COURSE OUTLINE

Information Technology Studies · The G. Raymond Chang School of Continuing Education · Ryerson University

COURSE NUMBER:	CXEC320
COURSE NAME:	Java Advanced
PRE-REQUISITE:	It is strongly recommended that students complete CXCP211 (Java Introduction) or have equivalent Java programming experience prior to taking this course.
REQUIRED TEXT	Introduction to Java Programming: Comprehensive Version, 9th Edition Y. Daniel Liang Pearson Education Inc. (Prentice Hall) 2013
ISBN NUMBER:	ISBN-13: 978-0-13-293652-1

COURSE DESCRIPTION:

Note; This course has been updated starting the Fall 2013 term. This 42 hour non-credit course provides learners with the skills required to create Java applications using object-oriented programming principles and applying OO design methodologies. Topics include using SWING components, the Java Graphics API, applets, exception handling, file IO, data structures, the Java Collection framework, and JDBC. Learners create stand-alone and multi-threaded applications, as well as client/server applications using TCP/IP sockets.

COURSE EVALUATION:

This course is comprised of a combination of lectures and labs. Upon successful completion of this course the learner will be assigned a letter grade based on the final grade. The final grade will be calculated based on the following components:

Assignments 50%
Tests 50%

Students are expected to complete all assignments, tests, and exams by the dates indicated in the course syllabus distributed by the instructor at the first class meeting. However, Ryerson University policies allow a student who misses a test or final exam to request an alternate arrangement to write a makeup exam for one of the following reasons only: medical illness, religious observance, or compassionate grounds, provided the student`s request is in writing and is accompanied by the appropriate documentation within the proscribed timeframe as outlined by the applicable policies listed at www.ryerson.ca/senate/policies. Please refer to the end of this outline for details.

1. Exploring Graphical User Interface Components

Upon completion of this unit the learner will be able to:

a. Explore the Java GUI API hierarchy
b. Design user interfaces using frames, panels and simple GUI
c. Analyze the role of layout managers and use the *FlowLayout*, *GridLayout*, and *BorderLayout* managers
d. Utilize *JPanel* to group components in a subcontainer
e. Apply common features such as borders, tool tips, fonts, and colours on *Swing* components
f. Create and manipulate buttons, labels, text fields, radio buttons, and check boxes

2. Drawing Graphics, Shapes, and Images

Upon completion of this unit the learner will be able to:

a. Draw graphics using the methods in the *Graphics* class
b. Override the *paintComponent* method to draw graphics on a GUI component
c. Use a panel as a canvas to draw graphics
d. Draw strings, lines, rectangles, ovals, arcs, and polygons
e. Obtain font properties using *FontMetrics*
f. Display an image on a GUI component
g. Develop reusable GUI components

3. Handling Exceptions and Text I/O

Upon completion of this unit the learner will be able to:

a. Explore exceptions and exception handling
b. Distinguish between the *Error* and *Exception* type, as well as between the checked versus unchecked type
c. Examine the Java exceptions class hierarchy
d. Throw exceptions in a method
e. Write a *try-catch-finally* block to handle exceptions
f. Develop applications with exception handling
g. Rethrow exceptions in a *catch* block
h. Construct chained exceptions
i. Define custom exception classes
j. Write data to a file using the *PrintWriter* class
k. Read data from a file using the *Scanner* class
l. Develop an application that replaces text in a file

4. Introducing Abstract Classes and Interfaces

Upon completion of this unit the learner will be able to:

a. Design and use abstract classes
b. Generalize numeric wrapper classes, *BigInteger* and *BigDecimal*, using the abstract *Number* class
c. Specify common behaviour for objects using interfaces
d. Define interfaces and classes that implement interfaces
e. Make objects cloneable using the *Cloneable* interface
f. Compare and contrast concrete classes, abstract classes, and interfaces

