application
 - Lab Activity – Inserting documentation for classes in the JBANK banking application
 - Lab Activity – Generating API docs for JBANK classes using the javadoc tool
- 3.2 Java language elements
 - 3.2.1 Elements Introduction
 - 3.2.2 Keywords
 - 3.2.3 Identifiers
 - 3.2.4 Use of braces, semicolons and commas and white space
- 3.3 Where and when of data storage
 - 3.3.1 Data storage introduction
 - 3.3.2 Stack, Heap, Static and Constant Storage
 - 3.3.3 Variables
- 3.4 Concept of data types
 - 3.4.1 Java Language types
 - 3.4.2 Java primitives and storage requirements
 - 3.4.3 Java references
 - 3.4.4 Object data (instance)
 - 3.4.5 Class data (static)
 - 3.4.6 Method data (local)
 - 3.4.7 Constants
- 3.5 Syntax
 - 3.5.1 Variables
 - Lab Activity – Defining Variables
 - 3.5.2 Class
 - 3.5.3 Method
 - 3.5.4 Constructor
 - 3.5.5 Modifiers (public, private, protected, default, static, final)
 - Lab Activity – Applying access modifiers
- 3.6 Object creation, mutability and destruction
 - 3.6.1 Object Creation - Constructors
 - Lab Activity – Using constructors
 - 3.6.2 Object Creation as it relates to strings
 - 3.6.3 Mutability

- 3.6.4 Garbage collection
- 3.6.5 Finalizers
- 3.7 Case Study: Banking application JBANK
 - 3.7.1 Case Study: JBANK banking application
 - Lab Activity – Creating the classes for Phase I of the banking application

4 Java language operators and control structures

- 4.1 Object Operations
 - 4.1.1 Object operators
- 4.2 Numeric data and operations
 - 4.2.1 Assignment operators
 - 4.2.2 Arithmetic operators
 - Lab Activity – Arithmetic operators
 - 4.2.3 Precedence of operators
 - 4.2.4 Associative properties of operators
 - 4.2.5 Arithmetic calculations
 - Lab Activity – Using +_ operator with numbers and text data
 - 4.2.6 Boolean Data
 - 4.2.7 Comparison and Logical Operators
 - 4.2.8 Conditional operator
 - 4.2.9 Bitwise operators
- 4.3 Concepts of casting and conversion
 - 4.3.1 Casting and conversion
- 4.4 Character and String data
 - 4.4.1 Character data type
 - 4.4.2 Introduction to String and String Buffer class
 - Lab Activity – String concatenation
- 4.5 Control structures –
 - 4.5.1 Decision Making and repetition
 - 4.5.2 Logic
 - 4.5.3 If statement
 - Lab Activity – If statement
 - 4.5.4 Multiple if conditions
 - 4.5.5 Nested if
 - 4.5.6 Switch statements
 - Lab Activity – Switch statement
 - 4.5.7 Loop
 - 4.5.8 Do while statement
 - Lab Activity – Do while statement
 - 4.5.9 While statement
 - Lab Activity – While statement
 - 4.5.10 For statement
 - Lab Activity – For Loop
 - 4.5.11 Use of break, continue and label
- 4.6 Exploring the java.lang.System class member in and out
 - 4.6.1 The java.lang.System class
 - Lab Activity – The java.lang.System class
 - Lab Activity – The Console class
- 4.7 Dissecting Sample Code
 - 4.7.1 Dissecting sample code
 - 4.7.2 Lab Activity – Leasing vehicles

5 Basics of Defining and using classes

- 5.1 A Review of Class and Object Definitions
 - 5.1.1 A review of class and object definitions
- 5.2 Four steps to creating objects
 - 5.2.1 Designing classes

- 5.2.2 Define a class
- 5.2.3 Create an object
- 5.2.4 Using the object
- Lab Activity – Four steps to creating objects
- 5.3 Attributes(class and instance)
 - 5.3.1 Attributes – class and instance
 - 5.3.2 Variables – class, instance, local variables
 - 5.3.3 Object data – instance variables
 - 5.3.4 Class data -- exploring static
 - 5.3.5 Immutability – use of final for constants.
 - Lab Activity – Creating objects, encapsulation concepts, and attributes
- 5.4 Encapsulation
 - 5.4.1 Concepts
 - 5.4.2 Access modifiers private, public, default, protected
- 5.5 Constructors
 - 5.5.1 Where and when of Object creation- the constructor process
 - 5.5.2 Default constructor
 - 5.5.3 Defined constructor
- 5.6 Method types and syntax
 - 5.6.1 Method signature
 - 5.6.2 Main method
 - 5.6.3 Instance methods
 - 5.6.4 Class methods
 - Lab Activity – Constructors and Methods
- 5.7 The Variable this
 - 5.7.1 Using this in constructors and methods
- 5.8 Method data and encapsulation
 - 5.8.1 Data sources
 - 5.8.2 Implementing encapsulation using methods
- 5.9 Overloading
 - 5.9.1 Overloading methods
 - 5.9.2 Overloading constructors
 - Lab Activity – Overloading methods and constructors
- 5.10 Java type lifespan – creation, mutability and destruction
 - 5.10.1 Initializing data – instance, class and local
 - 5.10.2 Scope of variables
 - 5.10.3 Lifetime of an object
 - Lab Activity – Scope of variables
 - 5.10.4 Mutability of an object
 - 5.10.5 Destruction of an object
 - 5.10.6 Finalizers
- 5.11 Case Study: Banking Application
 - 5.11.1 Applying concepts to the Banking application
 - Lab Activity – Completing the JBANK Phase1 application

