**CIS 163 Project 1**

**A Change Jar program**

**Due Date**

* At the beginning of the lab; see the schedule, last page of the syllabus.

**Before Starting the Project**

* Review Chapter 1 – 6 of the CIS163 book
* Read this entire project description before starting

**Why are we doing this project.**

* This is a review project from CIS162. Re-discovering the basics of Java.

**Learning Objectives**

After completing this project you should be able to:

* *have a good working knowledge of the topics covered in CIS162*
* *create classes with associated methods*
* *use complex* if statements
* *read and write data* from external text files
* *use* static methods and properties available in the Java library

**You must complete each step fully before proceeding on. No credit is given to any given step unless the previous steps have been completed and is functioning!**

**Before you turn in your work: use the** [**Java Style Guide**](http://www.cis.gvsu.edu/studentsupport/javaguide) **to document your project. (10 pts)**

**Step 1: Create an IntelliJ project named “ChangeJarPrj”**

* Create a package named: changeJarPack (right click on “ChangeJarPrj” and select new/package)
* Create a class named: ChangeJar (right click on “changeJarPack” and select new/class)
  + The properties and methods for this class are in step 2.
* Create a JUnit Test Case named: ChangeJarTest (Right click on ChangeJar, if you have questions, see the instructor)
  + Log on to BB and cut and paste the file found in the project 1 folder under Course Documents.

**Step 2: Implement the following methods for the class “ChangeJar”:**

Implement the following methods and properties in ChangeJar class. For properties, you will need four instance variables: quarters (integer), dimes (integer), nickels (integer), pennies (integer). There are no dollars bills (or dollar coins) in this ChangeJar. For methods, you will need to implement the following (include any setters or getters that are needed). Unless otherwise stated, you can assume the input has no errors for step 2 (i.e., a valid set of numbers) contained within (later steps this does not apply).

* public ChangeJar() Default constructor that sets the ChangeJar to zero.
* public ChangeJar(double amount) A constructor that initializes the instance variables with the provided value converted to quarters, dimes, nickels, and pennies. For example, if amount was 1.34 then you would have 5 quarters, 1 nickel, 4 pennies.
* public ChangeJar (ChangeJar other) A constructor that initializes the instance variables with the other ChangeJar parameter.
* public ChangeJar(String amount) A constructor that accepts a string as a parameter with the provided value converted to quarters, dimes, nickels, and pennies. For example, if amount was “1.34” then you would have 5 quarters, 1 nickel, 4 pennies. More examples:
  + “1.3” is 5 quarters, 1 nickel
  + “1.34” is 5 quarters, 1 nickel, 4 pennies
  + “0.34” is 34 cents; 1 quarter, 1 nickel, 4 pennies
  + “0.04” is 4 cents
  + “500.” is not allowed, instead “500.0” is used.
* public boolean equals(Object other) A method that returns true if “this” ChangeJar object is exactly the same (in terms of the amount in the ChangeJar) as the other object (Note: you must cast the other object as a ChangeJar object).
* public static boolean equals(ChangeJar jar1, ChangeJar jar2) A static method that returns true if ChangeJar object jar1 is exactly the same (in terms of amount in the ChangeJar) as if ChangeJar object jar2.
* public int compareTo(ChangeJar other) A method that returns 1 if “this” ChangeJar object is greater than (based upon the total in the ChargeJar) the other ChangeJar object; returns -1 if the “this” ChangeJar object is less than the other ChangeJar; returns 0 if the “this” ChangeJar object is equal to the other ChangeJar object.
* public static int compareTo(ChangeJar jar1, ChangeJar jar2) A method that returns 1 if ChangeJar object jar1 is greater (in terms of the amount in the ChangeJar) than ChangeJar object jar2; returns -1 if the ChangeJar object jar1 is less than ChangeJar jar2; returns 0 if the ChangeJar object jar1 is equal to ChangeJar object jar2.
* public void subtract(int quarters, int dimes, int nickels, int pennies) A method that subtracts the parameters from the “this” ChangeJar object. You may assume all of the parameter are positive.
* public void subtract (ChangeJar other) A method that subtracts ChangeJar other to the “this” ChangeJar object. (For step 2 there are no worries about errors)
* public void dec() A method that decrements the “this” ChangeJar by 1 penny.
* public void add(int quarters, int dimes, int nickels, int pennies) A method that adds the parameters from the “this” ChangeJar object. You may assume all of the parameter are positive.
* public void add (ChangeJar other) A method that add ChangeJar other to the “this” ChangeJar object.
* public void inc() A method that increments the “this” ChangeJar by 1 penny.
* public String toString() Method that returns a string that represents a ChangeJar with the following format: “10 quarters, 1 dime, 0 nickels, 1 penny”. **Be sure to use proper pluralization**. For example, 1 penny or 2 pennies.

