***CSC 3020***

***Java Programming***

**Winter Term 2023**

**Project 02**

**40 points**

**Due 12/04/2023 (11:45 A.M.)**

**The goal of this Project is:**

1. Being able to Analyze, Design, implement, and test a practical real-world application.
2. Learn about user-defined classes
3. Learn about composition
4. Learn about inheritance
5. Learn about interface and polymorphism
6. Works with files and implement exceptions

**Requirements:**

* Using java IDE software, implement the algorithm. Includes informative comments.
* Test the code for each problem and verify that the algorithm works.

**Restrictions:**

You must work individually. Use only material from class or from the textbook (chapters 1- 13). All code must be the work of the individual. Do not share your code or copy from external resources.

**Submission**

Submit 11 files (6 files for Q1, 5 files for Q2, and 6 files for the extra credit problem). Convert all files to text files. Upload all files to the Canvas by the due date. DO NOT compress or email your files. Class names and interface names must be the same as given in bold font. Upload the files to Canvas in order as listed in each question.

**Question 01 – Hospital Billing System (25 Points)**

In this project, you will design various classes and write a program to computerize the billing system of a hospital.

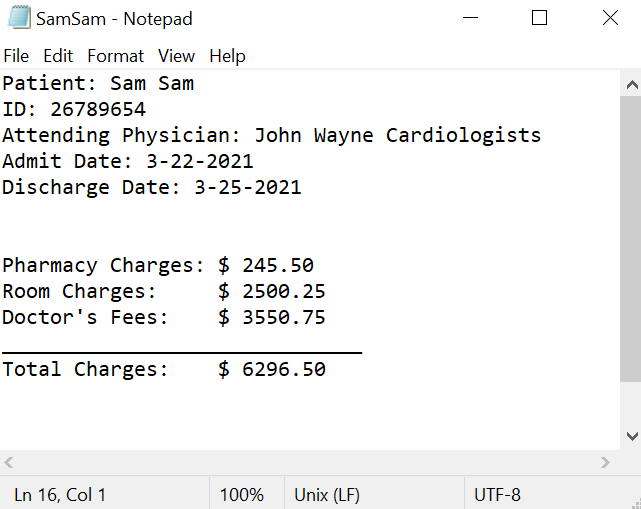
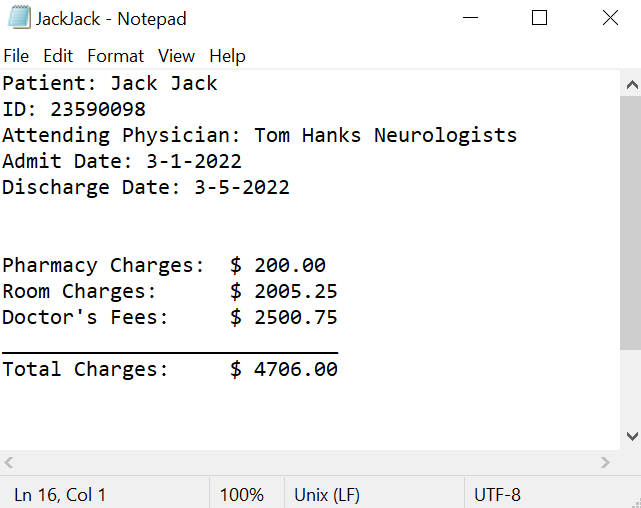
1. (2 points) Design the ***class Person***. Two common attributes of a person are the person’s first name and last name. The typical operations on a person’s name are to set the name and print the name.
2. (2 points) Design the ***class Doctor***, inherited from the class Person with an additional data member to store a doctor’s specialty. Add appropriate constructors and methods to initialize, access, and manipulate the data members.
3. (4 points) Design the ***class Bill*** with data members to store a patient’s ID and the patient’s hospital charges such as pharmacy charges for medicine, the doctor’s fee, and the room charges. Add appropriate constructors and methods to initialize, access, and manipulate the data members.
4. (2 points) design the ***class Date***. Three common attributes of a date are month, day number, and year. Some of the operations that need to be performed on a date are to set the date and to print the date.
5. (5 points) Design the ***class Patient***, inherited from the class Person with additional data members to store a patient’s ID, date of birth, attending physician’s name, the date when the patient was admitted in the hospital, and the date when the patient was discharged from the hospital. (Use the class Date to store the date of birth, admit date, discharge date, and the class Doctor to store the attending physician’s name.) Add appropriate constructors and methods to initialize, access, and manipulate the data members.
6. (4 points) Override method toString in each of the above classes.
7. (6 points) Write a program (***TestHospital class***) to test your classes. Print patient’s name, attending physician, admit date, discharge date, and all charges to console window and to a .txt file as shown below. (use patient’s first and last name for file name; if patient’s name is AAA BCD, the output file name should be AAABCD.txt.). Your printout must match the sample output below. Your program should process more than one patient.