6 System, Strings, String Buffer, Math and Wrapper classes

- 6.1 System class
 - 6.1.1 Use of the System class for input and output
 - 6.1.2 Input using System.in
 - Lab Activity – Input using System.in
 - 6.1.3 Output using System.out
- 6.2 String class
 - 6.2.1 Creating strings
 - 6.2.2 String methods
 - Lab Activity – String methods
 - 6.2.3 Casting and conversion
 - Lab Activity – Casting and conversion
- 6.3 StringBuffer class

- 6.3.1 Methods
- 6.4 Input selection and repetition using System, String and String buffer classes
 - 6.4.1 The Console class
- 6.5 Wrapper classes
 - 6.5.1 Introduction to wrapper classes
 - 6.5.2 Integer
 - 6.5.3 Double
 - 6.5.4 Character
 - 6.5.5 Boolean
- 6.6 Math class
 - 6.6.1 Static methods
 - Lab Activity – Using the Math class
- 6.7 Math package
 - 6.7.1 Storing numeric data in Objects
 - Lab Activity – Using the Math package
- 6.8 Working with dates and random numbers
 - 6.8.1 Creating dates
 - 6.8.2 Setting dates
 - 6.8.3 Formatting dates
 - Lab Activity – Working with dates
 - 6.8.4 Random Numbers
- 6.9 Case Study: Banking application JBANK
 - 6.9.1 Accepting and converting input for the JBANK
 - Lab Activity – System, String, StringBuffer and use of Console class
 - Lab Activity – Wrapper classes, Math class, Date class

7 Arrays

- 7.1 Arrays
 - 7.1.1 Introduction to arrays
 - 7.1.2 Arrays of primitives
 - 7.1.3 Arrays of objects
- 7.2 Declaring and initialize an Array object
 - 7.2.1 Declaring arrays
 - 7.2.2 Use subscripts to access elements of an array
- 7.3 Initializing Arrays
 - 7.3.1 Array of primitives
 - 7.3.2 Array of objects
 - 7.3.3 Using initializer blocks
- 7.4 Using arrays
 - 7.4.1 Accessing array elements
 - 7.4.2 Passing an Array to a method
 - Lab Activity – Passing an Array to a method
 - 7.4.3 Parallel arrays
 - Lab Activity – Creating and Traversing through arrays
 - 7.4.4 Searching and sorting an array
 - Lab Activity – Searching and sorting an array
 - 7.4.5 Reusing an array
 - Lab Activity – Extended use of Arrays
- 7.5 Multidimensional arrays
 - 7.5.1 Initialization of array objects
 - 7.5.2 Traversing a multidimensional array
 - Lab Activity – Traversing a multidimensional array
- 7.6 Case Study: Banking Application
 - 7.6.1 Implementing Arrays in the JBANK Application
 - Lab Activity – Implementing Arrays in the JBANK Application

8 Classes and Inheritance

- 8.1 Inheritance and Object Oriented Programming

- 8.1.1 Inheritance
- 8.1.2 Abstraction
- 8.1.3 The problem of Multiple inheritance
- 8.2 Java Language Support for Inheritance
 - 8.2.1 Java language keywords in inheritance
 - 8.2.2 Object Class
- 8.3 Access Modifiers and Inheritance
 - 8.3.1 Role of access modifiers in inheritance
- 8.4 Overriding
 - 8.4.1 Method overriding
 - 8.4.2 Overriding of Object class methods
 - 8.4.3 Overloading vs. Overriding
- 8.5 Use of this and Super
 - 8.5.1 Accessing parent and sub-class methods and data
- 8.6 Inheritance and Constructors
 - 8.6.1 Handling constructors in inheritance
- 8.7 Extending Classes
 - 8.7.1 Abstract Classes
 - Lab Activity – JBANK Banking application: Implement Abstraction in Phase II of the banking application
 - 8.7.2 Final Classes
 - Lab Activity – JBANK Banking application: Implementing inheritance, extending from abstract and concrete classes.
 - Lab Activity – JBANK Banking application: Abstraction at several levels – Checking Account
- 8.8 Interfaces
 - 8.8.1 What and why of interfaces
 - 8.8.2 Implementing Interfaces
 - Lab Activity – Animals
- 8.9 Polymorphism, Dynamic Binding and Virtual Method Invocation
 - 8.9.1 Polymorphism
 - 8.9.2 Virtual method invocation, or dynamic binding
- 8.10 Case Study: Banking Application
 - 8.10.1 Applying concepts to the Banking application
 - Lab Activity – Polymorphism in the Banking Application