**Step 3: Software Testing: Using a JUnit named “TestChangeJar”**

Software developers must plan from the beginning that their solution is correct.

* Within this file you will see comments on where to place the JUnit test cases.

**Step 4: Software Testing: Using a main program**

A start to a main program has been provided (see below) and your task is to add on **many** Java statements that would test each method separately and completely.

**Main Method**

**public** **static** **void** main (String[] args) {

ChangeJar s = **new** ChangeJar("2.82");

System.*out*.println("2.82 Amount: \n" + s);

s = **new** ChangeJar("8");

System.*out*.println("8 Amount: \n" + s);

s = **new** ChangeJar(".28");

System.*out*.println(".28 Amount: \n" + s);

ChangeJar s1 = **new** ChangeJar();

System.*out*.println("0 Amount: \n" + s1);

s1.add(1,1,1,100);

System.*out*.println("1,1,1,100 Amount: \n" + s1);

ChangeJar s2 = **new** ChangeJar(41.99);

s2.add(0,0,0,99);

**for** (**int** i = 0; i < 100; i++)

s2.dec();

System.*out*.println("amount: \n" + s2);

**// ADD more test cases here**

**}**

**Sample Results**  
2.82 Amount:

11 Quarters

0 Dimes

1 Nickel

2 pennies

8 Amount:

32 Quarters

0 Dimes

0 Nickels

0 pennies

.28 Amount:

1 Quarter

0 Dimes

0 Nickels

3 pennies

0 Amount:

0 Quarters

0 Dimes

0 Nickels

0 pennies

1,1,1,100 Amount:

1 Quarter

1 Dime

1 Nickel

100 pennies

amount:

167 Quarters

2 Dimes

0 Nickels

3 pennies

**Now, compare Step 3’s approach to testing to Step 4’s approach. Is one better than the other? Also, remember to use the Java Style Guide for documenting your program.**

**Step 5: Create the following additional methods in the ChangeJar class:**

* public void save(String fileName)A method that saves the “this” ChangeJar to a file; use the parameter filename for the name of the file.
* public void load(String fileName)A method that loads the “this” ChangeJar object from a file; use the parameter filename for the name of the file.
* public ChangeJar takeOut (double amount) Take out amount using the available quarters, dimes, nickels and pennies. For example: 1.23 would be 4 quarters, 2 dimes, 3 pennies. Example 2. The ChangeJar holds 4 quarters, 3 dimes, 0 nickels, 3 pennies; now takeout(.80) which would be 2 quarters and 3 dimes.
  + This sounds easy, it is not.
* public static void Mutate(Boolean on) A method that turns ‘off’ (false) and ‘on’ (true) any subtract/add methods in ChangeJar based on the value (true,false) of the variable ‘on’. In other words, when on is false, this prevents any subtract/add method from changing (mutate) the state of the “this” object as it relates to the amount in the ChangeJar.
* A change from step 2; allow for an error in the input arguments for **all** constructors and methods, and throw an IllegalArgumentException exception if an error occurs. For example, “-1.23” is not a valid input string for a constructor in step 2 and an exception is thrown.
  + Sample code snippet many help:

if (amount < 0)

throw new IllegalArgumentException();

* + Other errors examples for the constructor that accepts a string:
    - “12.123”: to many decimal points
    - “abc”: must contain only digits

**Following will help you with reading and writing to a file:**

The data file is shown directly below and contains only one line of data. Listing 6.30 of your book shows the use of the Scanner Class.

