Assignment 2

Encapsulation, inheritance, and polymorphism

Submit a single ZIP file called **assignment2.zip** which must contain an IntelliJ project with your Java files. This assignment has 50 marks. A marking scheme is posted on the course webpage.

For this assignment, you will apply the OOP principles of encapsulation and inheritance to the redesign of the electronic store classes from Assignment #1. A list of the classes that must be contained in your program are given below, along with a summary of their state and behaviour.

It is up to you to decide on and implement the appropriate class hierarchy. You will likely need to add more classes to define the best class hierarchy – look for shared state/behaviour between different classes and define a parent class if you need to. **Note that you will not need to define/implement all these methods/attributes in each class if you make proper use of inheritance.** Additionally, you must use proper encapsulation in designing these classes. All your instance variables should be private, unless you can justify them being protected/public. You should be able to complete the assignment without creating any public/protected variables. If you include a protected/public variable, also include a comment with your justification. You can add additional methods (e.g., getter/setter methods) to the classes as you see fit. A large portion of your mark will come from the quality of your class design and proper use of encapsulation/inheritance.

Two test classes are included on the assignment page. Your code should work with these test case classes. When you run the test case classes, the output should be very similar to the output included at the end of this document. The only difference may be in the formatting of the objects, though your toString() methods should closely mirror the formatting included here. You should test your code with your own additional test cases as well. The supplied test classes only provide a minimal amount of testing.

Desktop Class:

State:

1. double price – represents how much the desktop costs
2. int stockQuantity – represents how many units of this desktop are in stock
3. int soldQuantity – represents how many units of this desktop have been sold
4. double cpuSpeed – the CPU speed in Ghz
5. int ram – the amount of RAM in GB
6. boolean ssd – whether the hard-drive is an SSD (true) or HDD (false)
7. int storage – the size of the hard-drive in GB
8. String profile – a string describing the size of the desktop case

Behaviour:

1. Desktop(double price, int quantity, double cpuSpeed, int ram, boolean ssd, int storage, String profile) – constructor for the class
2. double sellUnits(int amount) – simulates selling *amount* units. If there are enough units in stock to meet the request, the quantity attributes must be updated appropriately and the total revenue (revenue = units * price) should be returned. If there are not enough units in stock, no sale should take place and 0.0 should be returned.
3. String toString() – returns a string representing the desktop. This should summarize the state of the desktop, including each attribute. See the example output included at the bottom for examples.

ElectronicStore Class:

State:

1. final int MAX_PRODUCTS = 10 – the maximum number of different product instances that this store can contain. This value should be set to 10 and never changed. Your class should refer to this variable so that the maximum number of products can be changed easily.
2. String name – the name of the electronics store
3. double revenue – the total revenue the store has made through sales. This should initially be 0 but should be updated as products are sold.
4. Product[ ] products – an array to store product objects that are in the store. This array should have size equal to MAX_PRODUCTS. **In case you are working on this before we have finished discussing polymorphism in class, all you need to know is that you can store any class that is below Product in the class hierarchy within this array**. So if you have additional classes that extend Product (or in general, are further down the hierarchy), you can store them in this array without any problems.

Behaviour:

1. ElectronicStore(String name) – constructor for the class.
2. String getName() – returns the name of the store
3. boolean addProduct(Product p) – if there is space remaining in the products array, this method should add p to the products array at the next available array slot and return true. If there is no space remaining in the products array, this method should just return false.
4. void sellProducts() – this method should print out the store's products (see example output below for an idea of how it should look), read an integer from the user representing the product to sell (i.e., what index in the products array), and read an integer from the user representing how many units of the product to sell. You may assume the user will only enter integer numbers but must verify that the values are valid (e.g., a valid item index, an amount greater than 0). If the values supplied by the user are valid, the specified

number of units of the specified product should be sold, if possible. All appropriate variables must be updated in all instances (e.g., revenue, number of units in stock, etc.). If any of the input is invalid, no sales should take place.

5. void sellProducts(int item, int amount) – should sell *amount* units of the product stored at index *item* in the products array, if possible. All appropriate variables must be updated in all instances (e.g., revenue, number of units in stock, etc.). If any of the input is invalid, no sales should take place.
6. double getRevenue() – returns the total revenue the store has made through sales.
7. void printStock() – should print out the products of the store. See the example output at the bottom of the document for examples.