9 Understanding packages

- 9.1 Java Packages
 - 9.1.1 A collection of classes
 - 9.1.2 Class loading and how it works
 - 9.1.3 Locating explicit package declarations
- 9.2 Packages and names in your programs
 - 9.2.1 Three effects of packages on object-oriented design
 - 9.2.2 Packaging your classes
- 9.3 Accessing Packages
 - 9.3.1 Accessing packages
- 9.4 Case Study: Banking Application: Building Packages
 - 9.4.1 Exploring the API packages
 - Lab Activity – Exploring the API packages
 - 9.4.2 Creating a banking package
 - Lab Activity – Build a Banking package
- 9.5 Using AWT
 - 9.5.1 Understanding Model View Controller pattern
 - 9.5.2 Applying inheritance concepts
 - 9.5.3 The AWT hierarchy
- 9.6 Case Study: Banking Application: Designing GUI's
 - 9.6.1 Applying concepts to the Banking application
 - Lab Activity – Designing a GUI to represent an ATM for customers
 - Lab Activity – Designing GUI interfaces

10 Creating GUI using AWT

- 10.1 Reviewing AWT
 - 10.1.1 The AWT classes
- 10.2 Steps to Create a GUI
 - 10.2.1 Designing the class
 - 10.2.2 Creating the components
 - Lab Activity – Creating the Components (TellerGui class)
 - Lab Activity – Creating the Components (ATMGui class)
 - 10.2.3 Selecting containers
 - Lab Activity – Selecting containers (TellerGui class)
 - Lab Activity – Selecting containers (ATMGui class)
 - 10.2.4 Layout managers
 - 10.2.5 Sizing components and containers
 - Lab Activity – Layout managers (TellerGui class)
 - Lab Activity – Layout Managers and adding components (ATMGui class)
 - 10.2.6 Displaying the GUI
- 10.3 GUI Functionality
 - 10.3.1 Event Model
 - 10.3.2 Event Objects
 - Lab Activity – Identifying Event Handler features in the TellerGui class
 - Lab Activity – Implement Event handling for the ATMGui class
 - 10.3.3 Listener classes
 - 10.3.4 Adapter classes
 - 10.3.5 Inner classes
 - 10.3.6 Anonymous classes
- 10.4 Case Study: Banking Application
 - 10.4.1 Applying concepts to the banking application
 - Lab Activity – The finishing touches to the Model View Controller pattern for the ATMGui class

11 Applets and graphics

- 11.1 Applets
 - 11.1.1 What are applets
 - 11.1.2 Launching Applets
 - 11.1.3 Applets and Security
 - 11.1.4 HTML document to host a Java applet
- 11.2 Creating an applet
 - 11.2.1 Creating an applet
 - Lab Activity – Creating an Applet to display employee information
- 11.3 Applet and AWT Class Hierarchies
 - 11.3.1 Class Hierarchies
- 11.4 Applet Methods
 - 11.4.1 Applet inherited methods
 - 11.4.2 Applet lifecycle methods
- 11.5 Graphical User Interface (GUI) components in Applets
 - 11.5.1 Components
 - 11.5.2 Color
 - 11.5.3 Font
 - 11.5.4 Other Applet Methods
 - Lab Activity – Changing the Location of components in an Applet
- 11.6 Applets and Event-Driven Programming
 - 11.6.1 Applets and Event-Driven Programming
 - Lab Activity – Changing the Location of the components in an Applet
- 11.7 Graphic objects
 - 11.7.1 Creating Graphic Objects
 - 11.7.2 Painting and repainting the graphics
 - Lab Activity – Calculator
- 11.8 Case Study: Banking Application
 - 11.8.1 Applying concepts to the banking application