5. Working with Event-Driven Programming and GUI Components

Upon completion of this unit the learner will be able to:

a. Analyze events, event sources, and event classes
b. Define listener classes, register listener objects with the source object, and write code to handle events
c. Construct listener classes using inner classes
d. Create listener classes using anonymous inner classes
e. Explore various coding styles for creating and registering listener classes
f. Write programs to handle *MouseEvent* and *KeyEvent* events
g. Create listeners for *JCheckBox*, *JRadioButton*, and *JTextField* events
h. Explore *JTextArea*, *JComboBox*, *JListBox*, *JScrollBar*, and *JSlider* components
i. Display multiple windows in an application

6. Examining Applets, Multimedia, and Binary I/O

Upon completion of this unit the learner will be able to:

a. Convert GUI applications into applets
b. Embed applets in Web pages
c. Explore the applet security sandbox model
d. Develop a Java program that runs both as an application and as an applet
e. Override the applet life-cycle methods *init*, *start*, *stop*, and *destroy*
f. Locate resources such as images and audio using the *URL* class
g. Distinguish between text and binary I/O
h. Read and write to and from a file using *FileInputStream*, *FileOutputStream*, *DataInputStream*, *DataOutputStream*, *BufferedInputStream*, and *BufferedOutputStream* classes
i. Store and restore objects using *ObjectOutputStream* and *ObjectInputStream*
j. Implement the *Serializable* interface to make objects serializable
k. Read and write to and from a file using the *RandomAccessFile* class

7. Utilizing Generics

Upon completion of this unit the learner will be able to:

a. Explain the benefits of generics
b. Define and use generic classes and interfaces
c. Construct and invoke generic methods and bounded generic types
d. Develop a generic sort method to sort an array of *Comparable* objects
e. Explain the benefits of wildcard generic types

8. Analyzing Data Structures and Algorithms

Upon completion of this unit the learner will be able to:

a. Utilize common methods defined in the *Collection* interface for operating collections
b. Apply the *Iterator* interface to traverse the elements in a collection
c. Explore how and when to use *ArrayList* or *LinkedList* to store a list of elements
d. Distinguish between the *Vector* and *ArrayList* class, and use the *Stack* class for creating stacks
e. Store unordered, nonduplicate elements using a set
f. Compare and contrast the performance of sets and lists
g. Describe the differences between *Collection* and *Map*, and recognize when and how to use *HashMap*, *LinkedHashMap*, or *TreeMap* to store values associated with keys
h. Obtain singleton and unmodifiable sets, lists, and maps using the static methods in the *Collections* class
i. Analyze the time complexity of various sorting algorithms
j. Design, implement, and analyze bubble, merge, quick, heap, bucket, and radix sorting algorithms

9. Developing Multithreading and Parallel Programming

Upon completion of this unit the learner will be able to:

a. Develop task classes by implementing the *Runnable* interface
b. Create and control threads using the *Thread* class
c. Execute tasks in a thread pool
d. Synchronize threads using locks
e. Facilitate thread communications by invoking conditions on locks
f. Restrict the number of accesses to a shared resource using semaphores
g. Apply the resource-ordering technique to avoid deadlocks
h. Describe the life cycle of a thread
i. Create synchronized collections using the static methods in the *Collections* class
j. Develop parallel programs using the *Fork/Join Framework*

10. Building Network-Based Applications

Upon completion of this unit the learner will be able to:

a. Create servers using server sockets, and clients using client sockets
b. Implement Java networking programs using stream sockets
c. Develop a client/server application
d. Obtain Internet addresses using the *InetAddress* class
e. Implement servers for multiple clients
f. Develop applets that communicate with the server
g. Send and receive objects on a network

11. Implementing Database Applications with JDBC

Upon completion of this unit the learner will be able to:

a. Explore relational databases
b. Use SQL to create and drop tables, and to retrieve and modify data
c. Load a driver, connect to a database, execute statements, and process result sets using JDBC
d. Utilize prepared statements to execute precompiled SQL statements
e. Implement callable statements to execute stored SQL procedures and functions
f. Explore database metadata using the *DatabaseMetaData* and *ResultSetMetaData* interfaces

RYERSON ACADEMIC POLICIES:

Partial information on certain Ryerson University Senate policies is provided herein; For complete information on Ryerson`s academic policies, visit the Senate website at www.ryerson.ca/senate/policies.