Fridge Class:

State:

1. double price – represents how much the fridge costs
2. int stockQuantity – represents how many units of this fridge there are in stock
3. int soldQuantity – represents how many units of this fridge have been sold
4. int wattage – the wattage rating of the fridge
5. String color – the color of the fridge
6. String brand – the brand name of the fridge
7. double cubicFeet – The volume of the fridge in cubic feet
8. boolean hasFreezer – Whether the fridge has a freezer or not

Behaviour:

1. Fridge(double price, int quantity, int wattage, String color, String brand, double cubicFeet, boolean freezer) – constructor for the class
2. double sellUnits(int amount) – simulates selling *amount* units. If there are enough units in stock to meet the request, the quantity attributes must be updated appropriately and the total revenue (revenue = units * price) should be returned. If there are not enough units in stock, no sale should take place and 0.0 should be returned.
3. String toString() – returns a string representing the fridge. This should summarize the state of the fridge, including each attribute. See the example output included at the bottom for examples.

Laptop Class:

State:

1. double price – represents how much the laptop costs
2. int stockQuantity – represents how many units of this laptop there are in stock
3. int soldQuantity – represents how many units of this laptop have been sold
4. double cpuSpeed – the CPU speed, in Ghz
5. int ram – the amount of RAM in GB
6. boolean ssd – whether the hard-drive is an SSD (true) or HDD (false)

7. int storage – the size of the hard-drive in GB
8. double screenSize – the size of the screen in inches

Behaviour:

1. Laptop(double price, int quantity, double cpuSpeed, int ram, boolean ssd, int storage, double screenSize) – constructor for the class
2. double sellUnits(int amount) – simulates selling *amount* units. If there are enough units in stock to meet the request, the quantity attributes must be updated appropriately and the total revenue (revenue = units * price) should be returned. If there are not enough units in stock, no sale should take place and 0.0 should be returned.
3. String toString() – returns a string representing the laptop. This should summarize the state of the laptop, including each attribute. See the example output included at the bottom for examples.

Product Class:

State:

1. double price – represents how much the product costs
2. int stockQuantity – represents how many units of this product are in stock
3. int soldQuantity – represents how many units of this product have been sold

Behaviour:

1. Product(double price, int quantity) – constructor for the class
2. double sellUnits(int amount) – simulates selling *amount* units. If there are enough units in stock to meet the request, the quantity attributes must be updated appropriately and the total revenue for selling (revenue = units * price) should be returned. If there are not enough units in stock, no sale should take place and 0.0 should be returned.

ToasterOven Class:

State:

1. double price – represents how much the toaster oven costs
2. int stockQuantity – represents how many units of this toaster are in stock
3. int soldQuantity – represents how many units of this toaster have been sold
4. int wattage – the wattage rating of the toaster oven
5. String color – the color of the toaster oven
6. String brand – the brand name of the toaster oven
7. int width – the width of the toaster oven
8. boolean convection – whether the toaster has convection heating or not

Behaviour:

1. ToasterOven(double price, int quantity, int wattage, String color, String brand, int width, boolean convection) – constructor for the class
2. double sellUnits(int amount) – simulates selling *amount* units. If there are enough units in stock to meet the request, the quantity attributes must be

updated appropriately and the total revenue for selling (revenue = units * price) should be returned. If there are not enough units in stock, no sale should take place and 0.0 should be returned.

3. String toString() – returns a string representing the toaster oven. This should summarize the state of the toaster oven, including each attribute. See the example output included at the bottom for examples.

