King Saud University

College of Computer and Information Sciences

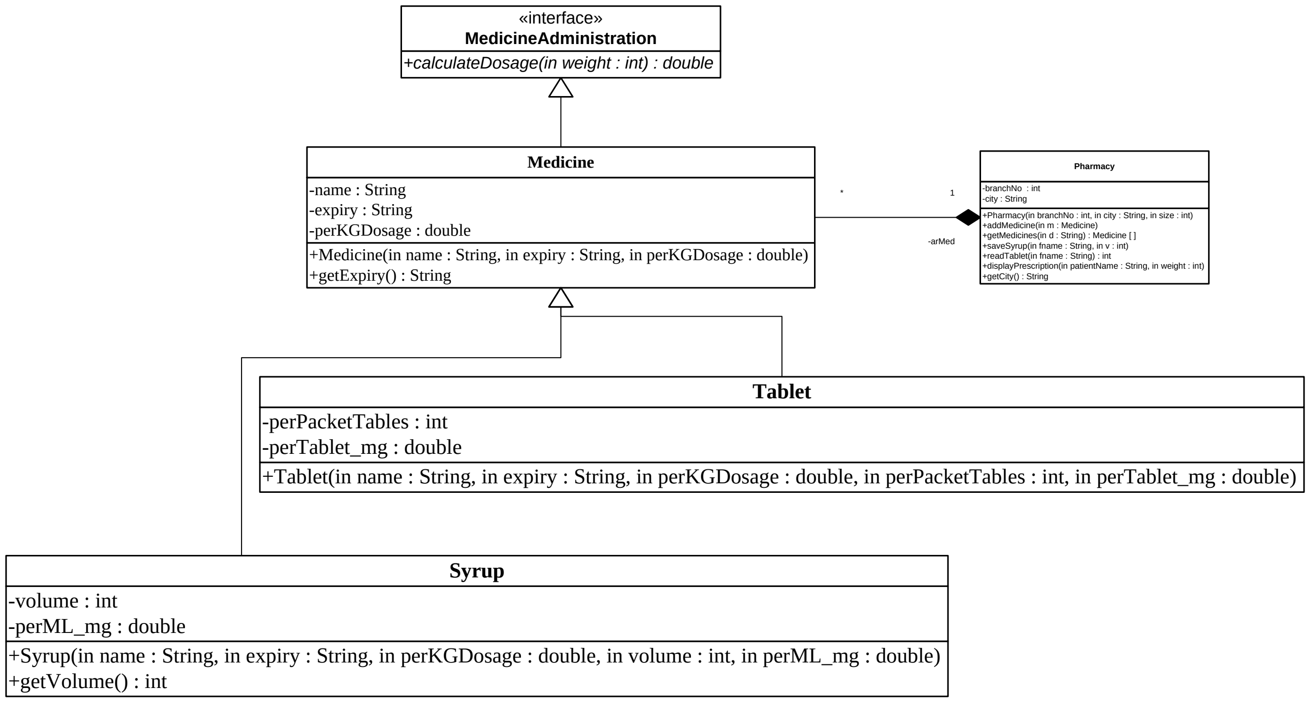
Computer Science Department

Course Code:	CSC 113
Course Title:	Computer Programming II
Semester:	Spring 2017
Exercises Cover Sheet:	**Final Exam**

Student Name:	
Student ID:	
Student Section No.	

Tick the Relevant	Computer Science B.Sc. Program ABET Student Outcomes	Question No. Relevant Is Hyperlinked	Covering %
X	**a) Apply knowledge of computing and mathematics appropriate to the computer science;**		
	b) Analyze a problem, and identify and define the computing requirements appropriate to its solution		
X	**c) Design, implement and evaluate a computer-based system, process, component, or program to meet desired needs;**		
X	**d) Function effectively on teams to accomplish a common goal;**		
	e) Understanding of professional, ethical, legal, security, and social issues and responsibilities;		
	f) Communicate effectively with a range of audiences;		
	g) Analyze the local and global impact of computing on individuals, organizations and society;		
	h) Recognition of the need for, and an ability to engage in, continuing professional development;		
X	**i) Use current techniques, skills, and tools necessary for computing practices.**		
	j) Apply mathematical foundations, algorithmic principles, and computer science theory in the modeling and design of computer-based systems in a way that demonstrates comprehension of the tradeoffs involved in design choices;		
	k) Apply design and development principles in the construction of software systems of varying complexity;		

Exercise1:


«interface»
MedicineAdministration
+calculateDosage(in weight : int) : double
Medicine
-name : String
-expiry : String
-perKGDosage : double
+Medicine(in name : String, in expiry : String, in perKGDosage : double)
+getExpiry() : String
*
1
-arMed
Pharmacy
-branchNo : int
-city : String
+Pharmacy(in branchNo : int, in city : String, in size : int)
+addMedicine(in m : Medicine)
+getMedicines(in d : String) : Medicine [ ]
+saveSyrup(in fname : String, in v : int)
+readTablet(in fname : String) : int
+displayPrescription(in patientName : String, in weight : int)
+getCity() : String
Tablet
-perPacketTables : int
-perTablet_mg : double
+Tablet(in name : String, in expiry : String, in perKGDosage : double, in perPacketTables : int, in perTablet_mg : double)
Syrup
-volume : int
-perML_mg : double
+Syrup(in name : String, in expiry : String, in perKGDosage : double, in volume : int, in perML_mg : double)
+getVolume() : int

