605.201 Mini-Project 2:

Please do the following to complete this assignment.

Purpose:

The purpose of this project is to provide non-trivial practice in the use of Java object-oriented programming features to implement an object-oriented design and have a bit of fun doing it.

Resources Needed:

You will need a computer system with Java 7 or greater SE edition run-time and Java Development Kit (JDK). You may optionally use a Java IDE for example NetBeans, Eclipse, etc. However application builders are not allowed.

Submitted Files:

Design and Analysis:

This is an informal essay-style single-spaced word-processed document. The file formats accepted will be announced at project assignment. The length of the document should be between 1 and 1.5 pages. The following subjects should be discussed in this order:

1. General program design. How is the program organized? What major data structures were used? How did you divide the functionality among your classes? How are commands processed? Etc.
2. What alternative approaches were considered and why were they rejected?
3. What did you learn from doing this project and what would you do differently?

Source files:

Each public class *must* be contained in a separate Java source file. Only one source file will have a main() method and this source will be named **VendingMachineSimulator.java**. Other source/class names are up to you following the guidelines specified so far in the course.

The format of the Java source must meet the general Java coding style guidelines discussed so far during the course. Pay special attention to naming guidelines, use of appropriate variable names and types, variable scope (public, private, protected, etc.), indentation, and comments. Classes and methods should be commented with JavaDoc-style comments (see below). Please use course office hours or contact the instructor directly if there are any coding style questions.

JavaDocs:

Sources should be commented using JavaDoc-style comments for classes and methods. Each class should have a short comment on what it represents and use the @author annotation. Methods should have a short (usually 1 short sentence) description of what the results are of calling it. Parameters and returns should be documented with the @param and @return annotations respectively with a short comment on each.

JavaDocs must be generated against every project Java source file. They should be generated with a -**private** option (to document all protection-level classes) and a –d [dir] option to place the resulting files in a **javadocs** directory/folder at the same level as your source files. See the JavaDocs demonstration for more details.

Submit file:

The submit file is to be a Zip file containing your design and analysis document, your Java sources, and your javadocs directory/folder. Any appropriate file name for this Zip file is acceptable.

If you know how to create a standard Java JAR file, this is also acceptable for your source code. However, make sure you include the source code in your JAR file.

Collaboration:

It is encouraged to discuss technical or small design parts of this project with your fellow students. However the resulting design and implementation must be your own. For example, it is acceptable to discuss different ways of maintaining the vending machine state but not detailed design or implementation information on processing the purchase command. When in doubt, ask during office hours or contact your instructor.

Program Specification:

1. Create a new multi-class Java program which implements a vending machine simulator which contains the following functionality:
   A) At program startup, the vending machine is loaded with a variety of products in a variety of packaging for example soda/tonic/Coke in bottles, peanuts in bags, juice in cartons, etc. Also included is the cost of each item. The program should be designed to easily load a different set of products easily (for example, from a file).

      Also at program startup, money should be loaded into the vending machine. Money should consist of different monetary objects for the specified currency for example $1 bills, $5 bills, quarters, dimes, etc. Your program should be designed to use different national currencies easily (for example the Euro). Money should be maintained as paper bills and coins, not just amounts.

   B) A menu of commands must be provided. At a minimum the menu should consists of the following commands:
      1. Display the list of commands
      2. Display the vending machine inventory. For each item, this command should result in displaying a description and current quantity.
      3. Display the money currently held in the vending machine.
      4. Purchase an item. The result of this selection should be the following actions:
         1. Prompt the user to indicate what item to purchase
         2. Prompt the user to specify what monetary items are being used for payment (the actual items for example quarters, dimes, etc.), not a money amount
         3. If the user specified enough money to purchase the selected item, the item is purchased (deducted from inventory), supplied money is added to the vending machine, and any change is returned in the form of monetary items (quarters, dimes, etc.).

4. If the user did not specify enough money for the selected item, the transaction is aborted with the supplied money not added to the machine (not accepted) and the product not purchased (i.e. the state of the vending machine is unchanged).
5. Exit – exits the program displaying a departing message.

2. Additional points to consider:
   A) You can use the Java Standard Edition (SE) API library as supplied by Oracle (AKA Sun) except the collection classes other than String and standard arrays (i.e. not ArrayList, Map, Vector, etc.). These other collections will be covered later in the course.
   B) When developing complex classes, consider creating a main() method to test them out. Once tested successfully, delete the main() method.
   C) You should generate error messages when appropriate, for example on invalid input values or not enough money supplied for the selected item to purchase. Exceptions will be covered later in the course so for this program displaying appropriate messages on the console is fine.
   D) Code to input data from the console will be supplied. Java I/O programming will be covered later in the course.

Other Activates:

1. Observe the presentation on JavaDocs.

2. Observe the Vending Machine Simulator demonstration for an example of one implementation.

3. Create a compressed zipped folder containing your Design and Analysis document, your Java source code files, and your javadocs folder.

4. Submit your compressed zipped folder as directed by your instructor.

Assignment Rubric:

Part	**70%**	**80%**	**90%**	**100%**	**% of Grade**
Design and Analysis Document	All but one subject addressed with relevant, information. Few minor typographical issues. Document is close to assigned length	All assigned subjects address with mostly relevant information. Nicely formatted document. Document is close to assigned length	All assigned subjects address with accurate and relevant. Nicely formatted document. Document is within assigned length	All assigned subjects address with accurate, relevant, and insightful information. Very nicely formatted. Document is within assigned length	15%
Functionality	Majority of required function parts work as indicted in the assignment text. One major or 3	Most required function parts work as indicted in the assignment text above and	Nearly all required function parts work as indicted in the assignment text above and submitted documentation. One	All required function parts work as indicted in the assignment text above and submitted	40%

	minor defects. All major functionality at least partially working (example change provided but not correct). Design document does not fully reflect functionality.	submitted documentation. One major or 3 minor defects. All major functionality at least partially working ((example change provided but not correct).	to two minor defects.	documentation.	
Code	Majority of the code conforms to coding standards as explained and demonstrated so far in the course (ex. method design, naming, formatting, etc.). Five to six minor coding standard violations. Some useful comments. Some JavaDocs commenting. Code compiles with multiple warnings or fails to compile with difficult to diagnose error.	Most of the code conforms to coding standards as explained and demonstrated so far in the course (ex. method design, naming, formatting, etc.). Three to four minor coding standard violations. Mostly useful comments. Public class JavaDocs complete. Code compiles with one to two warnings.	Almost all code conforms to coding standards as explained and demonstrated so far in the course (ex. method design, naming, formatting, etc.). One to two minor coding standard violations. Appropriate level of useful comments. Public class JavaDocs complete. Code compiles. Code compiles with no errors or warnings.	All code conforms to coding standards as explained and demonstrated so far in the course (ex. method design, naming, formatting, etc.). Appropriate level of useful comments. Complete JavaDocs as specified. Code compiles with no errors or warnings.	35%
Submit package	More than one file submitted in incorrect format. Files not enclosed in the specified compressed file.	All but one file submitted in correct format. Files not enclosed in the specified compressed file.	All file submitted in correct format but not in the specified compressed file.	All file submitted in correct file formats and compressed as specified.	10%