12 Exceptions

- 12.1 Idea behind exceptions
 - 12.1.1 The limitations of traditional methods of exception handling
 - 12.1.1 Error handling in the Java Platform
- 12.2 Types of exceptions
 - 12.2.1 Types of errors
- 12.3 How Exception Handling Works
 - 12.3.1 Throwing and handling exception objects
- 12.4 Exceptions Objects
 - 12.4.1 Class Throwable
 - 12.4.2 Exception and its subclasses
 - 12.4.3 RunTime Exceptions
 - 12.4.4 Error Exceptions
- 12.5 Dealing with Exceptions
 - 12.5.1 Advertising exceptions
 - 12.5.2 Handle and or Declare a throws clause or the 'catch or specify requirement'
 - 12.5.3 How to use the try-catch block
 - Lab Activity – Testing for a run-time exception
 - 12.5.4 How to specify the Exceptions a method can throw
 - 12.5.5 Catching multiple Exceptions
 - 12.5.6 The finally block
 - Lab Activity – Using finally
- 12.6 Defining and using your own Exception
 - 12.6.1 Creating your own exception class of objects
 - 12.6.2 Using user-defined exceptions
 - 12.6.3 Handling user-defined exceptions
 - Lab Activity – Creating your own exception classes
- 12.7 Structuring a method and the execution sequence
 - 12.7.1 Try-catching-finally
 - 12.7.2 Execution sequence-normal execution of a method
 - 12.7.3 Execution when an exception is thrown
 - 12.7.4 Execution when an exception is not caught
 - 12.7.5 Nested try blocks
 - 12.7.6 Re-throwing Exceptions some additional uses of the throw clause
- 12.8 Overriding and Exceptions
 - 12.8.1 Rules for overriding methods – declaring exceptions
- 12.9 Case Study: Banking Application
 - 12.9.1 Applying concepts to the Banking application
 - Lab Activity – Creating Exceptions for the banking application

13 Files, Streams, Input and output

- 13.1 Understanding Files
 - 13.1.1 The File class
 - Lab Activity – Displaying filestatistics
 - 13.1.2 The Random Access file class
 - Lab Activity – Using the RandomAccessFile to seek positions within a file
- 13.2 Input and Output classes
 - 13.2.1 Java io package and the io class hierarchy
 - 13.2.2 Understanding Streams
 - 13.2.3 Low-level and high-level streams
- 13.3 A How To for Input and Output operations
 - 13.3.1 Binary or Individual Byte I/O
 - 13.3.2 Character or textual I/O
 - 13.3.3 Primitive Data Type I/O

- 13.4 Storing Objects in a file
 - 13.4.1 Writing objects - Serialization
 - 13.4.2 Writing objects
 - 13.4.3 Reading objects from a file
- 13.5 Case Study: Banking Application
 - 13.5.1 Applying concepts to the banking application
 - Lab Activity – Writing Customer objects to a file
 - Lab Activity – Reading Customer objects from a file

14 Collections

- 14.1 The java.util Package
 - 14.1.1 The java.util package
- 14.2 Collections
 - 14.2.1 Understanding collections
 - 14.2.2 Collection storage technologies
 - 14.2.3 Properties of collections
 - 14.2.4 Types of collections
- 14.3 The Collections Framework
 - 14.3.1 Introduction to the collections framework
 - 14.3.2 Collection interfaces
 - 14.3.3 Collection classes
 - 14.3.4 Use of Set objects – HashSet and TreeSet
 - 14.3.5 Use of list objects
 - Lab Activity – An ArrayList
 - 14.3.6 Use of Map objects
 - 14.3.7 Iterators
 - Lab Activity – Traversing an ArrayList
 - Lab Activity – Creating a collection to store Integer objects
 - 14.3.8 Sorting and shuffling – Methods of the Collection class
- 14.4 Case Study: Banking Application
 - 14.4.1 Applying concepts to the banking application
 - Lab Activity – Using a SortedSet to manage a sorted set of Customer objects
 - Lab Activity – Use of a Vector to retrieve data from a file and store Customer objects

15 Threads

- 15.1 Threads and Multithreading
 - 15.1.1 Understanding threads and Multithreading
- 15.2 The Thread Class and Runnable Interface
 - 15.2.1 Thread class
 - 15.2.2 Runnable Interface
 - 15.2.3 Thread Class methods
- 15.3 Creating and running threads
 - 15.3.1 The run() method
 - 15.3.2 Running threads
 - Lab Activity – Displaying Messages using Threads
- 15.4 The lifecycle of a thread
 - 15.4.1 Lifecycle of a thread
- 15.5 Managing threads
 - 15.5.1 Blocked state - Suspending threads
 - 15.5.2 Thread scheduling
 - Lab Activity – Thread scheduling and priority assignments
 - 15.5.3 Stopping threads
 - 15.5.4 Synchronizing and deadlocks
 - 15.5.5 Communication between threads
 - Lab Activity – Thread Communication and thread control