Due Date: 18.11.2019 23:55

CENG211 – Programming Fundamentals

Homework #2

In this homework you are expected to implement an "**Art Vault Application**" in Java. You should fulfill the concepts of:

- Inheritance
- Polymorphism
- CSV File I/O
- Array Lists
- Abstract Classes
- Interfaces

In the Art Vault Application, different types of data that are related to each other are stored.

- The first usage of this application is to list famous people and artworks to inform the user about the artwork and its creator.

- The second usage of this application is to perform a search with any given information.

- The third usage of this application is to sell and buy artworks. For trading, application also calculates the creation/design costs of artworks. For example, a 550 kilograms bronze sculpture costs 550x180 TL when the cost of a kilo of bronze is 180TL.

In this application, there are world-wide famous **Person**s. An **Artist** has information that includes name, birth date, date of death, nationality and the periods that she/he lived. An **Architect** has name, birth date, date of death and nationality information as well.

There are also very well-known **Artwork**s which are **Painting**, **Sculpture** or **Architecture**. A painting has *name, style, artist* and *dimensions* e.g. length=40, width=70 in centimeters. A sculpture has *name, style* and *artist* information as well as *material* and *weight* information. An architecture also has *name, style, architects* and *dimensions* e.g. length = 100, width = 200, height = 300 in meters. Remember that there can be more than one architect of an architecture.

All the information about artworks and famous people is kept in a **Vault**. By the way, users can access all related information about them.

There are also **Buyer** and **Seller** classes who has **Wallet**s and **PrivateCollection**s in this application. They can **Trade** with calculated costs of artworks.

You should implement necessary classes that are mentioned in the above scenario as well as related abstract classes (e.g. Person, Artwork), related interfaces, and other helper classes (e.g. CSVReader, ArtVaultAppMenu, ArrayListSorter, and etc.) with the information given below:

First usage: printing the lists of items. In this operation, user will choose what to print firstly (e.g. print the list of artists). After that, user will choose how to sort the list (i.e. provide **IComparable** interface, Hint: You can use Object Class, instanceof operator, and casting). Sorting will be as follows:

- Paintings can be sorted by name, style and artist
- Sculptures can be sorted by name, style, artist and material
- Architectures can be sorted by name and style
- Artists can be sorted by name, birthday, nationality
- Architects can be sorted by name, birthday, nationality

Second usage: searching the vault. Each item should be searchable (i.e. provide **ISearchable** interface). Searching by full word is enough (Program does not have to gather related information when user write "Leo" or "Leonardo" or "leonardo da vinci"; it is okay to gather when "Leonardo da Vinci" is typed as in the CSV file). User shall type a word and program should gather ALL related information as follows:

- Painting: name, style and artist
- Sculpture: name, style, artist and material
- Architecture: name, style and architects
- Artist: name, born, died, nationality, periods
- Architect: name, born, died, nationality

For both operations, default sorting will be by name (For list operation if user chooses to sort by style, items should be also sorted by name for each style).

Third usage: In this application, **Artwork** class is abstract and you should implement its abstract `calculateCost()` method in each subclass. Artwork's cost is calculated up to the following information:

- The creation cost of a painting is calculated by multiplying its dimensions and its unit (1 cm^2) price according to its style. Unit price is 7 TL if the style is "Renaissance", 5.5 TL if the style is "Baroque". Other styles' unit price is 4.5 TL default.

- The creation cost of a sculpture is calculated by multiplying its weight and its material's unit (1 kg) price. If the material is marble unit price is 15 TL. If the material is bronze it is 180 TL.

- The design cost of an architecture is calculated by multiplying its dimensions and its unit (1m^3) price according to its style. Unit price is 1 TL if the style is "Gothic", is 0.8 TL if the style is "Baroque". Other styles' unit price is 0.6 TL default.

With the calculated costs trading can be done. Remember that not all artworks are tradable. Although each artwork has a creation or a design cost, you cannot sell each artwork. Therefore you need to implement **ITradable** interface which includes `isTradable()` and `tradeToBuyer()`

Non-tradable items:

- For painting: items that have "Gothic" style are non-tradable.
- For sculpture: items that have "Baroque" style are non-tradable.
- For architecture: items that have "Renaissance" style are non-tradable.

Trade operation will be performed automatically. There will be one seller and four buyers. Seller will have every artwork in the Vault. Each buyer will have 5 million TL in their wallets. Each buyer will buy one artworks and this will be a random operation (they can buy any random artwork from seller).

When `tradeToBuyer` is called, some money will be taken from the buyer's wallet and will be added to the seller's wallet. Also artwork that is traded will be taken from the seller's private collection and will be added to the buyer's private collection. These operations should be seen by user. Trade operation should be reset in each cycle (Program does not have to remember what is traded or how much money buyers and seller have when trade operation is chosen again).

Program should work continuously unless user wants to exit.

