COURSE DESCRIPTION

Dept., Number	**CSC 20**	Course Title	**Programming Concepts and Methodology II**
Semester hours	**3**	Course Coordinator	**Roxalie Jones**
		URL (if any):	**http://gaia.ecs.csus.edu/~jonesr/**

Catalog Description

Application of object-oriented techniques for systematic problem analysis and specification, design, coding, testing, and documentation. Semester-long project approach emphasizing larger programs. Managing program complexity using abstraction. Introduction to algorithm analysis and Big-O notation. Advanced language features. Basic sorting and searching algorithms. Recursion. Lecture two hours, technical activity and laboratory two hours. Prerequisite: CSC 15.

Textbook

Michael Main, Data Structures and Other Objects Using Java, 3rd Edition, Addison-Wesley, 2006.

Course Goals

1. Experience designing and developing large, complex programs in the context of a semester-long project.
2. Use of object-oriented techniques to foster abstraction.
3. A disciplined approach to the design, coding, testing and debugging of programs.
4. Use of appropriate tools and techniques for each step of program development.
5. Illustration of the power of abstraction via the use of multiple representations of linear lists.
6. Reinforcement and expansion of object-oriented skills.
7. Basic sorting and search algorithms and their analysis.

Prerequisites by Topic

Thorough understanding of:

- Implementation of programs using an object-oriented programming language.
- Use of classes/objects/methods to solve problems.
- Fundamental syntax of the programming language used in the course.
- Concepts of type, variable, one dimensional arrays of simple types.
- Concepts of assignment; arithmetic, relational, and Boolean expressions.

Basic understanding of:

- Design of programs using an object-oriented programming language.
- Design and implementation of classes.
- Program development process and the relevant tools associated with that process.
- Stepwise refinement and modularity.

- Debugging techniques, including the use of a symbolic debugger.
- Programming style and program documentation concepts.
- Parameter passing and its implications.
- Scope rules.
- Strings.
- Interactive text I/O.
- One-dimensional arrays of objects.
- Standard algorithms such as: sequential search, simple sorts.

Exposure to:

- Procedural abstraction, data abstraction, abstract data types (ADT), and information hiding.
- Software testing techniques.
- Compiling and executing programs, and the concept of a virtual machine.
- Multi dimensional arrays.
- Lifetime of variables.
- Use of class libraries.

Major Topics Covered in the Course

The topic of program design permeates virtually all topics discussed in the course.

1. Problem analysis (1 hour).
2. Programming methodology (9 hours).
    - Module by module development, implementation, and testing using concepts such as drivers, stubs, transaction files, and advanced debugging techniques
    - Developing appropriate test data and testing strategies.
3. Advanced language features (6 hours).
    - Strings.
    - Stream I/O including text files.
    - Objects and classes including abstract classes, interfaces, inheritance and polymorphism.
    - Exceptions and exception handling.
    - Introduction to graphical user interface (GUI) and event-handling.
4. Importance of abstraction in program development (8 hours).
    - Relationship between the object-oriented approach and the concept of abstraction.
    - Abstraction as a tool for dealing with complex problems.
    - Illustration of abstraction through the use of multiple representations of a linear list.
5. Algorithms (6 hours).
    - Introduction to analysis of algorithms and Big-0 notation.
    - String manipulation.
    - Introduction to recursive algorithms.
    - Basic sorting and searching algorithms.

Outcomes

Thorough understanding of:

- Design and implementation of medium-size programs using object-oriented techniques, data abstraction, and procedural abstraction including proper programming style, program documentation, and testing and debugging techniques.
- The program development process in Java: interpreting, compiling, executing a program, the concept of the virtual machine and the use of a symbolic debugger.
- Classes/objects and one-dimensional arrays of objects.
- Sub-programs and their appropriate use: parameter passing, scope of identifiers, information hiding and the issue of copying objects versus referencing objects.
- Basic sorting and searching algorithms such as insertion sort and binary search.

Basic understanding of:

- The benefit of abstraction and information hiding, and the support that object-oriented programming provides, including tools and techniques to allow generic data types.
- The basic criteria involved in designing classes and their relationships.
- Efficiency concerns associated with the selection of a proper algorithm.
- Advanced programming language issues including: lifetime of variables, dynamic memory allocation and the use of references, multi dimensional arrays, exceptions and exception handling, inheritance, interfaces, abstract classes.
- Recursion as a problem-solving technique and its implementation in programming languages.
- Representation of various abstract data types and their associated algorithms: strings, linear lists, (including cursor and linked representation).

Exposure to:

- Big-O notation and ability to analyze simple algorithms.
- CRC cards and UML diagrams.
- Polymorphism.
- Event-driven programming and use of GUI components.

Laboratory Projects

Assignments usually consist of problem-solving efforts, program design, and the development of subprograms that are part of the total effort to build the semester project. Examples of real programs will be presented and discussed. Assignments on problem solving and the various stages of program design and development, e.g., specification, testing, structured walkthroughs, will be delivered by students. Hands-on demonstrations on computers, e.g., performance of algorithms, software tools for program design and development, advanced language features. Construction of a large program via the development and testing of its modules. Usually involves a list of objects, and one of the attributes of the objects in the list is itself a list. Alternate representations of linear lists are substituted into an application program to illustrate the effectiveness of encapsulation and abstraction.

Estimated Curriculum Category Content (Semester hours)

Area	*Core*	*Advanced*	*Area*	*Core*	*Advanced*
Algorithms	0.5		Data Structures	1.0	
Software Design	0.5		Prog. Languages	1.0	
Comp. Arch.					

Oral and Written Communications

No significant component.

Social and Ethical Issues

No significant component.

Theoretical Content

No significant component.

Problem Analysis

The laboratory assignments are designed so that the students build an application over the course of the semester. Although students are not expected to do the analysis of the application themselves, extensive discussions take place on its characteristics.

Solution Design

The design of the system is done by the instructor, but extensive discussions of software engineering principles, including modularization, data encapsulation, information hiding, data abstraction, algorithm analysis, and the advantages and disadvantages of various representations of data structures occur throughout the semester. The lab assignments are designed to equip the students with a good understanding of engineering design concepts.

/sj