Policy on Grading, Promotion, and Academic Standing No. 46
Student Code of Academic Conduct No. 60
Undergraduate Academic Consideration and Appeals Policy No. 134
Examination Policy No. 135
Course Management Policy No. 145
Accommodation of Student Religious Observance Obligations Policy No. 150

REFUND POLICY and OFFICIAL WITHDRAWAL FROM THE COURSE:

FULL REFUND: granted to registrants who officially withdraw at least five business days prior to the first scheduled class.
75% REFUND: granted to registrants who officially withdraw after the full refund deadline but 24 hours prior to the fourth scheduled class for courses of 42 or more hours in duration. No refunds will be granted after the fourth scheduled class.
GOOD ACADEMIC STANDING (no refund): granted to students who officially withdraw 24 hours before the eighth scheduled class meeting.
NON ATTENDANCE OF THE COURSE IS NOT CONSIDERED AN OFFICIAL WITHDRAWAL: Students must officially withdraw in person at Enrollment Services and Student Records or on line at www.my.ryerson.ca by the appropriate deadline.

MISSED TERM WORK OR EXAMINATIONS:

Exemption or deferral of a test, assignment, or final examination is not permitted except for documented medical, religious or compassionate grounds. The instructor must be notified by e-mail prior to the test and appropriate documentation submitted. For missed tests or coursework, the instructor may arrange a make-up test or re-weigh the course requirements depending on course policy. For a missed final exam, if the majority of the course work has been completed with a passing performance, and the documentation is acceptable, an INC grade will be entered by the instructor. An INC grade will not be granted if term work was missed or failed. The make-up examination must be written by a specified date within 3 months after the submission of the incomplete grade. Failure to do this will result in an F grade.

Medical: If possible, students must notify the instructor by e-mail/phone before the test/assignment/exam. The Student Medical Certificate form (www.ryerson.ca/senate/forms/medical/pdf) must be completed by the student and medical physician and presented to the instructor or Program Manager within 3 days of the missed piece of work.
Religious Observance: If you will require religious accommodation, you must complete and submit the Student Declaration of Religious Observance Accommodation form (www.ryerson.ca/senate/forms/relobservforminstr.pdf) and present it to the instructor or Program Manager **within the first two weeks of the semester**.
Compassionate grounds: The instructor/Program Manager will use discretion in considering compassionate grounds (such as a death in the family). Students must present reasonable and appropriate documentation to support their request. Business /or personal travel and/or normal employment commitments generally do not constitute grounds for academic consideration.

ACCESS CENTRE:

Students with existing Access Centre accommodations should meet with their instructor at the beginning of the course.

Students who register with the Access Centre during the semester must speak to their instructor as soon as their needs are identified.

ACADEMIC INTEGRITY:

Plagiarism and/or cheating are unacceptable. Details of the Student Code of Academic Conduct are provided at www.ryerson.ca/senate/policies/pol60.pdf. If you are unsure of what is acceptable, you should consult with your instructor.

PLAGIARISM:

The Ryerson Student Code of Academic Conduct defines plagiarism and the sanctions against students who plagiarize. All Chang School students are strongly encouraged to go to the academic integrity website www.ryerson.ca/academicintegrity and complete the tutorial on plagiarism.

The University has subscribed to the Turnitin service which helps instructors identify internet plagiarism and helps students maintain academic integrity. The work submitted by students in this course may be submitted to Turnitin. Students who do not want their work submitted to this plagiarism detection service must, by the end of the second class, consult with the instructor to make alternate arrangements.