|  |  |
| --- | --- |
| **Course:** | **COSC 190 - Intermediate Programming** |
| **Course Description:** | You will receive instruction in working with data structures and creating recursive methods. You will utilize standard file input/output techniques. You will learn about the basic tenants of functional programming. You will become familiar with and be able to manipulate such advanced data structures as stacks and queues. The course content includes introductory GUI development, thread-based programming, and builds towards an introduction of network programming techniques (sockets and database access). |
| **Pre Requisites:**  **Co Requisites:** | * COSC 180 |
| **Course Hours:**  **Credit Units:** | * 90 hours (6 hours/week x 15 weeks) * 6.0 credits |
| **Student Assessment:**  **Grade/Passing Grade:**  **PLAR Method:** | * 3-4 assignments (15%), 2 midterm exams (45%), final exam (40%) * 50% * Evidence file and/or Challenge Exam |
| **Learning Resources:** | * Liang, Y. Daniel. (2015). *Introduction to Java Programming: Comprehensive Version* (11th ed.). Upper Saddle River, New Jersey: Prentice Hall. * Safari Web Site -> *Java Threads and The Concurrency Utilities* * Safari Web Site -> *Learning network Programming with Java* * Safari Web Site -> *Learning Java Functional Programming* * Lecture notes and supplemental materials. * Java API documentation. |
| **Learning Outcomes:** | 1. Analyze common array algorithms for searching. 2. Create recursive methods. 3. Design robust programs using appropriate exception handling. 4. Use dynamic data structures. 5. Use functional programming techniques 6. Design programs that present information through a Graphical User Interface (GUI). 7. Design programs for data storage and retrieval from files. 8. Create programs that use multithreading. 9. Create programs that Use Networking Techniques 10. Create programs that access remote Databases |
| **Updated/Updated by:** | **Alex Wang Date: Jan 2023** |
| **Approved by Program Head:**  **Approved by Academic Chair:** | **Joseph Herbert Date: Jan 2023**  **Date:** |