MedicineAdministration Interface***:***

- Methods:
  - ***calculateDosage (weight:*** *int)***:*** This method calculates and returns the total dosage to be administered to the patient based on his weight (the weight of the patient). The total dosage is calculated using the following formula:
    - For ***Syrup***: the total dosage = (perKGDosage * weight) / perML_mg. This will return the number of milliliters of the syrup.
    - For ***Tablet***: the total dosage = (perKGDosage * weight) / perTablet_mg. This will return the total number of tablets.

Medicine class

- Attributes:
  - ***name:*** the name of the Medicine.
  - ***expiry***: the expiry date of the Medicine.
  - ***perKGDosage***: This attribute describes the strength of the medicine in milligrams per Kilogram of the patient's weight.
- Methods:
  - ***Medicine (name:*** *String****,*** ***expiry:*** *String****,*** ***perKGDosage:*** *double* ***)***: constructor.
  - ***getExpiry()*:** this method returns the expiry date of the Medicine.

Syrup class:

- Attributes:
    - ***volume***: the total volume of the syrup in milliliters
    - ***perML_mg***: Number of milligrams of the medicine per milliliter.
- Methods:
    - ***Syrup (name:** String**, expiry:** String**, perKGDosage:*** *double,* **volume***: int,* ***perML_mg****: double)***:** constructor.
    - ***getVolume():***this method returns the volume of the Syrup.

Tablet class***:***

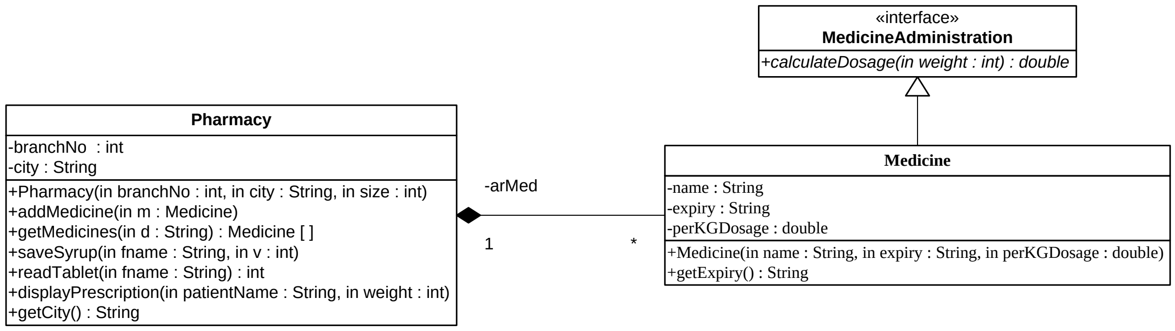
- Attributes:
    - ***perPacketTables***: number of tablets in one blister packet.
    - ***perTablet_mg***: Number of milligrams of the medicine per tablet
- Methods:
    - ***Tablet(name:** String**, expiry:** String**, perKGDosage:*** *double,* ***perPacketTables****: int,* ***perTablet_mg****: double)*: constructor

QUESTION: Translate into Java code:

1. The interface ***MedicineAdministration***
2. The class ***Medicine.***
3. The class ***Tablet.***

Exercise 2:

Let's consider the class ***Medicine*** described in exercise 1.



«interface»
MedicineAdministration
+calculateDosage(in weight : int) : double
Pharmacy
-branchNo : int
-city : String
+Pharmacy(in branchNo : int, in city : String, in size : int)
+addMedicine(in m : Medicine)
+getMedicines(in d : String) : Medicine [ ]
+saveSyrup(in fname : String, in v : int)
+readTablet(in fname : String) : int
+displayPrescription(in patientName : String, in weight : int)
+getCity() : String
-arMed
1
*
Medicine
-name : String
-expiry : String
-perKGDosage : double
+Medicine(in name : String, in expiry : String, in perKGDosage : double)
+getExpiry() : String

