

## IS 2033 – PA4 ASSIGNMENT

**TOTAL POINTS: 100**

**CODE DUE: TUE 12/3/19 BY 11:59 PM**

**Do not submit code that doesn't compile, run, or generate correct output. DO NOT WAIT UNTIL THE DAY BEFORE TO COMPLETE THIS PROGRAM – YOU WON'T FINISH IT!**

**You should plan, but the plan will NOT be graded.**

**READ CAREFULLY "ALL" THE INSTRUCTIONS!!!**

- **No assistance given via email.**
- **If needed, seek help during open tutoring or instructor office hours.**

**UTSA HONOR CODE:** As a UTSA student, you are **bound** by the honor code, so DO NOT cheat on any of your coursework. **By submitting this assignment, you are attesting to your own authorship** based on material from the IS 2033 textbook and/or your professor. Cheating can result in any one, or combination, or all of the following: reduced or failing grade for the assignment, a signed statement of the infraction, reduced or failing grade for the course, reporting of student name to the Department Chair and faculty, Dean's Office and COB faculty, and/or elevation to Student Conduct and Community Standards.

**OBJECTIVE (this is not the program purpose):** Code a program that uses the concepts covered in chapters 1-7 and lecture. This is *not* the program purpose!

**PREP WORK:** Chapters 1-7 (see Ch 7 Arrays of Objects PowerPoint slides posted with this PA); and all labs up through the present.

**GRADING:** You'll be graded on *how well* you follow the program instructions and the **accuracy of your output as reflected in the prompts, the output specifications, the actual Java code given to you in these instructions, and the sample output**. Each line of output can be associated with multiple points in the code! The instructions, prompts and output are what the user wants. You are **not** at liberty to change anything, but code to these requirements.

**PROGRAM INSTRUCTIONS:** Code a program that stores laptop inventory objects in a one-dimensional (1D) array typed as the Laptop class. Create 3 separate class files in a **project folder**; declare variables at the class level; use the **for** loop to populate the array and extract data from the array; and code multiple methods.

1. **Work in groups of only two from the same section.** The group will only **submit one project with all the files**. This means **only one** of the group members should submit the PA. **ALL submissions** will be ignored if both group members post submissions. The **project folder** and the **main() file** will reflect the **last names of both group members**. E.g., **BunnyDuck001PA4**. Through line and method comments, group members will claim authorship and the work has to be evenly distributed. Example:

```
int age = 0;    //By Bugs Bunny: STORES AN AGE
```

```
/**
 * By Daffy Duck
 * main() captures a first and last name, an age, and a zip code.
 * It multiplies the age and zip code to get a product.
 * Output will be made to the console screen and to a
 * GUI window.
 */
```

**WARNING:** One group member CANNOT and MUST NOT carry the load for these assignments. All group members have to be actively engaged in developing and writing code to do well on the exams and in Java II. **All** group communication is to be done through your team in Blackboard.

2. **Develop your code by FIRST planning the logic.** *You won't need to submit your plan.*
  - a. There is an example of a PA4 plan on Blackboard. Use that to assist with your plan.
  - b. The [prompts](#) (to return press Alt then left arrow) tell you what input variables you will need.
  - c. The [final output specifications](#) will tell you the type of calculations you will need (if any) and whether you will need to declare additional variables.
  - d. The [sample output](#) will tell you the order of logic for your code.
3. Download the **Creating Projects in DrJava PDF** (instructor's version) on how to create a **project folder** that manages multiple .java files stored in the same folder. The project folder has the same name as the team members .java file. You are creating *3 separate .java files* to be stored in the project folder:
  - a. **TeamMembersLastNamesSectionNoPANo.java** contains *only* the **main()**.
  - b. **LaptopInventory.java:** All methods and fields are not static. Only 3 fields. All other variables, including Scanner, localized to the methods that use them.
  - c. **Laptop.java:** All methods and fields are not static. Only 4 fields. All other variables localized to the methods that use them. Methods that are prompts do not return from the keyboard. Input values are assigned to fields.
4. The main() will instantiate (create) an object of the *LaptopInventory* class and use that object to call **processLaptops()** and **displayLaptops()**. Make sure you exit main().
5. The **LaptopInventory** class will have 3 methods:
  - a. an empty constructor
  - b. processLaptops()
  - c. displayLaptops()