*3 points will be deducted for not including comments.*

*Sample output - Console Window*

|  |
| --- |
| **Enter patient's ID: 26789654**  **Enter patient's first name: Sam**  **Enter patient's last name: Sam**  **Enter doctor's first name: John**  **Enter doctor's last name: Wayne**  **Enter doctor's specialty: Cardiologists**  **Enter Admit Date (day/month/year): 3/22/2021**  **Enter Discharged Date (day/month/year): 3/25/2021**  **Enter pharmacy Charges, room Rent, and doctor Fee: 245.50 2500.25 3550.75**  **Patient: Sam Sam**  **ID: 26789654**  **Attending Physician: John Wayne Cardiologists**  **Admit Date: 3-22-2021**  **Discharge Date: 3-25-2021**  **Pharmacy Charges: $ 245.50**  **Room Charges: $ 2500.25**  **Doctor's Fees: $ 3550.75**  **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**  **Total Charges: $ 6296.50**  **Do you want to continue? Enter 'y' for yes, 'n' for no**  **y**  **Enter patient's ID: 23590098**  **Enter patient's first name: Jack**  **Enter patient's last name: Jack**  **Enter doctor's first name: John**  **Enter doctor's last name: Wayne**  **Enter doctor's specialty: Neurologists**  **Enter Admit Date (day/month/year): 3/1/2022**  **Enter Discharged Date (day/month/year): 3/5/2022**  **Enter pharmacy Charges, room Rent, and doctor Fee: 200.0 2005.25 2500.75**  **Patient: Jack Jack**  **ID: 23590098**  **Attending Physician: Tom Hanks Neurologists**  **Admit Date: 3-1-2022**  **Discharge Date: 3-5-2022**  **Pharmacy Charges: $ 200.00**  **Room Charges: $ 2005.25**  **Doctor's Fees: $ 2500.75**  **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**  **Total Charges: $ 4706.00**  **Do you want to continue? Enter 'y' for yes, 'n' for no**  **n**  **Process finished with exit code 0** |

*Sample output - text files*

** **

**Question 02 - Ship Classes (15 Points)**

1. (2 points) Design a ***Ship class*** that the following members:

* A field for the name of the ship (a string)
* A field for the year that the ship was built (a string)
* A constructor and appropriate accessors and mutators
* A toString method that displays the ship’s name and the year it was built

1. (2 points) Design a ***CruiseShip class*** that extends the Ship class. The CruiseShip class should have the following members:

* A field for the maximum number of passengers (an int )
* A constructor and appropriate accessors and mutators
* A toString method that overrides the toString method in the base class. The CruiseShip class’s toString method should display the ship’s name, year, and the maximum number of passengers.

1. (2 points) Design a ***CargoShip class*** that extends the Ship class. The CargoShip class should have the following members:

* A field for the ship capacity in tonnage (an int )
* A constructor and appropriate accessors and mutators
* A toString method that overrides the toString method in the base class. The CargoShip class’s toString method should display the ship’s name, year, and the ship’s capacity.

1. (2 points) Design a ***TankerShip class*** that extends the Ship class. The TankerShip class should have the following members:

* A field for type of liquid or gas in bulk (a String) and a field for capacity in tonnage (a double).
* A constructor and appropriate accessors and mutators
* A toString method that overrides the toString method in the base class. The TankerShip class’s toString method should display the ship’s name, year, type of liquid, and capacity.

1. (3 points) Override method compareTo in CruiseShip (compare by number of passengers), class CargoShip (compare by capacity), and TankerShip (compare by type of liquid).
2. (2 points) Demonstrate the classes in a program (***ShipDemo class***) that has a Ship arrayList. Assign various CruiseShip, CargoShip, and TankerShip objects to the array elements. The program should then step through the array, calling each object’s toString method.
3. (2 points) Create three objects of CruiseShip class and store them in an array; sort the array; print all objects before and after you sort them.