ElectronicStoreStockTester Class Output

If you run the ElectronicStoreStockTester class, you should get the following output (bold font added to help find specific parts of the output, your code does not need to produce bold output):

```
Watts Up Electronics's Stock Is:
0. Compact Desktop PC with 3.0ghz CPU, 16GB RAM, 250GB HDD drive. (100.0 dollars each, 10 in stock, 0 sold)
1. Server Desktop PC with 4.0ghz CPU, 32GB RAM, 500GB SSD drive. (200.0 dollars each, 10 in stock, 0 sold)
2. 15.0 inch Laptop PC with 2.5ghz CPU, 16GB RAM, 250GB SSD drive. (150.0 dollars each, 10 in stock, 0 sold)
3. 16.0 inch Laptop PC with 3.5ghz CPU, 24GB RAM, 500GB SSD drive. (250.0 dollars each, 10 in stock, 0 sold)
4. 15.5 cu. ft. Sub Zero Fridge (White, 250 watts) (500.0 dollars each, 10 in stock, 0 sold)
5. 23.0 cu. ft. Sub Zero Fridge with Freezer (Stainless Steel, 125 watts) (750.0 dollars each, 10 in stock, 0 sold)
6. 8 inch Danby Toaster (Black, 50 watts) (25.0 dollars each, 10 in stock, 0 sold)
7. 12 inch Toasty Toaster with convection (Silver, 50 watts) (75.0 dollars each, 10 in stock, 0 sold)

Buy-nary Computing's Stock Is:
0. Compact Desktop PC with 3.0ghz CPU, 16GB RAM, 250GB SSD drive. (150.0 dollars each, 5 in stock, 0 sold)
1. Server Desktop PC with 3.5ghz CPU, 32GB RAM, 500GB SSD drive. (250.0 dollars each, 5 in stock, 0 sold)
2. 15.0 inch Laptop PC with 2.5ghz CPU, 16GB RAM, 250GB HDD drive. (100.0 dollars each, 15 in stock, 0 sold)
3. 16.0 inch Laptop PC with 3.5ghz CPU, 24GB RAM, 500GB SSD drive. (175.0 dollars each, 15 in stock, 0 sold)
4. 15.5 cu. ft. Sub Zero Fridge (Black, 250 watts) (350.0 dollars each, 10 in stock, 0 sold)
```

```
5. 23.0 cu. ft. Sub Zero Fridge with Freezer (White, 125 watts) (600.0
dollars each, 10 in stock, 0 sold)
6. 6 inch Danby Toaster (Graphite, 50 watts) (25.0 dollars each, 10 in
stock, 0 sold)
7. 10 inch Toasty Toaster with convection (Red, 50 watts) (75.0
dollars each, 10 in stock, 0 sold)
```

Ohm-y Goodness Electronics's Stock Is:

```
0. Low-Profile Desktop PC with 3.0ghz CPU, 16GB RAM, 250GB SSD drive.
(175.0 dollars each, 10 in stock, 0 sold)
1. Standard Desktop PC with 3.5ghz CPU, 32GB RAM, 1000GB HDD drive.
(150.0 dollars each, 15 in stock, 0 sold)
2. 16.0 inch Laptop PC with 3.5ghz CPU, 16GB RAM, 500GB SSD drive.
(350.0 dollars each, 5 in stock, 0 sold)
3. 13.0 inch Laptop PC with 2.5ghz CPU, 8GB RAM, 125GB SSD drive.
(500.0 dollars each, 5 in stock, 0 sold)
4. 12.0 cu. ft. Sub Zero Fridge (Black, 250 watts) (250.0 dollars
each, 5 in stock, 0 sold)
5. 15.0 cu. ft. Sub Zero Fridge (White, 125 watts) (275.0 dollars
each, 5 in stock, 0 sold)
6. 8 inch Danby Toaster (Graphite, 50 watts) (30.0 dollars each, 10 in
stock, 0 sold)
7. 12 inch Toasty Toaster with convection (Red, 50 watts) (80.0
dollars each, 10 in stock, 0 sold)
8. Low-Profile Desktop PC with 3.0ghz CPU, 16GB RAM, 250GB SSD drive.
(175.0 dollars each, 10 in stock, 0 sold)
9. Standard Desktop PC with 3.5ghz CPU, 32GB RAM, 1000GB HDD drive.
(150.0 dollars each, 15 in stock, 0 sold)
```

ElectronicStoreSellingTester Class Output

If you run the ElectronicStoreSellingTester class, you should get the following output (bold font added to help find specific parts of the output, your code does not need to produce bold output):

