**Instructions – Final Exam Practice 2**

**Overview:** An online store wants to test out a simple prototype program for importing and processing a sample of their product data. You are given an excel file with data about products. Products can be one of a variety of categories, but products with a category of “Electronics” also have mandatory warranties attached to them that increase the price. You need to create a program that imports all the product data, creates an Order class that can have several Products or ElectronicProducts, and then prints out relevant information about the products in the order.

Remember that partial credit is given. Do as much as you can.

**Libraries Required:**

* import random
* choose one:
  + import pandas as pd
  + import openpyxl

**External Files Required:**

* product\_data.xlsx (download from learning suite)
  + includes product\_id, product\_name, product\_category, price, warranty\_type, and warranty\_price\_percent
  + warranty\_type and warranty\_price\_percent are only present for Electronics products. The warranty\_price\_percent represents how much the price will increase from the included mandatory warranty.

**Classes Required:**

* Product
  + Instance variables:
    - product\_id, product\_name, product\_category, price
  + Methods:
    - \_\_init\_\_ (the constructor)
    - display\_info\_and\_return\_price
      * prints out “*product\_name: price*”
      * returns the price instance variable
* ElectronicProduct (must inherit from Product)
  + Instance variables:
    - Everything from Product through the use of super()
    - warranty\_type, warranty\_price\_percent
  + Methods:
    - \_\_init\_\_ (the constructor)
    - display\_info\_and\_return\_price
      * prints out “*product\_name: price. Price including warrant\_type warranty: warranty boosted price”*
      * returns the warranty boosted price
* Order
  + Instance variables:
    - order\_id, list\_of\_products (will store a list of products, but should be an empty string by default)
  + Methods:
    - \_\_init\_\_ (the constructor)
    - add\_products\_to\_order
      * Parameters (besides self):
        + A list of products, an integer for the number of products to add to the order
      * Will randomly choose x products from the list you give it and append them to the object’s list\_of\_products variable. Will also print “*Added x products to order #order\_id”*
    - show\_all\_products\_and\_total:
      * prints “*Order #order\_id has the following products:”*
      * then it will run the display\_info\_and\_return\_price method for all the products in the order.
      * Calculates the total price of all the products
      * Prints out *“The order total is x”*

**Logical Flow:**

There are many ways to do this. All that matters is that your code implements the requirements. The exact order, what libraries you use, the specific code, etc. can all vary. The only exception would be doing something in a very, very obviously inferior way (such as hard coding/manually typing in all the information from the files instead of importing it using pandas/openpyxl, etc.)

1. Import the product\_data.xslx file (you can use relative or absolute paths), and have each row in the file become either a Product object, or if the Product\_Category is “Electornics”, then a ElectronicProduct object.
   1. Every column in the excel file represents what should become an instance variable for the Product or ElectronicProduct objects.
   2. ElectronicProduct must inherit from Product (e.g. ElectronicProduct is the child class of Product)
   3. Every product (whether Product or ElectronicProduct) should be stored in a list.
2. Create 5 Order objects and store them in a list:
   1. Order objects don’t come from any excel file. They have instance variables of order\_id and list\_of products.
   2. order\_id should be a random number between 1 and 50000, inclusive.
   3. list\_of\_products should be set to an empty list.
   4. Store all 5 Order objects in a list.
3. For every order, add a random number (1-5 inclusive) of products to it by calling the add\_products\_to\_order method.
   1. add\_products\_to\_order is a method in the Order class.
      1. It has 3 parameters: self, a list of products, and the number of products to add
      2. It should randomly choose a product from the list of products and append it to the Order’s list\_of\_products variable however many times you tell it to.
         1. For example, if you randomly generated the number 3 before calling add\_products\_to\_order, you would pass in 3 along with the list of products into the method, then randomly choose 3 products from the product list and append those to the list\_of\_products variable of the Order.
      3. It should print out “*Added x products to order #order\_id*”
4. After the products have been added to the Orders, run the show\_all\_products\_and\_total method on all the Orders.
   1. show\_all\_products\_and\_total is a method in the Order class.
      1. It prints: “*Order #order\_id has the following products*:”
      2. It will then run the display\_info\_and\_return\_price method on all the products in the Order’s list\_of\_products variable.
         1. display\_info\_and\_return\_price is a method in both the Product and ElectronicProduct classes.
            1. Product version:

Prints: *“product\_name: price”.* I also recommend including a “\t” at the beginning so it looks nicer.