- d. There are NO value-returning and/or value-receiving methods in the *LaptopInventory* class. You will create a 1D array called **laptops** at the class level as a null array. The *type* for this array will be **Laptop**. You will give this array dimension once the size is known.
- e. There are 3 fields (class variables).
  - 1) One is a boolean *field* called **print**, initialized to false, that tests whether to print the final output in `displayLaptops()`.
  - 2) This boolean is set to true when laptop objects are processed; otherwise, to false when the thank you message is printed.
- f. In `processLaptops()`, the user will be prompted
  - i. to see if inventory data will be entered.
    - 1) If yes, then the user will be prompted for the number of laptops to be added to inventory. That number gives size to the laptops array; then, processing of the laptop objects into the array will commence.
    - 2) If no, the program exits with this message: **Thank you! Exiting program.** *The final output cannot print if this message prints.*
- b. In `displayLaptops()`, the output is printed when **print** is true. Final output is formatted according to specs (see [sample output](#)). Actual code for formatting final output is below in bold. Note there is a **total** variable in the TOTAL COST line which means `getInventoryCost()` needs to be called for each object and added to that variable.

```

switch(i)
{
    case 0: //For $ sign.
        laptopInventory += String.format("%n%-30s %9d @ $%,8.2f ea. %8s $%,17.2f",
            laptops[i].getLaptop(),
            laptops[i].getLaptopQty(),
            laptops[i].getLaptopCost(), " ",
            laptops[i].getInventoryCost());
        break;
    default: //For no $ sign.
        laptopInventory += String.format("%n%-30s %9d @ $%,8.2f ea. %9s %%,17.2f",
            laptops[i].getLaptop(),
            laptops[i].getLaptopQty(),
            laptops[i].getLaptopCost(), " ",
            laptops[i].getInventoryCost());
} //END switch on i to determine when to include $ sign

laptopInventory += String.format("%n%n%27s TOTAL COST OF LAPTOP INVENTORY "
    + "%6s $%,17.2f%n", " ", " ", total);

```

6. The **Laptop** class will have the **set** methods to capture the laptop information and the **get** methods to return the captured data. This class has 4 fields of which 3 are used separately in

their own set methods to capture data specific to the field. **None of the set methods are value-returning in this class.**

- a. Code an empty constructor.
  - b. Code an overloaded constructor that accepts the data for all the fields. Assign the parameter variables to fields with the **same** name.
  - c. Code a method called **setLaptop()** that prompts for the laptop name. It'll have to receive an integer for use in the prompt.
  - d. Code a method called **setLaptopCost()** that prompts for the cost of the laptop.
  - e. Code a method called **setLaptopQty()** that prompts for the number of laptops to be added to inventory.
  - f. Code **get** methods for each field that has a set method. These *get* methods only *return* the fields. In addition to a *get* method for each *set* method, there'll be a **getInventoryCost()** method that returns the calculation of qty and cost for a given laptop.
7. Remember, fields or class variables are used by more than one method.
8. **Commenting Your Program:**
- a. In each program, YOU MUST insert a **program purpose** in the first comment box. The content of that first comment box was shown to you in the *Anatomy of a Java Program* lecture for chapter 1.
  - b. Use Javadoc comment boxes beginning with `/**` and ending with `*/` for your comment boxes.
  - c. Insert a Javadoc comment box above your methods explaining what is going on in the method including the `main()` which is also a method.
  - d. Line comment the import statements and the variables declared at the class level and/or in any method [including `main()`].
  - e. **Comments are worth about 5% of the PA grade.**
9. **Formatting Rules:** Refer to the *Java Style Guide* PDF posted on [Blackboard](#) in IS 2033.
- a. **Always test** your output to validate that your program is functioning properly with the correct output and spacing (line advances and spacing after punctuation). The `%n` can function differently when using separate `printf` statements versus one `printf`.

**PROMPTS:** Code the **bold** from the prompts below in the `printf` statements that capture data into your program. Once again, the prompts tell you your input variables. **Keep in mind that these prompts are not all in the same class.**

1st Prompt: Code in `processLaptops()`.

**Are there laptops to be added to inventory? Enter 'Y' or 'N':**

If the answer to the above question is "N" display the following message:

**Thank you! Exiting program.**

If the answer is "Y" then proceed to prompts 2 through 5.

2nd Prompt: Code in processLaptops().

**How many different laptops to add to inventory?**

3rd Prompt: Code in setLaptop() where 9 represents 1, 2, 3, etc., that is received through a parameter variable.

**Enter laptop no. 9:**

4th Prompt: Code in setLaptopCost(). The Xs represent the laptop name.

**Enter the cost for the XXXXXXXXX:**

5th Prompt: Code in setLaptopQty(). The Xs represent the laptop name.

**Adding how many XXXXXXXXXs to inventory?**

**FINAL OUTPUT SPECIFICATIONS:** The output will print the header message, laptop name, quantity, cost, and inventory value for each laptop in inventory. The header should be triple-line advanced from the last prompt or output. The printing of the output will be in a method called displayLaptops(). The first set of Xs in the header is the name of the month. Go to Appendix I for code on dates. Capitalized Xs require capitalization of the actual output. Zs represent zero-suppression of leading digits. The 9s are numbers in the value.

**LAPTOP INVENTORY AS OF XXXXXXXX 99, 9999**

XXXXXXXXXX	Z,ZZ9 @ \$Z,ZZ9.99 ea.	\$	Z,ZZZ,ZZ9.99
XXXXXXXXXX	Z,ZZ9 @ \$Z,ZZ9.99 ea.		Z,ZZZ,ZZ9.99
<b>TOTAL COST OF LAPTOP INVENTORY</b>		<b>\$</b>	<b>ZZZ,ZZZ,ZZ9.99</b>

⚠ Headers (Titles): Triple-line advance 1<sup>st</sup> line of header title using 2 "%n". All capital Xs means all CAPS.