You are expected to read and parse necessary objects using the given CSV file **CENG211_HW2_ArtVaultData.csv**:

- In the file, the information is as follows:
  Object Number (1-Painting, 2-Sculpture, 3-Architecture, 4-Artist, 5-Architect) and related data about objects that are given below:
  - Painting: 1, Name, Style, Artist, Dimension-1, Dimension-2
  - Sculpture: 2, Name, Style, Artist, Material, Weight
  - Architecture: 3, Name, Style, Dimension-1, Dimension-2, Dimension-3, Architects
  - Artist: 4, Name, Born, Died, Nationality, Periods
  - Architect: 5, Name, Born, Died, Nationality

The data in the CSV file is mixed. You are expected to store that data and give a well arranged output according to requests of user.

Expected Output Format:

```
Please enter the number of the operation you want to perform:

1) Print the lists
2) Search the vault
3) Trade
0) Exit

1


Please choose the list:

1) Print the list of artists
2) Print the list of architects
3) Print the list of paintings
4) Print the list of sculptures
5) Print the list of architectures
0) Exit

3


Please choose the sorting method:

1) Sort by name
2) Sort by artist
3) Sort by style
0) Exit

2

Painting:
        Name: Bacchus
        Artist: Caravaggio
        Style: Baroque
        Dimensions: 95cm x 85cm

Painting:
        Name: The Conversion of Saint Paul
        Artist: Caravaggio
```

```
Please enter the number of the operation you want to perform:

1) Print the lists
2) Search the vault
3) Trade
0) Exit

2

Enter the keyword:

Michelangelo

Artist:
        Name: Michelangelo
        Born: 1475
        Died: 1564
        Nationality: Italian
        Periods: High Renaissance, Italian Renaissance, Renaissance, Mannerism

Painting:
        Name: The Creation of Adam
        Artist: Michelangelo
        Style: Renaissance
        Dimensions: 280cm x 570cm

Sculpture:
        Name: Bacchus
        Artist: Michelangelo
        Style: Renaissance
        Material: Marble
        Weight: 2587 kg

Sculpture:
        Name: David
        Artist: Michelangelo
        Style: Renaissance
        Material: Marble
```

```
Please enter the number of the operation you want to perform:

1) Print the lists
2) Search the vault
3) Trade
0) Exit

3
Seller's money: 0,00 TL
Buyer 1's money: 5000000,00 TL
Buyer 2's money: 5000000,00 TL
Buyer 3's money: 5000000,00 TL
Buyer 4's money: 5000000,00 TL
Trade started:

Buyer 1 bought:

Painting:
        Name: The Creation of Adam
        Price: 1117200,00 TL

Buyer 2 bought:

Architecture:
        Name: Casa Mila
        Price: 82406,40 TL

Buyer 3 bought:

Painting:
        Name: Guernica
        Price: 1220278,50 TL

Buyer 4 bought:

Sculpture:
        Name: David
        Price: 27000,00 TL

Trade completed:

Seller's money: 2446884,90 TL
Buyer 1's money: 3882800,00 TL
Buyer 2's money: 4917593,60 TL
Buyer 3's money: 3779721,50 TL
Buyer 4's money: 4973000,00 TL
```

Important Notes:

1. You should mind that not every architect has a dedicated info line. Some architects are mentioned only in the lists of architectures (e.g. Barbara Schock-Werner is one of the architects of Cologne Cathedral, but does not have separated architect info).
2. You should keep in mind that some classes and some parameters are related and have in common. Thus, you should pay attention that when creating abstract classes.
3. The data in the "**Expected Output Format**" does NOT represent the actual results, it simply indicates "how the output should look like" in console.
4. You are NOT allowed to use Java's **Comparable or Comparator** in this homework. You should implement your own IComparable interface.
5. You can use standard **java.io** packages to read files. Do NOT use other 3rd party libraries.
6. You should use relative paths (e.g. `Files/sample.csv`) instead of absolute paths (e.g. `C:\\user\\eclipse-workspace\\MyProject\\Files\\sample.csv`).
7. You are expected to write clean, readable, and tester-friendly code (e.g. make the selections with numbers). Please try to maximize reusability and prevent from redundancy in your methods.

Assignment Rules:

1. In this lecture's homework, there is no cheating allowed. If any cheating has been detected, they will be **graded as 0** and there will be no further discussion on this.
2. You are expected to submit your homework in groups. Therefore, **only one of you** will be sufficient to submit your homework.
3. Make sure you export your homework as an **Eclipse project**. You can use other IDEs as well, however, you must test if it supported by Eclipse.
4. Submit your homework through **CMS**.
5. Name and export your Java Project with your assigned **group ID** (which will be announced on CMS) as the given format below:

   G25_CENG211_HW2.zip

6. Please be informed that your submissions may be anonymously used in software testing and maintenance research studies. Your names and student IDs will be replaced with non-identifying strings. If you do not want your submissions to be used in research studies, please inform the instructor (Dr. Tuglular) via e-mail.