CST8116 Lab Exam 01 (21F) Section 3xx

Lab Exam

# Instructions - Overview

* Follow the instructions starting on page 2 of this handout to design a problem solution, create a Java program, and test your program.
* This exam must be completed and submitted by the end of your lab period.
* Note: Each lab section will be given a unique problem statement to help detect cheating.

# Instructions – Submission Notes

* Submit your paper work as well as your Java source code files (.java files) to Brightspace by the end of the lab period.
* Your lab professor may specify additional submission requirements in the lab period.

# Instructions – Open Book Exam

* The first part of this exam is closed book and done on paper. Once you have written a UML diagram or pseudocode or flowchart to your lab professor’s satisfaction (60-70%) complete, you can open your computer and complete the code.
* Once you start coding, this is an open book lab exam, you may use your textbooks, lecture notes, exercises, assignments, hybrids for help.
* You are not permitted to talk to, or chat with other students during the lab exam.
* You are not permitted to talk about, or share materials from, your lab exam, with students in other lab sections.

# Part 1 ON PAPER Understand the Problem and Class Design

* The client wants a program which will count the number of Toonies ($2), Loonies ($1), quarters($0.25), dimes ($.10), and ($.05) nickels and compute the total amount. The program needs to store the total number of each kind of coin. Design a class based on this word problem.
* Testing values are: 2 toonies, 4 loonies, 1 quarter, 3 dimes and 5 nickels.
* Using the word problem above, design a class with the needed fields, one no parameter constructor, and needed methods. Document your design using a detailed UML Class diagram or pseudocode or flowchart. You should aim to have as complete a solution as possible but it is ok if parts are missing. Aim for 60-70% complete.
* Have the professor sign off on your design
* Note: The calculations used in this word problem are fictional.

# Part 2 WITH PROFESSOR OK - Write the Java code for the class

* Follow your UML class diagram or pseudocode or flowchart and code the class you designed.
* Use Java coding conventions for identifiers: class, constructor, field, variable, and method names
* Mark fields private, constructor and methods public.
* Select and use appropriate data types for the fields as well as method parameters as needed.
* Comment your code as requested and demonstrated in the course.

# Part 3 Test the class you created, using a main method in a separate class.

* Create a second class with an identifier that follows these instructions:
  + *FullName*LabExam1Section*Number*
  + For example, a student with first name “Abc” and last name “Xyz” in section 123 would name the class as:

AbcXyzLabExam1Section123, with resulting file name AbcXyzLabExam1Section123.java

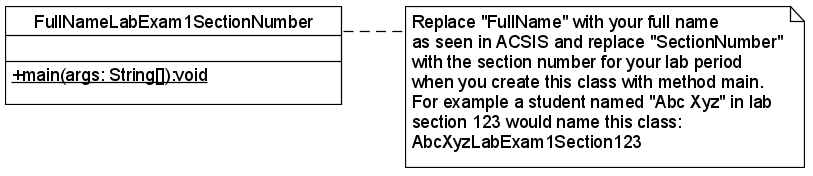
* Using this class with a method main, write some simple program tests to verify your class.
  + Instantiate one object based on your designed class using the no parameter constructor, set a value into each field, get a value from each field printing the set value(s) on screen followed by the retrieved value(s) on screen, and show the result returned from the worker method as well.
  + There is no need for user input to the program, use literal values in the source code of your test program (within method main). Ensure that your full name is written out on screen after the tests above are run.
* Use the testing values given above as part of the word problem.

## PART 4 Upload Your Work

Upload your 2 .java files as well as a screenshot of the running program

Submit your paper to your professor

# UML Class Diagram for testing class



# Sample of expected program output (your program should be similar)

Change is 2 toonies, 4 loonies, 1 quarter, 3 dimes and 5 nickels

2

4

1

3

5

Testing work method: 8.8

Program by Abc Xyz

# Grading (7 Points)

|  |  |  |  |
| --- | --- | --- | --- |
| Criteria | Missing / Poor (0) | Below Expectations (0.5) | Meets Expectations (1) |
| Screen shot executing program | Missing or incorrect or does not test the specified class. | Program shows output of some of the requested tests, program may not run to completion or crashes and / or student full name does not appear as part of the output. | Demonstrates the running program, program runs to completion with expected test output as well as student’s full name. |
| .java files: comments | Missing or incorrect. | .java files contain some programmer comments but missing information or incomplete. | .java files contain comment header with format specified for course. Each class method has a comment above the header with a brief statement about the method. |
| .java files: syntax | Missing or incorrect. | Class file(s) may have small syntax mistakes that would prevent them from compiling | Class file(s) do not have syntax mistakes and should compile. |
| .java files: conventions | Missing or incorrect. | Java coding conventions for identifiers, indentation are not fully followed. | Java coding conventions for identifiers and indentation are fully followed. |
| .java files: Constructor, Gets, Sets | Missing or incorrect. | Logic for the specified class, including accessors, mutators, no parameter constructor, may have small logic mistakes. | Logic for the specified class, including accessors, mutators, no parameter constructor, have no logic mistakes. |
| .java files: Worker Method | Missing or incorrect. | Logic for the specified class worker method may have small logic mistakes. | Logic for the specified class worker method has no logic mistakes. |
| .java files: Testing class with main | Missing or incorrect. | Class does not have specified identifier, i.e. student full name, test name, and lab section number.  and / or  Method main has logic mistakes, i.e. missing all or part of: instantiate specified class with no parameter constructor, test mutators with specified sample data, and test accessors and work method via program outputs, and write out Students full name. | Class does have specified identifier, i.e. student full name and lab section number.  and  Method main has no logic mistakes, i.e. has all of instantiate specified class with no parameter constructor, test mutators with specified sample data, and test accessors and work method via program outputs, and write out Students full name. |