## Project 3 – Group Project – Bicycle Parts Distributorship

This project is the third of three connected projects.  The first two are individual projects.  The third is a group project. Your group has several implementations of Project 2.  Your group should augment the design and code from one (or more) of the Project 2 implementations to include the new capabilities described in this project description.

NOTE: The Bicycle Parts Distributorship project specifications use the term database as a general description of the underlying data that must be persistent across executions of your program.  To satisfy the persistent requirement, your database will have to be a file (or files) that your program reads/writes.  You do not have to create an SQL or NOSQL database.

### Project Tips from Previous Students

* You should emphasize the importance of good time management. **Start early!** When the group project is assigned, you may think there is plenty of time. Do not delay, thinking you will catchup in a couple of weeks.
* I recommend that teams perform an agile software development process, which is a sequence of sprints where each sprint incrementally adds code to your project. Divide the software capabilities into small related collections, which can be incrementally developed. You begin by developing the first collection of capabilities, where development includes testing. Then you develop each subsequent collection of capabilities. At a minimum, you should push your code to GitHub after each sprint.
* Several of our classes have a portion of time dedicated to project group work; however, you should plan to meet outside of class. These meetings can be in-person or via Skype/Facetime. The most successful teams are those that communicate a lot, and hold each other accountable.
* Usually, the most successful teams meet in-person and perform pair/partner programming with ongoing discussions. This style of development results in less hassle trying to integrate your code in the end and if you get stuck you have group members right there to work through it.
* If you divvy your programming into three chunks with minimal group meetings, integrating the separate chunks into a working program will be challenging. You should plan for at least three days to all the code together at the end.
* Have multiple people contribute to GUI development. Do not assign it to a one person.
* Using Serializable object I/O is a convenient way to read/write your integral data structures to a “database” file.

### Tips for Successfully Working in a Group

The following tips for successfully working are an edited version of those found in [http://www.cs.cmu.edu/~pausch/Randy/tipoForGroups.html.](http://www.cs.cmu.edu/~pausch/Randy/tipoForGroups.html)

* **Meet people properly**.  Introduce each other. Make sure you can pronounce each other’s names.  Exchange contact information (email, phone numbers) and establish acceptable times for contact each other.
* **Find things in common**.   Discover something in common with each. Having common interests will make it easier to address issues where you have differences.  At a minimum, you like computer science and experience the same weather. Maybe you like the same sports or music or movies.
* **Establishing Meetings** – Ensure everyone knows meeting dates and are accountable for being prepared. This can be done by email.  Do not assume roommates deliver phone messages.
* **Electronic Communication** – Include all team members on electronic communication. This ensures everyone is kept up-to-date.
* **Written Records.**  Keep a record of tasks and who is responsible for what and by when.
* **Hold Meetings in Nice Places**.  Meet in a comfortable area (e.g., HCC or Trinkle lab) with appropriate internet connection with surfaces to accommodate laptops and/or notebooks. Long meetings may need snacks or coffee.
* **Let Everyone Talk**.  Listen when someone is talking, even if you do not like what they are saying.  Cutting someone off is rude, and not worth whatever small time gain you might make.  Do not finish someone’s sentences.  And remember: talking louder or faster does not make your idea any better.
* **Check Egos at Door**.  Record ideas with a descriptive label and not the originator, e.g., “the troll bridge story,” not “Jane’s story.”
* **Complement Each Other**.  Focus on the good, complement it, and then raise any objections or concerns you have about the rest of it.
* **Be Open and Honest**.  Discuss problems with group members, and ask me if you need help.  Be forgiving of mistakes, but address issues when they come arise.
* **Avoid Conflict**. When stress occurs and tempers flare, take a short break, clear your heads, apologize, and start again.  Apologize for upsetting your teammates, even if you think someone else was at fault.
* **Phrase alternatives as questions**.  When your group is considering option A, it is better for you to say, “*What if we tried B?*” than to say “*I think we should do B.*”

### Effective Meetings

Your group will meet to analyze the project, create the specification for Writing 3, create a design, implement code, test the system, and develop your presentation. Meetings are most effective when the adhere to the following simple rules.

* Establish an agenda for the meeting – the agenda lists the objectives of the meeting and keeps the meeting on focus. Any actions from previous meetings should be on the agenda.
* Start on time and end on time.
* End the meeting with actions – An action should have the following attributes.
  + A description of the action
  + The person(s) to whom the action is assigned.
  + The date the action is to be completed.

### Tools for Collaboration

* Trello.com – allows you to define tasks, assign to individuals, and track using features such as backlog, in-progress, and completed.
* Slack.com – allows groups to send messages to all in the group.
* Google Docs – allows groups to share documents.
* Phone Calls or Text Messages

### Introduction

You are a programmer working for Bicycle Parts Distributorship.  BPD has a warehouse full of bicycle parts.  The code and underlying data structures from Projects 1 and 2 apply to Project 3, but you will have to modify them.  In this increment of the project BPD has identified several users of the software.  There is an office manager who keeps track of many business aspects, there is a warehouse manager, and there is one sales associate in each sales van.  In your program, these employees (office manager, warehouse manager, and sales associates) have accounts that allow them to perform actions that are specific to their duties.  The duties that each type of employee performs are described in various use case scenarios.  Additionally, there is a system administrator who manages accounts.

