



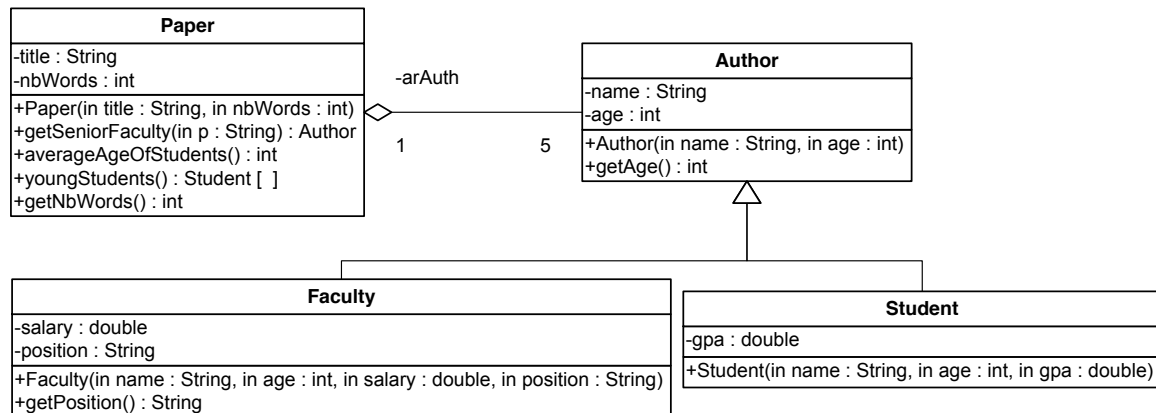
King Saud University

College of Computer and Information Sciences

Computer Science Department

	Course Code:	CSC 113	
	Course Title:	Computer Programming II	
	Semester:	Fall 2014	
	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:



Author class:

- Attributes:
 - **name**: the name of the author.
 - **age**: the age of the author.
- Methods:
 - **Author (name: String, age: int)**: constructor
 - **getAge()**: this method returns the age of the author.

Faculty class

- Attributes:
 - **salary**: the salary of the faculty.
 - **position**: the position of the faculty.
- Methods:
 - **Faculty (name: String, age: int, salary: double, position: String)**: constructor.
 - **getPosition()**: this method returns the position of the faculty.

Student class:

- Attributes:
 - **gpa**: the gpa of the student.
- Methods:
 - **Student (name: String, age: int, gpa: double)**: constructor.

Paper class:

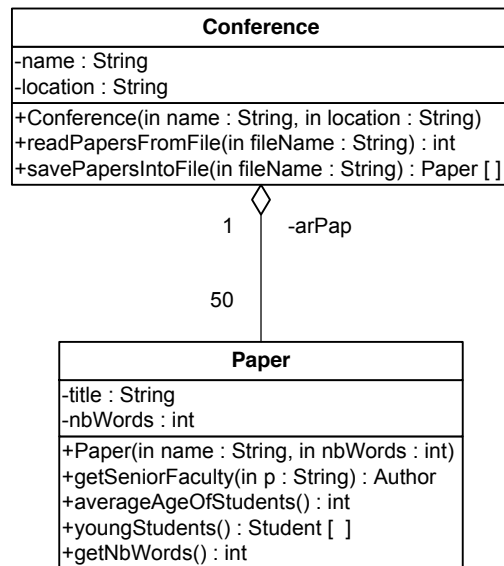
- Attributes:
 - **title**: the title of the paper.
 - **nbWords**: the number of words of the paper.
- Methods:
 - **Paper(title: String, nbWords: int)**: constructor

- ***getSeniorFaculty (p: String)***:this method receives a position p and returns the oldest Faculty having the position *p*.
- ***averageAgeOfStudents()***: This method returns the average age of all students of the paper.
- ***youngStudents()***: this method returns an array of students of the paper whose age is less or equal than the average age (the average age of all students of the paper).
- ***getNbWords ()***:this method returns the number of words of the paper.

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

Exercise 2:

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



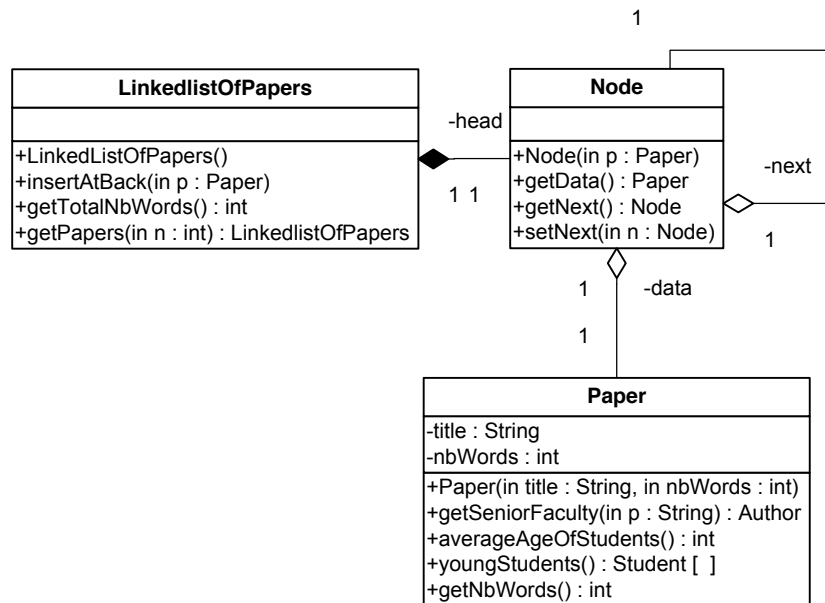
Conference class:

- Attributes:
 - **name**: the name of the conference.
 - **location**: the name of the city where the conference is held.
- Methods:
 - **Conference(name: String, location:String)**: constructor
 - **readPapersFromFile(fileName: String)**:this method reads objects of type paper from the file *fileName* and inserts them into the array of papers of the conference. It returns the number of objects read from the file and inserted in the array.
 - **savePapersIntoFile (fileName: String)**:this method processes all the papers of the conference. Papers containing more (or equal) than 500 words are saved into the file *fileName*. The remaining papers (those containing less than 500 words) are placed and returned in an array.

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

Exercise 3:

Let's consider the same Paper class described in exercise 1.



LinkedListOfPapers class:

- Attributes:
 - **head**: references the first element of the linked list.
- Methods:
 - **LinkedListOfPapers()**: constructor.
 - **insertAtBack (p: Paper)**: this method adds the paper *p* at the end of the linked list.
 - **getTotalNbWords()**: this method returns the total number of words of papers of the linked list.
 - **getPapers (n: int)**: this method returns a new linked list containing all papers of the current linked list having a number of words equal to *n*.

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