Returns the price

* + - * 1. ElectronicProduct version:

Prints*: “product\_name: price. Price including warranty\_type warranty: warranty\_boosted\_price*”

You need to calculate the warranty\_boosted\_price (you can call it whatever you want). It is the price \* (1 + warranty\_price\_percent).

Round the warranty\_boosted\_price to the 2nd decimal when you print it out.

Return the warranty\_boosted\_price

* + 1. Calculate the total price of the order by adding up the prices/warranty\_boosted\_prices of all the products in the order
    2. Print out “*The order total is order\_total.”* Round it to the 2nd decimal. You can also add “\t” to make it tabbed and look nice.

Upload just the python file to Learning Suite.

**Example Output:**

Note: since the number of products, and which products are added to orders is random, the results will vary greatly

Added 2 products to order #41143

Added 1 products to order #18311

Added 3 products to order #47618

Added 4 products to order #47478

Added 4 products to order #1399

Order #41143 has the following products:

Macbook Air: 720.31. Price including Standard warranty: 1440.62

Plush Unicorn: 596.18

The order total is: 2036.80

Order #18311 has the following products:

Women's Jeans: 113.12

The order total is: 113.12

Order #47618 has the following products:

Teddy Bear: 566.66

Women's Jeans: 113.12

In-ear Headphones: 41.13. Price including Standard warranty: 45.24

The order total is: 725.02

Order #47478 has the following products:

Baby Tablet: 238.49. Price including Special warranty: 310.04

Plush Unicorn: 596.18

Hand Mixer: 238.27

Drone with Camera: 67.8. Price including Standard warranty: 74.58

The order total is: 1219.07

Order #1399 has the following products:

Cooking Recipes: 737.91

Women's Jeans: 113.12

Sunglasses: 25.55

Teddy Bear: 566.66

The order total is: 1443.24

**Requirements:**

|  |  |
| --- | --- |
| **Requirement** | **Sub Requirements** |
| Imports product\_data.xlsx | * + - Each row should become a Product or ElectronicProduct object |
| Includes Product class | * + - Includes instance variables: product\_id, product\_name, product\_category, price     - Includes constructor method |
| Includes ElectronicProduct class | * + - Includes instance variables: everything from Product by using super(), warranty\_type, warranty\_price\_percent     - Includes constructor method |
| Includes Order class | * + - Includes instance variables: order\_id, list\_of\_products     - Includes constructor method     - Order\_id should be a random number between 1 and 50000 inclusive     - List\_of\_products should be an empty list |
| All Products/ElectronicProducts objects stored in a list |  |
| Creates 5 Order objects and stores them in a list | * Should display the title, author, and genre of the book |
| Add\_products\_to\_order method | * Parameters: self, list of products, number of products to add * Randomly chooses a product from the passed list however many times specified by the passed parameter * Prints out message that says how many products were added |
| Add\_products\_to\_order called on each Order object | * Number of products to add is randomly generated to be 1-5 inclusive |
| Show\_all\_products\_and\_total method | * Prints the orderID * Runs the display\_info\_and\_return\_price method for every product in the order * Calculates the order total * Displays the order total |
| Display\_info\_and\_return\_price method | * Should appear for both Product and ElectronicProduct * Prints out the product name and price * For ElectronicProduct, prints out the product name, price, and the price adjusted for the warranty * Returns the price/adjusted price |
| Show\_all\_products\_and\_total is run for all the orders |  |
| includes useful comments | * Should include a name and description at the top * Should also include comments throughout |