

COURSE DESCRIPTION

Dept., Number	CS201	Course Title	Accelerated Introduction to Computer Science
Semester hours	4	Course Coordinator	Matthew Bauer, Senior Lecturer

Current Catalog Description

Problem-solving and design using an object-oriented programming language. Introduces a variety of problem solving techniques, algorithms, and data structures in object-oriented programming. Prerequisites: CS105 or CS 115 or experience using any programming language. (3-2-4)

Textbook

Programming and Problem Solving with Java, Second Edition, Jones & Bartlett Publishers, Inc., copyright 2008 by Nell Dale, Chip Weems, ISBN: 0763734020

or

Java 6 Illuminated (w/ CD), Anderson, 2nd Edition, 2008

References

None

Course Outcomes

Students should be able to:

- Analyze and explain the behavior of simple programs involving the following fundamental programming constructs: assignment, I/O (including file I/O), selection, iteration, methods
- Write a program that uses each of the following fundamental programming constructs: assignment, I/O (including file I/O), selection, iteration, methods
- Break a problem into logical pieces that can be solved (programmed) independently.
- Develop, and analyze, algorithms for solving simple problems.
- Use a suitable programming language, and development environment, to implement, test, and debug algorithms for solving simple problems.
- Write programs that use each of the following data structures (and describe how they are represented in memory): strings, arrays
- Explain the basics of the concept of recursion.
- Write, test, and debug simple recursive functions and procedures.
- Explain and apply object-oriented design and testing involving the following concepts: data abstraction, encapsulation, information hiding, inheritance, polymorphism

- Use a development environment to design, code, test, and debug simple programs, including multi-file source projects, in an object-oriented programming language.
- Implement basic error handling
- Solve problems by creating and using sequential search, binary search, and quadratic sorting algorithms (selection, insertion)
- Determine the time complexity of simple algorithms.
- Apply appropriate problem-solving strategies
- Use APIs (Application Programmer Interfaces) and design/program APIs

Relationship between Course Outcomes and Program Outcomes

The following Program Outcomes are supported by the above Course Outcomes:

- a. An ability to apply knowledge of computing and mathematics appropriate to the discipline
- b. An ability to analyze a problem, and identify and define the computing requirements appropriate to its solution
- c. An ability to design, implement and evaluate a computer-based system, process, component, or program to meet desired needs
- i. An ability to use current techniques, skills, and tools necessary for computing practices
- j. An ability to apply mathematical foundations, algorithmic principles, and computer science theory in the modeling and design of computer-based systems in a way that demonstrates comprehension of the tradeoffs involved in design choices
- k. An ability to apply design and development principles in the construction of software systems of varying complexity

Prerequisites by Topic

CS105 or CS 115 or experience using any programming language.

Major Topics Covered in the Course

1. Fundamental data storage and manipulation (types and variables, statements and expressions)	5 hours
2. Functions	3 hours
3. Classes (classes and objects, instance variables and instance methods, and encapsulation).	5 hours
4. Flow of control (Boolean expressions, conditional statements, and loops).	10 hours
5. Vectors	5 hours
6. Review of CS115 material	10 hours
7. Inheritance (subclasses, dynamic binding, abstract classes, and interfaces).	5 hours
8. Strings	3 hours
9. Introduction to recursion.	3 hours
10. Searching and sorting algorithms (linear and binary search, selection sort, insertion sort, and quick sort - introduced via recursive versions).	5 hours
11. Algorithm analysis.	2 hours
12. Problem Solving approaches (This section is dispersed appropriately throughout the semester to illustrate the above techniques.)	5 hours
13. Software Engineering – design, testing, debugging (This section is dispersed appropriately throughout the semester to illustrate the above techniques.)	10 hours
Exams	4 hours
Final Exam	-
	75 hours

Assessment Plan for the Course

End of every semester Course Objective Assessments by CS department. End of semester Course Evaluations by IIT. Reviewed every Spring semester by CS Undergraduate Studies Committee for possible updates in the following Fall. Once every 4-5 years a detailed review of

all materials for the course is made by the CS Undergraduate Studies Committee.

How Data in the Course is Used to Assess Program Outcomes (unless adequately covered already in the assessment discussion under Criterion 4)

See the assessment discussion under Criterion 4

For a computer science program

Estimate Curriculum Category Content (Semester hours)

Area	Core	Advanced	Area	Core	Advanced
Algorithms	.66		Software design	.66	
Data structures	.66		Concepts of programming languages	2	