### System Administrator Use Case Scenarios

A system administrator must be logged in to perform use case scenarios described in the following subsections.

The system administrator is responsible for creating accounts, deleting accounts, and resetting passwords.    An account shall include such things as user, phone number, email, username, and password.  Your program shall have default system administrator account that has the following.

* Username: admin
* Password: minda

When a system administrator creates an account, the initial login credentials shall be as specified by the GUI panel, which will allow various information such as the following.

* First Name
* Last Name
* email
* Username
* Password

**Optional**: Users shall have the ability to change their password and user information such as phone number and email; however, users cannot change their usernames.

#### Add Office Manager Use Case Scenario

The system administrator shall add an office manager account to the system.

#### Add Main Warehouse Manager Use Case Scenario

The system administrator shall add a warehouse manager account to the system.

#### Add Sales Associate and Sales Van Use Case Scenario

Project 2 allowed you to add a sales van.  In Project 3, the system administrator shall be responsible for adding a sales associate and a sales van.  This connects a sales associate to a sales van.  For example, Betsy Soho has SalesVanA.  This use case scenario establishes a sales associate account for each sales associate along with the sales van for the sales associate.

### Office Manager Use Case Scenarios

The office manager has several use case scenarios described in the following subsections.  An office manager must be logged in to perform these scenarios.

#### Examine Part Use Case Scenario

The office manager shall have the ability to examine the information for a part by the following.

* Enter a part name.
  + **Optional**: Include a wild card as part of the match. The wild card allows matching parts that are similar.  For example, you may choose to allow sa\* to match saddle and saw.
* Enter a part number.
* **Optional**: Enter a quantity.  This examine request allows the office manager to include less than, equal to, or greater than, where the display shows all parts that satisfy the relational operator applied to the quantity.  For example, the office manager may use this option to display all parts that have more than 10 in all warehouses.

#### Order Parts Use Case Scenario

Each bicycle part shall have a minimum amount that BPD warehouse wants to always have in stock.  When the total stock falls below the minimum amount, the office manager is notified to generate an order to refill the part.  Since it is inefficient to order one part, the office manager shall have the ability query the warehouse inventory to determine all parts that are either below their minimum or close to the minimum.  The office manager shall use below minimum notifications and the inventory order query to determine how many new parts to order.

#### Sales Commission Use Case Scenario

Sales associates are paid on commission - they receive a salary that is 15% of their total sales.  The office manager shall have the ability to generate a pay check for each sales associate.  The office manager shall input a sales associate, a start date, and an end date.  The output shall be the amount to be paid to the sales associate for the period selected.

### Warehouse Manager Use Case Scenario

The warehouse manager has several use case scenarios described in the following subsections.  An office manager must be logged in to perform these scenarios.

#### Warehouse Inventory Update Use Case Scenario

The warehouse manager shall be responsible for performing the Warehouse Parts Inventory Update as described in Projects 1 and 2.

#### Examine Part Use Case Scenario

The office manager shall have the ability to examine the information for a part by the following.

* Enter a part name.
  + **Optional**: Include a wild card as part of the match. The wild card allows matching parts that are similar.  For example, you may choose to allow sa\* to match saddle and saw.
* Enter a part number.
* **Optional**: Enter a quantity.  This examine request allows the office manager to include less than, equal to, or greater than, where the display shows all parts that satisfy the relational operator applied to the quantity.  For example, the office manager may use this option to display all parts that have more than 10 in all warehouses.

### Sales Associate Use Case Scenario

Sales associates have several use case scenarios described in the following subsections.  A sales associate must be logged in to perform these use case scenarios.

#### Sales Van Loading Use Case Scenario

A sales associate must keep appropriate parts on their sales van to keep their bike shop customers adequately stocked.  A sales associate generates a sales van delivery file that is used when loading parts onto their sales van.  The requirements for this use case scenario are as described in Project 2, Section Sales Van Parts Inventory Update from Main Warehouse and Section Sales Van Parts Inventory Update from a Sales Van; however, for this increment each sales associate can only load their own sales van.

#### Sales Invoice Use Case Scenario

When a sales associate distributes parts to a bike shop, the sales associate selects that parts to be sold.  The sales associate shall select a part name (or number) and the quantity be sold.  The sales associate shall select as many parts as the bike shop wants to purchase.  The sales associate shall generate a sales invoice that is a list of parts sold that includes the name of the bike shop employee who received the parts.  The invoice shall show the part name, part number, the number of parts sold, the list price, the sales price, and the total cost.  The invoices shall be stored in the BPD database. The office manager shall use the invoices to compute the commission of each sales associate. The following is an example invoice.

Sales Invoice for Susan’s Bicycle Shop, September 21, 2017 at 10:15AM

Part Name            Part Number  Price  Sales Price Qnty    Total Cost

WTB\_saddle           1234567890   33.00        33.00   3        99.00

26inTube             1234567891    7.00         5.58   1         5.58

seatPost             1234567891   17.00        15.21   2        30.42

carbonHandleBars42cm 1234567891   47.00        47.00   1        47.00

Total                                                          182.00

Received By Signature: Susan the Owner

### Program Functional Requirements

Your program shall accomplish the following.