|  |  |  |
| --- | --- | --- |
| 1. **Analyze common array algorithms for searching and sorting.** | Describe how Insertion Sort Works  Write programs that utilize Insertion Sort on Arrays  Describe how Bubble Sort Works  Write programs that utilize Bubble sorts on arrays  Describe how Merge Sort Works  Write programs that Utilize merge Sort on Arrays  Describe how binary search works  Write programs that utilize binary searches on sorted arrays.  Use Big 0 Notation to describe the relative effectiveness of various search/sort algorithms. | Reading in chapter 22; lecture notes; in-class exercises. |
| **Assessment Tools:**  Hand-in assignments, examinations |  |  |
| 1. **Create recursive methods.** | Define recursion.  Identify the three components of a recursive method.  Create recursive methods.  Create recursive methods for that do sorting.  Develop and implement recursive methods to implement binary searches. | Reading in chapter 18; lecture notes, in-class exercises.  Reading in chapter 18; lecture notes, in-class exercises  .  Reading in chapter 18; lecture notes, in-class exercises.  Reading in chapter 18; lecture notes, in-class exercises. |
| Assessment Tools: Hand-in assignments, examinations |  |  |
| 1. **Design robust programs using appropriate exception handling.** | Describe the flow of program execution when an exception occurs.  Use pre-defined exception classes.  Create programmer-defined exceptions.  Write code using exception handling clauses (try/catch/finally).  Create code that can throw exceptions.  Apply assertions to help ensure program correctness. | Reading in chapter 12; lecture notes; in-class exercises.  Reading in chapter 12; lecture notes; in-class exercises.  Reading in chapter 12; lecture notes; in-class exercises.  Reading in chapter 12; lecture notes; in-class exercises.  Reading in chapter 12; lecture notes; in-class exercises.  Reading in chapter 12; lecture notes; in-class exercises. |
| **Assessment Tools:**  Hand-in assignments, examinations |  |  |
| 1. **Use dynamic data structures.** | Discuss the Vector and ArrayList classes for storing generic objects.  Use generic types to have type-safe collections of objects.  Describe a Collections framework.  Describe stacks and queues.  Demonstrate programs that use stacks and queues.  Demonstrate the use of iterators.  Use an enhanced for loop.  Use a foreach operator to move through a collection  Choose between a stack and queue to best suit an application.  Discuss Deep and Shallow object Copies (Cloning operations). | Reading in chapter 19, 20; lecture notes; in-class exercises.  Reading in chapter 19, 20; lecture notes; in-class exercises.  Reading in chapter 19, 20; lecture notes; in-class exercises.  Reading in chapters 19, 20; lecture notes; in-class exercises.  Reading in chapter 19, 20; lecture notes; in-class exercises.  Reading in chapter 19, 20; lecture notes; in-class exercises.  Reading in chapter 19, 20; lecture notes; in-class exercises.  Reading in chapters 19, 20; lecture notes; in-class exercises.  Readings in chapter 19, 20; lecture notes, in class-exercises. |
| **Assessment Tools:**  Hand-in assignments, examinations |  |  |
| 1. **Use Functional Programming techniques** | Identify the basic tenants of function programming  Describe Lambda Expressions  Identify standard Functional Interfaces  Write programs that use Lambda Expressions for sorting and searching applications  Use Lambda functions as aggregate operators on collection streams  Create Stream pipelines for parallel processing of large collections | Learning Java Functional Programming; in class exercises  Learning Java Functional Programming; in class exercises  Learning Java Functional Programming; in class exercises  Reading in chapter 30; In class exercises  Reading in chapter 30; In class exercises |
| **Assessment Tools:**  Hand-in assignments, examinations |  |  |
| 1. **Design programs that present information through a Graphical User Interface (GUI).** | Describe how a GUI based application differs in concept from a console based application  Write program that creates Basic Shapes that are drawn on a GUI platform.  Describe how images, stored both locally and remotely, can be displayed as part of a GUI program.  Create simple animations  Design applications that use the various GUI components (labels, text fields, buttons, and menus).  Plan appropriate interfaces by using various layout management schemes.  Create interactive GUI programs. | Reading in chapters 14-16, 34; lecture notes; in-class exercises.  Reading in chapters 14, 33; lecture notes; in-class exercises.  Reading in chapters 14-16, 33; lecture notes; in-class exercises.  Reading in chapters 14-16, 33; lecture notes; in-class exercises.  Reading in chapters 14-16, 33; lecture notes; in-class exercises.  Reading in chapters 14-16, 33; lecture notes; in-class exercises. |
| **Assessment Tools:**  Hand-in assignments, examinations |  |  |
| 1. **Design programs for data storage and retrieval from files.** | Describe an input/output stream.  Distinguish between binary files and text files.  Create programs that write data to and read data from text files.  Create programs that write data to and read data from binary files.  Create programs that use Object Serialization to store and retrieve the state of objects.  Create programs that can read and write to Data Files using a Random Access File Protocol | Reading in chapters 12, 17; lecture notes; in-class exercises.  Reading in chapters 12, 17; lecture notes; in-class exercises.  Reading in chapters 12, 17; lecture notes; in-class exercises.  Reading in chapter 12,17; lecture notes; in-class exercises.  Reading in chapter 12,17; lecture notes; in-class exercises.  Reading in chapter 12,17; lecture notes; in-class exercises.  Reading in chapter 12,17; lecture notes; in-class exercises. |
| **Assessment Tools:**  Hand-in assignments, examinations |  |  |
| 1. **Create programs that use multithreading.** | Discuss the advantages of multi-threaded programming.  Discuss the life cycle of a thread.  Create programs that create, manage and destroy threads.  Write programs that do sorting and searching utilizing a thread based approach  Describe “deadlock” and identify synchronization techniques that can be used to deal with deadlock situations.  Design multithreaded programs using synchronization. | * 1. External Reference notes; in-class exercises   2. External Reference notes; in-class exercises.   3. External Reference notes; in-class exercises.   4. External Reference notes; in-class exercises. |
| **Assessment Tools:**  Hand-in assignments, examinations |  |  |
| 1. **Create programs that use network programing techniques** | Describe socket communication.  Explain the concept of client-server programming.  Create programs that can communicate with each other using sockets | External Reference notes; in-class exercises.  External Reference notes; in-class exercises.  External Reference notes; in-class exercises. |
| **Assessment Tools:**  Hand-in assignments, examinations |  |  |
| 1. **Create Programs that access remote Databases** | Identify a protocol that can be used to access back-end databases  Identify how a program can connect to a back-end database  Create programs that accesses a back end data and display data that is returned from that database.  Create programs that upload information to back-end databases | External Reference notes; in-class exercises  External Reference notes; in-class exercises  External Reference notes; in-class exercises  External Reference notes; in-class exercises |
|  |  |  |

© Copyright

Saskatchewan Institute of Applied Science and Technology

No part of the work(s) contained herein may be reproduced or copied in any form or by any means - graphic, electronic, or mechanical including photocopying, recording, taping of information and retrieval systems - without written consent of the Saskatchewan Institute of Applied Science and Technology.