*2 points will be deducted for not including comments.*

*Sample Output - Console Window*

|  |
| --- |
| Name: Disney Magic  Year: 1998  Maximum passengers: 2400  ----------------------------  Name: Black Pearl  Year: 1800  Cargo capacity: 50000 tons  ----------------------------  Name: Lolipop  Year: 1960  Type: Oil  Capacity: 500000.0  ----------------------------  Before sorting three objects of CruiseShip:  [1] Name: Disney Magic1  Year: 1998  Maximum passengers: 2400  [2] Name: Disney Magic2  Year: 2002  Maximum passengers: 2500  [3] Name: Disney Magic3  Year: 2011  Maximum passengers: 400  After sorting three objects of CruiseShip:  [1] Name: Disney Magic2  Year: 2002  Maximum passengers: 2500  [2] Name: Disney Magic1  Year: 1998  Maximum passengers: 2400  [3] Name: Disney Magic3  Year: 2011  Maximum passengers: 400  Process finished with exit code 0 |

**Extra Credit – Encoding and Decoding (20 Points)**

1. Create an **interface** **MessageEncoder** that has a single abstract method encode(plainText), where plainText is the message to be encoded. The method returns the encoded message.
2. Create **Abstract class Cipher** that has a single abstract method cipherType(); the method returns a string.
3. Create a **class SubstitutionCipher** that extends class Cipher and implements the interface MessageEncoder defined in parts a and b. The constructor should have one parameter called shift. Define method cipherType so that the method returns the string “SubstitutionCipher”. Define the method encode so that each letter is shifted by the value in shift. For example, if shift is 3, *a* will be replaced by *d*, *b* will be replaced by *e*, *c* will be replaced by *f*, and so on. You may wish to define a private method that shifts a single character.
4. Create a **class ShuffleCipher** that extends class Cipher and implements the interface MessageEncoder defined in parts a and b. The constructor should have one parameter called n. Define method cipherType so that the method returns the string “ShuffleCipher”. Define the method encode so that the message is shuffled n times. To perform one shuffle, split the message in half and then take characters from each half alternately. For example, if the message is “abcdefghi”, the halves are “abcd” and “efghi”. The shuffled message is “aebfcgdhi”. You may wish to define a private method that performs one shuffle.
5. Create an **interface MessageDecoder** that has a single abstract method decode( cipherText), where cipherText is the message to be decoded. The method returns the decoded message.
6. Modify the classes SubstitutionCipher and ShuffleCipher defined in parts c and d, so that they implement MessageDecoder as well as the interface MessageEncoder.
7. Finally, create class **CodeProgram** with a *menu driven program* that allows a user to encode and decode messages read from a file and written to another file.

**Sample program output**

|  |
| --- |
| Welcome to the Cipher program.  Type 1 for Substitution Cipher.  Type 2 for Shuffle Cipher.  1  What is the key (shift amount) for your code?  3  SubstitutionCipher-shift amount = 3  Enter input file name  a.txt  Enter output file name  b.txt  Encode (E) or Decode (D)  E  Encoded text is saved in b.txt  Do you want to do another message (Y)  Y  Welcome to the Cipher program  Type 1 for Substitution Cipher.  Type 2 for Shuffle Cipher.  1  What is the key (shift amount) for your code?  3  SubstitutionCipher-shift amount = 3  Enter input file name  b.txt  Enter output file name  c.txt  Encode (E) or Decode (D)  D  Decoded text is saved in file c.txt  Do you want to do another message (Y)  Y  Welcome to the Cipher program  Type 1 for Substitution Cipher.  Type 2 for Shuffle Cipher.  2  What is the key (shuffle amount) for your code?  2  ShuffleCipher-shuffle amount = 2  Enter input file name  a.txt  Enter output file name  b2.txt  Encode (E) or Decode (D)  E  Encoded text is saved in file b2.txt  Do you want to do another message (Y)  Y  Welcome to the Cipher program  Type 1 for Substitution Cipher.  Type 2 for Shuffle Cipher.  2  What is the key (shuffle amount) for your code?  2  ShuffleCipher-shuffle amount = 2  Enter input file name  b2.txt  Enter output file name  c2.txt  Encode (E) or Decode (D)  D  Decoded text is saved in file c2.txt  Do you want to do another message (Y)  N  Process finished with exit code 0 |

**Sample output files:**

a.txt: theText Message

b.txt: encoded message from a.txt using Substitution Cipher.

c.txt: decoded message from b.txt using Substitution Cipher

b2.txt: encoded message from a.txt using Shuffle Cipher.

c2.txt: decoded message from b2.txt using Shuffle Cipher.

A screenshot of a computer screen

Description automatically generated