COURSE DESCRIPTION

Dept., Number	**CSC 15**	Course Title	**Programming Concepts and Methodology I**
Semester hours	**3**	Course Coordinator	**Ted Krovetz**
		URL (if any):	**http://krovetz.net/csus/**

Catalog Description

Programming concepts using an object-oriented programming language. Introduction to methodologies for program design, development, testing, and documentation. Topics include program design, algorithm design, number systems, classes and objects, methods (functions), control structures, arrays, and interactive input/output. Lecture two hours, technical activity and laboratory, two hours. Prerequisites: At least a C- grade in CSC 10 or equivalent programming experience in a high-level programming language, and a passing score on the Entry Level Math exam (ELM).

Textbook

Tony Gaddis, Starting out with Java: Early Objects, 3rd Edition, Addison-Wesley, 2008.

Course Goals

1. To give experience in designing, implementing, testing, and documenting computer programs using an object-based approach, modularity, and stepwise refinement.
2. To provide an introduction to data abstraction through the development and use of classes.
3. To provide an understanding of number systems, data types, control structures, and procedural abstraction.
4. To help students improve problem-solving skills.

Prerequisites by Topic

Thorough understanding of:

- The concept of a variable.

Basic understanding of:

- Design of simple algorithms using: sequence, repetition, conditionals; for example, using a loop to compute the sum of n integers.
- Converting those algorithms into executable programs using some programming language.

Exposure to:

- Use of some form of subprogram (e.g., method, procedure, function, subroutine).

Major Topics Covered in the Course

1. Using classes/objects, including class libraries (3 hours).
2. Designing and developing classes/objects, data abstraction (3 hours).
3. Designing, developing, and documenting programs using classes/objects (2 hours).
4 Algorithm development, detailed design (3 hours).
5. Program testing (1 hour).
6. Compiling, executing, interpreting, virtual machine (1 hour).
7. Identifiers, variables, constants, basic (primitive) types (1 hour).
8. Number systems (2 hours).
9. Strings and string manipulation (1 hour).
10. Interactive input/output (1 hour).
11. Operators, expressions, assignments (1 hour).
12. Boolean expressions, conditional statements (2 hours).
13. Iteration (2 hours).
14. Scope of identifiers, lifetime of variables (1 hour).
15. Procedural abstraction, methods (functions), stepwise refinement, parameters (2 hours).
16. Arrays, introduction to sorting and searching algorithms (2 hours).
17. Exams (2 hours).

Outcomes

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.
- Number system conversions (bases 2, 8, 10, 16).

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.

Laboratory Projects

1. Edit - compilation - execution cycle, use of Integrated Development Environment (1 week).
2. Sequential programming with arithmetic expressions (1 week).
3. Introduction to classes/methods using strings (2 weeks).
4. Designing, implementing, and testing a program with repetition and conditional structures, accumulation, methods (2 weeks).
5. Debugging techniques, using symbolic debugger (1 week).
6. Creating, testing, and using simple classes with methods (2 weeks).
7. Experimenting with parameters and object references (1 week).
8. Designing, implementing, and testing a more complicated class with methods (2 weeks).
9. Designing, implementing and testing a program with arrays (including arrays of objects) (2 weeks).
10. Experimenting with programs using inheritance and Graphical User Interface (GUI) (1 week).

Estimated Curriculum Category Content (Semester hours)

Area	*Core*	*Advanced*	*Area*	*Core*	*Advanced*
Algorithms	0.5		Data Structures	0.5	
Software Design	1.0		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

Most laboratory assignments demonstrate requirements analysis, and discussion in lab includes this topic. The student is not expected to conduct that analysis.

Solution Design

Most laboratory assignments have significant design components. In most cases, the assignment descriptions discuss significant issues of top-level (architecture) design. Discussion during the lab sessions focuses on design issues. Students are required to provide detailed design represented in the form of pseudocode when receiving assistance during the lab sessions. Please see the Laboratory Projects section above.

/sj