COURSE OUTLINE OF RECORD

Number: CS G154  TITLE: Data Structures with Java

ORIGINATOR: Cristian Racataian  EFF TERM: Fall 2014
FORMERLY KNOWN AS: Java Data Structures, Advanced
DATE OF OUTLINE/REVIEW: 03-06-2014
CROSS LISTED COURSE: TOP NO: 0707.20
CID: COMP 132

SEMESTER UNITS: 4.0
HRS LEC: 54.0  HRS LAB: 54.0  HRS OTHER: 0.0
CONTACT HRS TOTAL: 108.0
STUDY NON-CONTACT HRS RECOMMENDED: 108.0

CATALOG DESCRIPTION:
Advanced programming techniques and Object Oriented Programming principles in Java will be exploited in learning the concepts of data structures. Students will gain theoretical and hands-on experience with the implementation of typical data structures (e.g., arrays, queues, stacks, linked-lists, trees, hashing, and graphs) used in programming applications. Principles of recursion, sorting, searching, optimization, classes, objects, inheritance, and polymorphism will be explored and practiced.

JUSTIFICATION FOR COURSE:
C-ID Compliance

PREREQUISITES:
• CS G153: Java Programming, Introduction with a minimum grade of C or better
  or
• CS G175: C++ Programming with a minimum grade of C or better

COREQUISITES:

ADVISORIES:

ASSIGNED DISCIPLINES:
Computer science

MATERIAL FEE: Yes [ ] No [X] Amount: $0.00

CREDIT STATUS: Noncredit [ ] Credit - Degree Applicable [X] Credit - Not Degree Applicable [ ]

GRADING POLICY: Pass/No Pass [ ] Standard Letter [X] Not Graded [ ] Satisfactory Progress [ ]

OPEN ENTRY/OPEN EXIT: Yes [ ] No [X]

TRANSFER STATUS: CSU Transferable[ ] UC/CSU Transferable[X] Not Transferable[ ]

BASIC SKILLS STATUS: Yes [ ] No [X] LEVELS BELOW TRANSFER: Not Applicable

CALIFORNIA CLASSIFICATION CODES: Y - Not Applicable

NON CREDIT COURSE CATEGORY: Y - Not applicable, Credit Course

OCCUPATIONAL (SAM) CODE: C

REPEATABLE ACCORDING TO STATE GUIDELINES: No [X] Yes [ ] NUMBER REPEATS:

REQUIRED FOR DEGREE OR CERTIFICATE: No [ ] Yes [X]
Computer Science(Associate in Science for Transfer)

GE AND TRANSFER REQUIREMENTS MET:
CSU Transfer Course
  A. Transfers to CSU
  Core course for AS-T
UC Transfer Course
   A. Transfers to UC
Degree Applicable
   AS-T Degree Applicable

PROGRAM LEVEL LEARNING OUTCOME(S) Supported by this course:

Design software components and specification to satisfy small business and scientific problem requirements.

Design and implement algorithms that include basic computation techniques, simple I/O, conditional and iterative structures, and the definition of functions.

Describe formal tools of symbolic logic as they relate to real-life situations, program correctness, database queries, and algorithms.

Utilize object oriented principles for class hierarchies and inheritance to create computing solutions of simple to moderate complexity.

Implement programs at machine language level using fundamental high-level programming constructs.

COURSE LEVEL STUDENT LEARNING OUTCOME(S) Supported by this course:

1. Implement complex data storage mechanisms and manipulation algorithms.
2. Write programs that use abstract data structures
3. Explain software development methodologies and debugging techniques
4. Describe object-oriented class hierarchy and inheritance.
5. Implement, test, and debug simple recursive functions and procedures

COURSE OBJECTIVES:
I Gain theoretical and experiential understanding of the following subjects:
I.1. Object Oriented Programming
I.2. Classes and Objects
I.3. Inheritance and Polymorphism
I.4. Searching and Sorting
I.5. Stacks, Queues, and Trees
I.6. Linked-Lists
I.7. Hashing and Graphs
I.8. Algorithm Optimization and Efficiency
I.9. Software development and debugging techniques

COURSE CONTENT:
   LECTURE CONTENT:
      A. Review of Java: Primitive types; Arrays; Records, String and String processing
      B. Software development lifecycle and debugging techniques
      C. Declaration models, type-checking
      D. Garbage collection
E. Data representation in memory
F. Static, stack, and heap allocation
G. Runtime storage management
H. Pointers and references
   I. Type parameters and parameterized types-templates or generics
J. Recursive mathematical functions
K. Simple recursive procedures
L. Divide-and-conquer strategies
M. Recursive backtracking
N. Design of Stacks, Queues and Trees using arrays
O. Design of Stacks, Queues and Trees using linked-lists
P. Discussion of Classes and Objects
Q. Class hierarchies
R. Collection classes and iteration protocol
S. Redesign of Stacks and Queues as objects
T. Discussion and exercises in Hashing algorithms
U. Implementation strategies for trees
V. Searching and sorting algorithms
W. Algorithm design and optimization
X. Using inheritance to implement Priority Queues
Y. Using Polymorphism principles to store objects of varying types in a Collection Class

LABORATORY CONTENT:

A. Pointers and references
B. Classes and object oriented programming
C. Inheritance and polymorphism
D. Templates and parameterized types
E. Linked lists and iterators
F. Recursive techniques
G. Stacks using arrays or linked lists
H. Queues using arrays or linked lists
   I. Binary trees
   J. Hashing techniques
   K. Searching and sorting
   L. Graphs

METHODS OF INSTRUCTION:

A. Lecture:
B. Lab:
C. Online:
D. Independent Study:

INSTRUCTIONAL TECHNIQUES:

A. Lecture and/or discussion
B. Field experience
C. Work experience

COURSE ASSIGNMENTS:

Reading Assignments

Students will be assigned multiple chapters from the required books. External material will be made known to students to encourage further studies into specific topics. Various current (up-to-date) handouts will be made available to students on OOP (Object Oriented Programming) technologies, tools, and software development strategies.
Out-of-class Assignments

An optional library research paper will promote further study and research in current Java programming or other related topics selected by the student and approved by the instructor.

Writing Assignments

Students will be required to complete software development projects presented to them in the form of business automation problems requiring solution implementation. Students will be required to write documentation on their projects.

METHODS OF STUDENT EVALUATION:

Midterm Exam
Final Exam
Short Quizzes
Report
Projects (ind/group)
Problem Solving Exercises
Oral Presentations
Skills Demonstration

Demonstration of Critical Thinking:

Students will be demonstrating their laboratory projects. Optional research papers and classroom presentations will further demonstrate their ability in critical thinking and problem solving.

Required Writing, Problem Solving, Skills Demonstration:

Students will be required to complete software development projects presented to them in the form of business automation problems requiring solution implementation. Students will be required to write documentation for their projects.

TEXTS, READINGS, AND RESOURCES:

TextBooks:

Other:
1. A syllabus, and multiple reference material related to the latest software development technologies will be distributed by the instructor.

LIBRARY:

Adequate library resources include: Print Materials

Comments:

Attachments:

Attached Files