Watts Up Electronics's Starting Stock Is:

```
0. Compact Desktop PC with 3.0ghz CPU, 16GB RAM, 250GB HDD drive.
(100.0 dollars each, 10 in stock, 0 sold)
1. Server Desktop PC with 4.0ghz CPU, 32GB RAM, 500GB SSD drive.
(200.0 dollars each, 10 in stock, 0 sold)
2. 15.0 inch Laptop PC with 2.5ghz CPU, 16GB RAM, 250GB SSD drive.
(150.0 dollars each, 10 in stock, 0 sold)
```

3. 16.0 inch Laptop PC with 3.5ghz CPU, 24GB RAM, 500GB SSD drive. (250.0 dollars each, 10 in stock, 0 sold)
4. 15.5 cu. ft. Sub Zero Fridge (White, 250 watts) (500.0 dollars each, 10 in stock, 0 sold)
5. 23.0 cu. ft. Sub Zero Fridge with Freezer (Stainless Steel, 125 watts) (750.0 dollars each, 10 in stock, 0 sold)
6. 8 inch Danby Toaster (Black, 50 watts) (25.0 dollars each, 10 in stock, 0 sold)
7. 12 inch Toasty Toaster with convection (Silver, 50 watts) (75.0 dollars each, 10 in stock, 0 sold)

Buy-nary Computing's Starting Stock Is:
0. Compact Desktop PC with 3.0ghz CPU, 16GB RAM, 250GB SSD drive. (150.0 dollars each, 5 in stock, 0 sold)
1. Server Desktop PC with 3.5ghz CPU, 32GB RAM, 500GB SSD drive. (250.0 dollars each, 5 in stock, 0 sold)
2. 15.0 inch Laptop PC with 2.5ghz CPU, 16GB RAM, 250GB HDD drive. (100.0 dollars each, 15 in stock, 0 sold)
3. 16.0 inch Laptop PC with 3.5ghz CPU, 24GB RAM, 500GB SSD drive. (175.0 dollars each, 15 in stock, 0 sold)
4. 15.5 cu. ft. Sub Zero Fridge (Black, 250 watts) (350.0 dollars each, 10 in stock, 0 sold)
5. 23.0 cu. ft. Sub Zero Fridge with Freezer (White, 125 watts) (600.0 dollars each, 10 in stock, 0 sold)
6. 6 inch Danby Toaster (Graphite, 50 watts) (25.0 dollars each, 10 in stock, 0 sold)
7. 10 inch Toasty Toaster with convection (Red, 50 watts) (75.0 dollars each, 10 in stock, 0 sold)

Ohm-y Goodness Electronics's Starting Stock Is:
0. Low-Profile Desktop PC with 3.0ghz CPU, 16GB RAM, 250GB SSD drive. (175.0 dollars each, 10 in stock, 0 sold)
1. Standard Desktop PC with 3.5ghz CPU, 32GB RAM, 1000GB HDD drive. (150.0 dollars each, 15 in stock, 0 sold)
2. 16.0 inch Laptop PC with 3.5ghz CPU, 16GB RAM, 500GB SSD drive. (350.0 dollars each, 5 in stock, 0 sold)
3. 13.0 inch Laptop PC with 2.5ghz CPU, 8GB RAM, 125GB SSD drive. (500.0 dollars each, 5 in stock, 0 sold)
4. 12.0 cu. ft. Sub Zero Fridge (Black, 250 watts) (250.0 dollars each, 5 in stock, 0 sold)
5. 15.0 cu. ft. Sub Zero Fridge (White, 125 watts) (275.0 dollars each, 5 in stock, 0 sold)

6. 8 inch Danby Toaster (Graphite, 50 watts) (30.0 dollars each, 10 in stock, 0 sold)
7. 12 inch Toasty Toaster with convection (Red, 50 watts) (80.0 dollars each, 10 in stock, 0 sold)
8. Low-Profile Desktop PC with 3.0ghz CPU, 16GB RAM, 250GB SSD drive. (175.0 dollars each, 10 in stock, 0 sold)
9. Standard Desktop PC with 3.5ghz CPU, 32GB RAM, 1000GB HDD drive. (150.0 dollars each, 15 in stock, 0 sold)