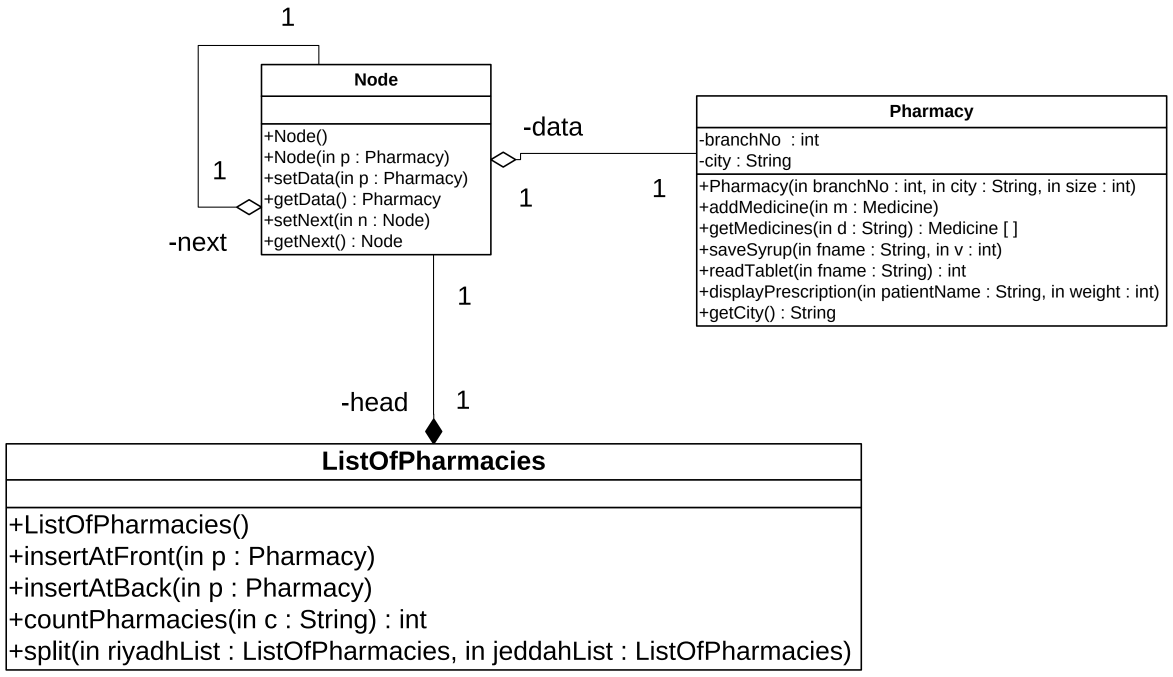
Pharmacy class***:***

- Attributes:
  - ***branchNo***: the branch number of the Pharmacy.
  - ***city***: the city name of the Pharmacy.
- Methods:
  - ***Pharmacy (branchNo:*** *int****,*** ***city:*** *String****,*** ***size:*** *int****)***: constructor
  - ***addMedicine(m:*** *Medicine****):*** this method adds the Medicine ***m*** to the Pharmacy. This method raises an *ArrayOutOfBoundException* if the array ***arMed*** is full.
  - ***getMedicines(d:*** *String****):*** This method returns an array containing all Medicine objects that will expire on date ***d***.
  - ***saveSyrup (fileName:*** *String****,*** ***v:*** *int****):***this method saves, into the file ***filename***, all Syrup objects of the Pharmacy having a volume greater than ***v***.
  - ***readTablet***(***fileName***: String): This method returns the number of Tablets stored in the file ***fileName***.
  - ***getCity():***this method returns the city name of the Pharmacy.

QUESTION: Translate into Java code the class ***Pharmacy.***

Exercise 3:

Let's consider the class ***Pharmacy*** described in exercise 2.



1
Node
+Node()
+Node(in p : Pharmacy)
+setData(in p : Pharmacy)
+getData() : Pharmacy
+setNext(in n : Node)
+getNext() : Node
1
-next
-data
1
1
Pharmacy
-branchNo : int
-city : String
+Pharmacy(in branchNo : int, in city : String, in size : int)
+addMedicine(in m : Medicine)
+getMedicines(in d : String) : Medicine [ ]
+saveSyrup(in fname : String, in v : int)
+readTablet(in fname : String) : int
+displayPrescription(in patientName : String, in weight : int)
+getCity() : String
1
-head
1
ListOfPharmacies
+ListOfPharmacies()
+insertAtFront(in p : Pharmacy)
+insertAtBack(in p : Pharmacy)
+countPharmacies(in c : String) : int
+split(in riyadhList : ListOfPharmacies, in jeddahList : ListOfPharmacies)

ListOfPharmacies class***:***

- Attributes:
  - ***head***: references the first element of the linked list.
- Methods:
  - ***ListOfPharmacies ()***: constructor.
  - ***insertAtFront (p:*** *Pharmacy*)**:**this method adds the Pharmacy ***p*** at the front of the linked list.
  - ***insertAtBack (p:*** *Pharmacy*)**:**this method adds the Pharmacy ***p*** at the end of the linked list.
  - ***countPharmacies(c:*** *String*)***:***this method returns the number of pharmacies in the city ***c***.
  - ***split (riyadhList:*** *ListOfPharmacies*, ***jeddahList:*** *ListOfPharmacies*)***:*** This method inserts all pharmacies of Riyadh in the list ***riyadhList*** and the pharmacies of Jeddah in the list ***jeddahList***.

QUESTION: Translate into Java code the following methods of the class ***ListOfPharmacies:***

1. The method ***countPharmacies.***
2. The method ***split.***