This course introduces object-oriented programming (OOP) using the Java language. Course topics include a review of structured programming concepts, use of a Java Integrated Development Environment (IDE), and an introduction to object-oriented design and coding methodology. The object-oriented approach to Java programming emphasizes data encapsulation, data abstraction, inheritance, polymorphism, use of built-in classes and libraries, class hierarchies, reusable design, applets incorporating graphical user interfaces, and event-driven programming.

PREREQUISITES
MATH 104 or higher
CSIS 123 Programming Fundamentals

EXPECTED STUDENT OUTCOMES IN THE COURSE (ESO)
Upon completion of this course, the student will be able to:

1. Apply terminology and coding techniques associated with object-oriented design and programming.
2. Use built-in Java classes and libraries, including string formatting, math functions, and input/output streams.
3. Design and implement a Java command-line interface (CLI) application to perform a significant computing task, such as solving an applied mathematics problem.
4. Design and code classes and class hierarchies based on software engineering principles.
5. Design and implement a Java application or applet utilizing a simple graphical user interface using common controls (text fields, buttons, images, and similar items).
6. Employ an integrated development environment (IDE) and tools to support development, testing, debugging, and distribution of code products (using Java Archives).
7. Code and document programs according to a programming style specification.

GENERAL EDUCATION OUTCOMES (ESO)
Specify which general education outcomes, if any, are substantially addressed by the course. Numbers in parentheses identify the Expected Student Outcomes linked to the specific General Education Outcome.

Outcomes ESO
PROGRAM-LEVEL OUTCOMES

CAREER AND TECHNICAL EDUCATION PROGRAM OUTCOMES
Specify which Career and Technical program outcomes, if any, are substantially addressed by the course by completing the “Career and Technical Education template” to show the relationship between course and program outcomes to assessment measures.

1. Use industry-specific software and/or apply troubleshooting skills to solve problems.
2. Create and define solutions to real-life business challenges.
3. Recognize the need for continued professional development.

CLASS-LEVEL ASSESSMENT MEASURES
Student accomplishment of expected student outcomes may be assessed using the following measures. (Identify which measures are used to assess which outcomes.)

1. Exercises (1-7)
2. Individual or group design projects (1-7)
3. Individual or group programming problems (1-7)
Individual instructors may order this outline as fits the needs of their individual courses. In addition, they may place more emphasis on some areas than on others. What is assured is that this particular list is covered in the course. Other topics may be added to a course as the instructor sees fit, and as time and interest allow. An *asterisk can be used to mark an item as optional.

I. Procedural programming in Java
   A. Procedural programming syntax and techniques
   B. Use of integrated development environment to enter, edit, debug, and test programs
   C. Compiled versus interpreted execution
   D. Primitive data types
   E. Basic input/output operations
   F. Control structure coding
      1. Decision structures
      2. Repetition structures
   G. Functions (methods)
   H. Call-by-value and call-by-reference comparison
   I. Scope and duration of variables and constants
   J. Primary data structures, including strings, single-dimension arrays, records
   K. Principles of structured, modular design
   L. Coding style and conventions

II. Object-Based Programming
   A. References
   B. Class mechanisms to support encapsulation and information-hiding (abstraction)
   C. Class components
   D. Taxonomy of data elements and methods
   E. Access specifiers
   F. Built-in classes
   G. User-defined classes
   H. Class object instantiation
   I. Method overloading
   J. Deep and shallow copies of parameters
   K. Constructor methods
   L. Run-time memory management (Java memory model and built-in garbage collection)

III. Object-Based Design
   A. Information hiding and encapsulation
   B. Abstract data types
   C. Attributes and behaviors of a class
   D. Composition (has-a) and inheritance (is-a) relationships

IV. Object-Oriented Programming
   A. Class mechanisms to support inheritance
   B. Class mechanisms to support polymorphism
   C. Implementation of friendly and package access
D. Built-in class hierarchies
E. User-defined class hierarchies
F. Class definition separation into multiple files within project

V. Simple Graphical User Interfaces (GUI)
   A. Use of simple graphical components to build an interface
   B. Human-centered development
   C. Principles of screen design
   D. Component-based programming in an API environment

VI. Fundamental Data Structures
   A. Strings and string processing
   B. Multi-dimension arrays with searching and sorting algorithms
   C. Classes for collections, sorting, searching