ANCHORAGE 256000

**Here is the code that would read the above file:**

**public** **void** sampleReadData(String fileName){

String city;

**int** population;

**try**{

// open the data file

Scanner fileReader = **new** Scanner(**new** File(fileName));

// read one String in of data and an int

city = fileReader.next();

population = fileReader.nextInt();

System.*out*.println (city + " " + population);

}

// could not find file

**catch**(FileNotFoundException error) {

System.*out*.println("File not found ");

}

// problem reading the file

**catch**(IOException error){

System.*out*.println("Oops! Something went wrong.");

}

}

**Here is the code that would write the above file:**

**public** **void** sampleWriteData (String fileName) {

PrintWriter out = **null**;

**try** {

out = **new** PrintWriter(**new** BufferedWriter(**new** FileWriter(fileName)));

}

**catch** (IOException e) {

e.printStackTrace();

}

String s = "ANCHORAGE";

out.println(s + " " + "256000");

out.close();

}

**Step 6: Software Testing: Complete the second part of the JUnit class named: TestChangeJar.**

* BE SURE to include JUnits test cases that show your ChangeJar is properly throwing exceptions from step 5. **Note: one JUnit test per exception!**
* Must have 100% coverage and MORE. (see the instructor regarding “MORE”)! Robust testing.
* Ask the instructor for what Robust testing is.

**Step 7: Challenge Requirement, read chapter 6 in your book.**

* The following should only be attempted after all of the other requirements have been completed.
* Create a GUI front end to your project and create 3 ChangeJars with associated JButtons.
* For full credit, call using JButtons, or any other JComponent so ALL of the methods you created in your ChangeJar class (i.e., load, save, inc, etc) are called from your GUI.
* Also have a JButton that calls the mutate(Boolean on)method (only one JButton is need for all ChangeJars).
* Show the values for quarters, dimes, nickels and pennies on the display for each ChangeJar.

Create a class called “MyChangeJarPanel” that has a private inner class that implements ActionListener, see chapter 6 in your book; here is some help.

public class MyChangeJarPanel extends JPanel {

private ChangeJar changeJar;

;

…..

public MyChangeJarPanel () {

changeJar = new ChangeJar(“1.23”);

….

}

private class ButtonListener implements ActionListener {

public void actionPerformed(ActionEvent e) {

…..

// a starting GUI has been provided.

--------------------------- YOUR’RE DONE ☺ -------------------------------

**Project 1: “ChangeJar” Program Rubric.**

|  |  |
| --- | --- |
| Student Name |  |
| Due Date |  |
| Date Submitted, Days Late, Late Penalty |  |

|  |  |  |
| --- | --- | --- |
| **Graded Item** | **Points** | **Comments and Points Secured** |
| Javadoc Comments and Coding Style/Technique  (<http://www.cis.gvsu.edu/studentsupport/javaguide>)   * Code Indentation (auto format source code in IDE) * Naming Conventions (see Java style guide) * Proper access modifiers for fields and methods * Use of helper (private) methods * Using good variable names * Header/class comments * Every method uses @param and @return * Every method uses a /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* separator * Overall layout, readability, No text wrap * Using /\*\* … / for each Instance variable * Has many inner “inner” comments | 10 |  |
| **Steps 1 – 2: Basic Functionality**   * public ChangeJar() * public ChangeJar(double amount) * public ChangeJar (ChangeJar other) * public ChangeJar(String amount) * public boolean equals(Object other) * public static boolean equals(ChangeJar jar1, ChangeJar jar2). * public int compareTo(ChangeJar other) * public static int compareTo(ChangeJar jar1, ChangeJar jar2) * public void subtract(int quarters, int dimes, int nickels, int pennies) * public void subtract (ChangeJar other) * public void dec() * public void add(int quarters, int dimes, int nickels, int pennies) * public void add (ChangeJar other) * public void inc() * public String toString() | 30 |  |
| **Step 3: JUnit test (basic)** | 5 |  |
| **Step 4: Main test** | 5 |  |
| **Step 5: Added functionality**   * public void save(String fileName) * public void load(String fileName) * **public ChangeJar takeOut (double amount)**   + **any error -5 pts** * public static void Mutate(Boolean on) * Allow for an error in the input for all constructors and methods | 1  4  10  5  10 |  |
| **Step 6: More Software Testing (advance)** | 10 |  |
| **Step 7: Challenge Requirement** | 10 |  |
| **Total** | **100** |  |

**Additional Comments: (How many hours did you spend developing this project?)**