BAKER COLLEGE
STUDENT LEARNING OUTCOMES

CS 2410 JAVA PROGRAMMING

3 Semester Hours

Student Learning Outcomes and Enabling Objectives

1. Construct Java™ technology applications that leverage the object-oriented features of the Java language, such as composition, encapsulation, inheritance, and polymorphism
    a. Illustrate how to write the code for a class that inherits another class and overrides one or more of its methods
    b. In a user-defined class, illustrate how to write code that overrides the toString method of the Object class to control the String object that's returned by this method
    c. In a user-defined class, illustrate how to write code that overrides the equals method of the Object class to control how the user-defined class tests for equality
    d. Demonstrate what it means for a subclass to extend a superclass
    e. Explain why the toString method and the equals method are available to all objects and when you might override these methods
    f. Describe what the @Override annotation does
    g. Describe the concept of encapsulation and explain its importance to object-oriented programming.
    h. Describe the access modifiers that you can use for the members of a class and how they facilitate encapsulation
    i. Explain what polymorphism is and how it works
2. Demonstrate their skill in selecting the appropriate Java flow control features to implement specific algorithms which provide problem solutions
    a. Illustrate how to code if/else statements and switch statements to control the logic of an application
    b. Illustrate how to code while, do-while, and for loops to control the repetitive processing of an application
    c. Illustrate how to code nested loops whenever they are required
    d. Illustrate how to use the break statement to jump out of a loop
    e. Illustrate how to use the continue statement to jump to the start of a loop
3. Execute a Java application to test correctness of solutions
    a. Demonstrate how to test and debug your applications.
    b. Demonstrate how to trace the execution of an application with println statements
    c. Demonstrate how to use an IDE to set breakpoints, step through code, inspect variables, and inspect the stack trace
    d. Distinguish between testing and debugging
    e. Distinguish between compile-time, runtime, and logic errors
4. Illustrate the proper use of Java data types and expressions

a. Illustrate how to code the instance variables, constructors, and methods of a class that defines an object
b. Illustrate how to write code that creates objects from a user-defined class and then uses the methods of the objects to perform tasks
c. Illustrate how to code a class that contains static fields and methods
d. Illustrate how to write code that calls static methods from a user-defined class
e. Illustrate how to write code that overloads a method
f. Differentiate between an object's identity and its state
g. Explain what a default constructor is and when the Java compiler automatically creates one
h. Explain what an access modifier is and how it affects how you can use the methods of a class from another class
i. Differentiate between a static method and an instance method
j. Differentiate between primitive types and reference types
k. List four ways you can use the this keyword within a class

5. Use arrays and Collections to represent data structures
    a. Illustrate how to write code that creates a one-dimensional array and stores a list of elements
    b. Illustrate how to write code that creates a multi-dimensional array and stores a table of elements
    c. Illustrate how to use enhanced for loops to work with the elements in an array
    d. Illustrate how to use the methods of the Arrays class to fill an array, sort an array, search an array for an element, copy an array, or check whether two arrays contain the same elements.
6. Implement error-handling techniques using exception handling
    a. Illustrate how to write code that uses a try or try-with-resources statement to handle exceptions
    b. Demonstrate how to use the methods of the Throwable class to get information about an exception
    c. Illustrate how to code a method that throws an exception to the calling method
    d. Describe three situations where you might want to throw an exception from a method
    e. Describe the classes in the Throwable hierarchy
    f. Explain how Java propagates exceptions, and how it uses the stack trace to determine what exception handler to use when an exception occurs
    g. Describe the order in which you code the catch clauses in a try statement
    h. Explain when the code in the finally clause of a try statement is executed, and what that code typically does
    i. Explain what it means to swallow an exception, and why that's almost never an acceptable practice
    j. Illustrate how to code a class that defines a custom exception and use that custom exception in an application
    k. Describe two situations where you might create a custom exception class
    l. Explain what exception chaining is and when you might use it

Big Ideas and Essential Questions

Big Ideas

- There are three things you must master:
  - How the language is structured (grammar)
  - How to name things you want to talk about (vocabulary), and
  - The customary and effective ways to say everyday things (usage)
- Learn how to structure your code so that
  - It works well
  - Other people can understand it
  - Future modifications and improvements are less likely to cause headaches

Essential Questions

1. What does it mean that Java is object-oriented, and how does that impact the way that you write Java programs?
2. What data structures, operations, and facilities are provided by the standard libraries?
3. What are the customary and effective ways to structure your code?
4. How is the language best put to use in practice?

These SLOs are approved for experiential credit.

Effective: Fall 2017