**\*\*\*END OF OUTPUT SPECIFICATIONS\*\*\***

**SUBMISSIONS REQUIREMENT:** Only one person in the group needs to submit for the entire group.

1. **First File Submission:** Copy ALL your .java code into a Word document. Upload the document to Blackboard.
2. **Second File Submission:**
  - a. **Project Folder:** Your Java files are already in a project folder.

1. To zip the folder, point to it then right click and
  - a) **Filzip** if you have it **OR**
  - b) Click **Send To** then click **Compressed (zipped) Folder**
- b. Upload your zipped project folder to Blackboard.
4. **Uploading to Blackboard:** **Make sure your browser is properly configured for Blackboard (see syllabus).**
  - a. Your submissions are to be uploaded to Blackboard through **Assignments** only.
  - b. **Upload the zip file, but no later than the due date by 11:55 pm**; otherwise, you don't have time to recover from any problems and your assignment may not be accepted by Blackboard.
  - c. Check to make sure your submission is uploaded. Please **do not ask me** to check whether your assignment has been uploaded. You can do this yourself. Or upload during a tutoring session when someone can help you.
  - d. If you submit your assignment before the due date, want to make changes or upload additional files, you can **re-upload** your files.
5. **NO ASSIGNMENTS WILL BE ACCEPTED LATE OR VIA E-MAIL. DO NOT UPLOAD PROGRAMS THAT DON'T COMPILE, RUN, or PRODUCE THE CORRECT OUTPUT. Insert the following note in the Comment section for the assignment on Blackboard then click Submit: My program doesn't compile. OR My program doesn't run. OR My program's output is incorrect.**

**\*\*\*\*SAMPLE OUTPUT\*\*\*\*:** *It is always good to test your code using sample data to see if your program meets the output specifications. **Check to make sure your output matches the sample output in spacing, line advances, and wording.** If not, make the necessary correction(s) in your code and re-run the output. **Copy and paste the output into a comment box at the end of your PA4 .java file that has the main().** The comment box needs to be outside of the close brace for the class. **Change your font in DrJava to Monospaced or Courier New if your output is out of alignment. **Worth 5 points!** Your output will not print in bold.***

**\*\*\*NOT ADDING INVENTORY\*\*\***

Are there laptops to be added to inventory? Enter 'Y' or 'N': N

Thank you! Exiting program.

**\*\*\*ADDING INVENTORY\*\*\***

Are there laptops to be added to inventory? Enter 'Y' or 'N': y

How many different laptops to add to inventory? 5

Enter laptop no. 1: HP Envy 13

Enter the cost for the HP Envy 13: 672.26

Adding how many HP Envy 13s to inventory? 50

Enter laptop no. 2: Asus ZenBook 13 UX333FA

© Linda Shepherd: Instructions and code.

© Laptops: HP, Asus, Dell, Alienware, Razer

Enter the cost for the Asus ZenBook 13 UX333FA: 714.28  
 Adding how many Asus ZenBook 13 UX333FAs to inventory? 45  
 Enter laptop no. 3: Dell XPS 13  
 Enter the cost for the Dell XPS 13: 831.92  
 Adding how many Dell XPS 13s to inventory? 40  
 Enter laptop no. 4: Alienware Area 51-m  
 Enter the cost for the Alienware Area 51-m: 1680.66  
 Adding how many Alienware Area 51-ms to inventory? 30  
 Enter laptop no. 5: Razer Blade Stealth  
 Enter the cost for the Razer Blade Stealth: 1091.60  
 Adding how many Razer Blade Stealths to inventory? 35

LAPTOP INVENTORY AS OF NOVEMBER 09, 2019

HP Envy 13	50 @ \$ 672.26 ea.	\$	33,613.00
Asus ZenBook 13 UX333FA	45 @ \$ 714.28 ea.		32,142.60
Dell XPS 13	40 @ \$ 831.92 ea.		33,276.80
Alienware Area 51-m	30 @ \$1,680.66 ea.		50,419.80
Razer Blade Stealth	35 @ \$1,091.60 ea.		38,206.00
TOTAL COST OF LAPTOP INVENTORY		\$	187,658.20

\*\*\*SCREEN CAPTURE OF FINAL OUTPUT AS SEEN IN DRJAVA CONSOLE PANE\*\*\*

LAPTOP INVENTORY AS OF NOVEMBER 09, 2019

HP Envy 13	50 @ \$ 672.26 ea.	\$	33,613.00
Asus ZenBook 13 UX333FA	45 @ \$ 714.28 ea.		32,142.60
Dell XPS 13	40 @ \$ 831.92 ea.		33,276.80
Alienware Area 51-m	30 @ \$1,680.66 ea.		50,419.80
Razer Blade Stealth	35 @ \$1,091.60 ea.		38,206.00
TOTAL COST OF LAPTOP INVENTORY		\$	187,658.20