Watts Up Electronics's Ending Stock Is:
0. Compact Desktop PC with 3.0ghz CPU, 16GB RAM, 250GB HDD drive. (100.0 dollars each, 0 in stock, 10 sold)
1. Server Desktop PC with 4.0ghz CPU, 32GB RAM, 500GB SSD drive. (200.0 dollars each, 10 in stock, 0 sold)
2. 15.0 inch Laptop PC with 2.5ghz CPU, 16GB RAM, 250GB SSD drive. (150.0 dollars each, 10 in stock, 0 sold)
3. 16.0 inch Laptop PC with 3.5ghz CPU, 24GB RAM, 500GB SSD drive. (250.0 dollars each, 0 in stock, 10 sold)
4. 15.5 cu. ft. Sub Zero Fridge (White, 250 watts) (500.0 dollars each, 10 in stock, 0 sold)
5. 23.0 cu. ft. Sub Zero Fridge with Freezer (Stainless Steel, 125 watts) (750.0 dollars each, 10 in stock, 0 sold)
6. 8 inch Danby Toaster (Black, 50 watts) (25.0 dollars each, 10 in stock, 0 sold)
7. 12 inch Toasty Toaster with convection (Silver, 50 watts) (75.0 dollars each, 0 in stock, 10 sold)

Buy-nary Computing's Ending Stock Is:
0. Compact Desktop PC with 3.0ghz CPU, 16GB RAM, 250GB SSD drive. (150.0 dollars each, 0 in stock, 5 sold)
1. Server Desktop PC with 3.5ghz CPU, 32GB RAM, 500GB SSD drive. (250.0 dollars each, 4 in stock, 1 sold)
2. 15.0 inch Laptop PC with 2.5ghz CPU, 16GB RAM, 250GB HDD drive. (100.0 dollars each, 0 in stock, 15 sold)
3. 16.0 inch Laptop PC with 3.5ghz CPU, 24GB RAM, 500GB SSD drive. (175.0 dollars each, 15 in stock, 0 sold)
4. 15.5 cu. ft. Sub Zero Fridge (Black, 250 watts) (350.0 dollars each, 10 in stock, 0 sold)
5. 23.0 cu. ft. Sub Zero Fridge with Freezer (White, 125 watts) (600.0 dollars each, 10 in stock, 0 sold)
6. 6 inch Danby Toaster (Graphite, 50 watts) (25.0 dollars each, 10 in stock, 0 sold)

7. 10 inch Toasty Toaster with convection (Red, 50 watts) (75.0 dollars each, 10 in stock, 0 sold)

Ohm-y Goodness Electronics's Ending Stock Is:
0. Low-Profile Desktop PC with 3.0ghz CPU, 16GB RAM, 250GB SSD drive. (175.0 dollars each, 10 in stock, 0 sold)
1. Standard Desktop PC with 3.5ghz CPU, 32GB RAM, 1000GB HDD drive. (150.0 dollars each, 0 in stock, 15 sold)
2. 16.0 inch Laptop PC with 3.5ghz CPU, 16GB RAM, 500GB SSD drive. (350.0 dollars each, 0 in stock, 5 sold)
3. 13.0 inch Laptop PC with 2.5ghz CPU, 8GB RAM, 125GB SSD drive. (500.0 dollars each, 5 in stock, 0 sold)
4. 12.0 cu. ft. Sub Zero Fridge (Black, 250 watts) (250.0 dollars each, 1 in stock, 4 sold)
5. 15.0 cu. ft. Sub Zero Fridge (White, 125 watts) (275.0 dollars each, 0 in stock, 5 sold)
6. 8 inch Danby Toaster (Graphite, 50 watts) (30.0 dollars each, 10 in stock, 0 sold)
7. 12 inch Toasty Toaster with convection (Red, 50 watts) (80.0 dollars each, 10 in stock, 0 sold)
8. Low-Profile Desktop PC with 3.0ghz CPU, 16GB RAM, 250GB SSD drive. (175.0 dollars each, 10 in stock, 0 sold)
9. Standard Desktop PC with 3.5ghz CPU, 32GB RAM, 1000GB HDD drive. (150.0 dollars each, 8 in stock, 7 sold)

Watts Up Electronics's total revenue was: 4250.0
Buy-nary Computing's total revenue was: 2500.0
Ohm-y Goodness Electronics's total revenue was: 7425.0