1. Your program shall have a warehouse database.  The first time your program runs the warehouse database shall be empty or nonexistent.
2. Your program shall read the warehouse database upon starting.  This shall establish the initial inventory in your internal data structures.
3. Your program shall write the internal data structures to the warehouse database upon exiting.  The inventory in your internal data structures shall be saved such that upon running your program again, it starts with the same data as it ended with.
4. Your program shall have accounts for several types of users. Each user shall be logged into the system to perform their actions.
   1. An administrative account, which shall perform the system administrator use case scenarios.
   2. An office manager account, which shall perform the office manager use case scenarios.
   3. A warehouse manager account, which shall perform the warehouse manager use case scenarios.
   4. Several sales associate accounts, which shall perform the sales associate use case scenarios.

### Object Oriented Requirements

The last section of this document provides a class diagram of Gusty’s solution. You are welcome to use any (or all) of the design.

1. Your program shall at least two Java packages
2. Your program shall define and use a Java Interface as part of its solution.
3. Your program shall define and use inheritance as part of its solution.
4. Your program shall use at least two of the patterns we studied.
5. Your program shall include JUnit testing for at least two of its classes.
6. You shall create the following UML constructs.
7. Use case diagram that shows all use cases. You can reuse work from Lab 3
8. Class diagrams – The class diagrams do not have to include fields and methods. I want a diagram that shows the relationships between your classes.

### User Interface Requirements

1. Your program shall have a JavaFX Graphical User Interface.  You may create the layout of the GUI panels.
2. When a user is logged in, the user only has access to features allowed for their role as described in the use case scenarios above.  For example, a sales associate cannot generate their sales commission because that is a feature of an office manager account.

### Group Project Progress Check

To earn the points for this check, you need to make a substantial contribution to the code on your group's GitHub repository before this date.

#### What to Submit for Group Progress Check

Submit a paragraph describing your code contribution and a link to the GitHub repository for the group.

### Individual Reflection and Peer Review

For the reflection, write a paper reflecting on your experience with the group project.  The paper shall include the following.

* A paragraph (or two) describing your contributions to the project.  Highlight anything significant that you did to help with the project (technical or non-technical). Be sure to provide implementation details about the most significant contribution that you made to the code.  If it was a full class, provide an overview.  If it was a method, provide details about where the method is found and how it works with the rest of the group's code.  Let me know how these features correspond to your GitHub activity.
* A paragraph (or two) on each team member that describes the positive contributions your team member made to the project.
* A paragraph (or two) on each team member that describes what you wish your team member had done differently. You may think a team member did everything that should have been done. If so, state this.
* A paragraph (or two) describing what you would do differently if you redid the project.

Also include a response to this question:

Your boss is so impressed with your project, that she's awarding your team $3000 in bonuses.  Decide how to divide them up

You: Amount: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Name 1: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Amount \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Name 2: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Amount \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

#### Peer Review

You will receive a Peer Review grade in Canvas. Your peer review grade will be based on the peer reviews of you by your groups members’ individual reflections.

### Project Demonstration

Your group's presentation and demonstration should be approximately 15-20 minutes long.    The main point of the presentation is to demonstrate the features that you have working in the final implementation.   The projects will be graded with the rubric found in this assignment,

Topics that you should cover in the presentation are:

1. Who are you building the software for and why does your version of Twitter meet their needs
2. What features did you include? Show use case diagram/s as needed.
3. How did you approach the design?
   1. Explain the big picture classes included in the final code. Show class diagram/s as needed.
   2. Explain how you satisfied the object-oriented requirements.
4. Demo the software to show off all the functionality

Each group member shall participate in the presentation and demonstration.

### What to Submit

* Presentation and demonstration in class during the last week of classes.   Each group member shall participate in the presentation and demonstration.
* Link to your GitHub repository.
* Zipped folder of your code. Only one team member must submit the zip file.
* Individual reflection and peer review described previously.

### Grading

Grades will be assigned based upon all aspects of the project – group progress check, GitHub contributions, code, design, presentation, individual reflection, and peer review.

If you do not contribute substantive code to this project (as demonstrated by your own reflection as well as the peer reviews of your group), you earn a grade of zero on this assignment.

### Rubric

|  |  |  |
| --- | --- | --- |
| Criteria | Points | Score |
| Project Compiles - Having a compiling project does not earn points, but if your project does not compile, you earn a zero on the project. | 0 |  |
| Code is submitted as a zip file – If you do not submit your code as a zip file, the maximum score will be a 75. | -25 |  |
| Presentation and Demonstration (aesthetics of) | 10 |  |
| Group Progress Check | 10 |  |
| GitHub Contributions | 10 |  |
| Correctness of Code (as Demonstrated) | 30 |  |
| Satisfying Object-oriented requirements | 20 |  |
| Individual reflection | 10 |  |
| Peer review | 10 |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

### UML Class Diagram of One Solution

****