**CSE 220 Homework Assignment 3 (Due 7/11/24)**

**1) (40 pts)** *(Class Relationships and Definitions)* a. Define the **three major class relationships** that exist in object-oriented programming, and provide a minimum of one example for each relationship. (You can use either examples from class or examples of your own design.) Your response should be 3-6 sentences in length.  
b. Explain in practical terms the differences between **arrays** and **ArrayLists** in Java in 2-4 sentences.   
c. Explain in practical terms the differences between the terms **method overriding** and **method overloading** as they relate to Java in 2-4 sentences.

**2) (60 pts)** *(Writing Classes and Methods)* For this problem you are going to implement three classes to simulate the underpinnings of an *art collectors’ database*. The database will consist of a **Collector** class (detailing information about buyers and sellers of art), a **Transaction** class (including information about a transaction involving any piece of art), and a **TransactionDB** class (maintaining information about a collection of transactions and providing some summary information about them).

**Important**: you may name your instance variables whatever you like, but attribute encapsulation will be enforced in grading your code: this means that your instance variables *must not* be **public**. Instead, you will want to create accessor methods (ex: **Transaction** should likely have an accessor method called **getYear** to return the year.)

More details regarding each class and its required public methods follow (<\_\_\_\_\_> represents a given attribute/identifier):

**a. Collector**: A Collector object is created using a constructor of the form   
***public Collector(int colID, String colName, char category)***, where *colID* is a numeric identifier, *colName* is a name of the collecting organization, and  
*category* is a single character indicating the organization’s nature (ex: ‘C’ for commercial, ‘N’ for non-profit, etc.)  
The Collector class must have a public *toString* method that returns a string of the form   
“*{ID: <colID> | Name: <colName> | Cat: <category>}”*.

**b. Transaction**: a Transaction object is created using a constructor of the form  
**public Transaction(String pieceName, Collector seller, Collector buyer, double sales, int year)**, where pieceName is a string description of the art piece, seller and buyer are Collector objects for the transaction , and sales and year are self-explanatory.  
The Transaction class requires a public *toString* method that returns a string of the form   
[<pieceName> sold by <seller> to <buyer> for $ <sales> in <year>]. Note that <seller> and <buyer> should rely on the *Collector* toString output for easiest implementation.

c. **TransactionDB**: a TransactionDB object is created using an empty constructor (i.e. TransactionDB() ), which should initialize an empty collection of transactions. *It’s up to you to decide the best structure to use for the collection of transactions – but the best options are probably an ArrayList or array.* TransactionDB requires the following methods:  
i. **public void add(Transaction newTr)**: Adds *newTr* to the transaction collection.  
ii. **public String toString()**: returns a string consisting of every individual transaction – each one separated by newlines. *Note that this should be simple to implement if you have correctly implemented toString for the Transaction class.*iii. **public double totalCost()**: returns the *cumulative cost (i.e. sum of all transaction sales)* in the database.  
iv. **public void printTrInDuration(int startYear, int endYear)**: prints all transactions that fall within the duration between the startYear and endYear (inclusive).

**As noted above, all instance variables must be private, but you are permitted (and will almost certainly want) some accessor methods. *It is highly recommended that you use the TestTransactions.java file included with the assignment to test your implemented classes, as that will be used to assess that they function correctly.* The included text file should give you an idea of the output you should expect from a correctly implemented program.**

**Responses to Problem #1 should be in .doc(x) or .pdf format, and they should be bundled in a single .zip file with separate .java files for each of the above classes (Collector.java, Transaction.java, TransactionDB.java) for Problem #2.**

**Your submission should have the filename “LN\_FN\_3.zip” where LN is your last name and FN is your first name.**