King Saud University

College of Computer and Information Sciences

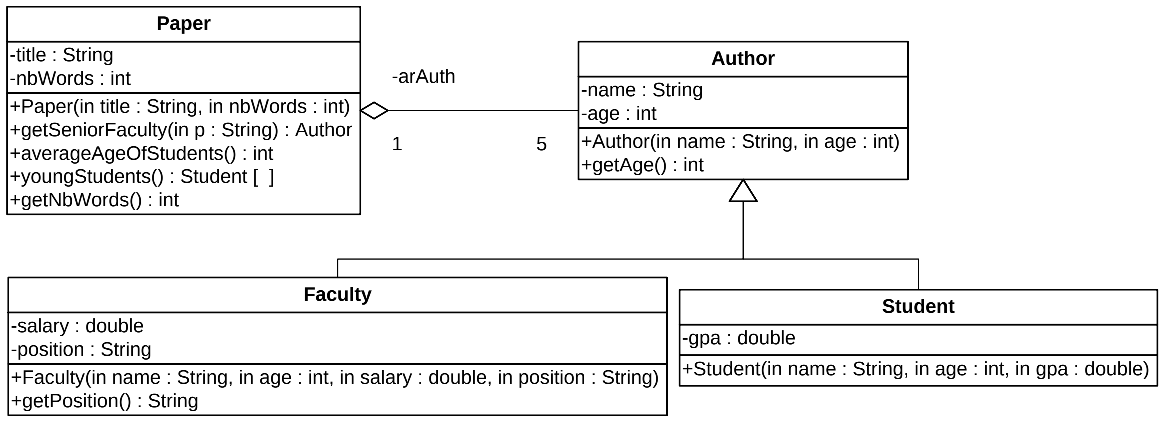
Computer Science Department

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:


Paper
-title : String
-nbWords : int
+Paper(in title : String, in nbWords : int)
+getSeniorFaculty(in p : String) : Author
+averageAgeOfStudents() : int
+youngStudents() : Student [ ]
+getNbWords() : int
-arAuth
1
5
Author
-name : String
-age : int
+Author(in name : String, in age : int)
+getAge() : int
Faculty
-salary : double
-position : String
+Faculty(in name : String, in age : int, in salary : double, in position : String)
+getPosition() : String
Student
-gpa : double
+Student(in name : String, in age : int, in gpa : double)

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***:***

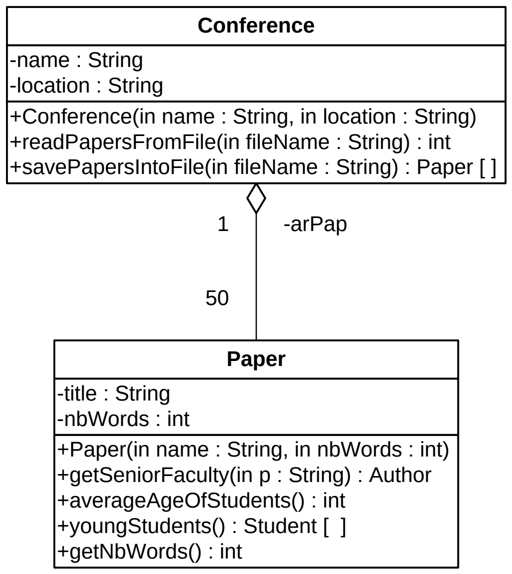
- Attributes:
    - ***title***: the title of the paper.
    - ***nbWords***: the number of words of the paper.
- Methods:
    - ***Paper(title: String, nbWords: int)***: constructor

- ***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
-name : String
-location : String
+Conference(in name : String, in location : String)
+readPapersFromFile(in fileName : String) : int
+savePapersIntoFile(in fileName : String) : Paper [ ]
1
-arPap
50
Paper
-title : String
-nbWords : int
+Paper(in name : String, in nbWords : int)
+getSeniorFaculty(in p : String) : Author
+averageAgeOfStudents() : int
+youngStudents() : Student [ ]
+getNbWords() : int

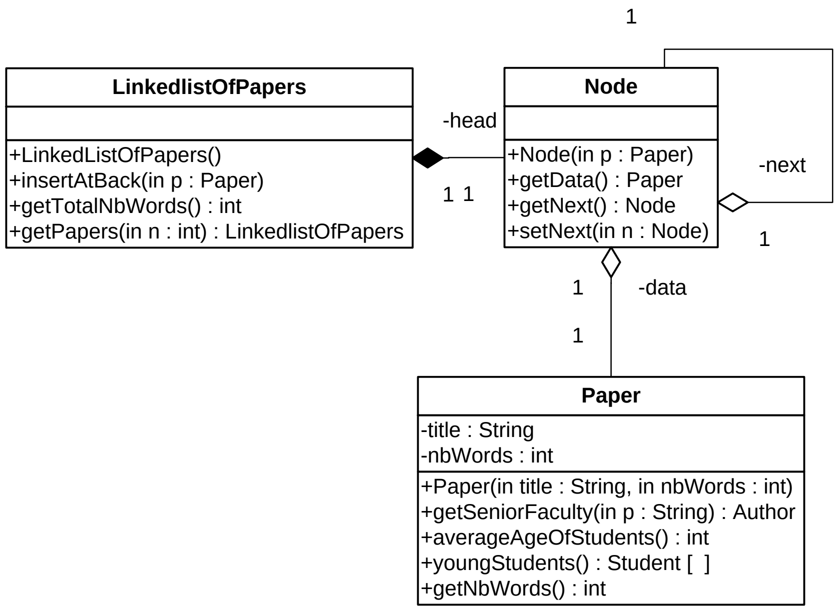
Conference class***:***

- 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.***

Ex.1

```
public class Paper implements Serializable { ................................................  /31
       private String title;
       private int nbWords;
       private Author arAuth[];   ................................................ 1
       private int nbA;           ................................................ 1
       public Paper(String title, int nbWords) { ................................................ /2
              this.title = title;
              this.nbWords = nbWords;
              nbA = 0;            ................................................ 1
              arAuth = new Author[5];  ................................................ 1
       }

       public Author getSeniorFaculty(String p) { ................................................ /10
              int max = 0;  ................................................ 1
              Faculty oldest = null; ................................................ 1
              Faculty f;
              for(int i=0; i<nbA; i++) { ................................................ 1
                     if( (arAuth[i].getAge() > max) && (arAuth[i] instanceof Faculty) ) { …1 + 1
                            f = (Faculty) arAuth[i]; ................................................ 1
                            if(f.getPosition().equals(p)) { ................................................ 1
                                          max = arAuth[i].getAge(); ................................................ 1
                                          oldest = f; ................................................ 1
                            }
                     }
              }
              Return oldest; ................................................ 1
       }
       public int averageAgeOfStudents() { ................................................ /8
              int j=0,sum = 0; ................................................ 1 + 1
              for(inti=0; i<nbA; i++) { ................................................ 1
                     if(arAuth[i] instanceof Student) { ................................................ 1
                            sum += arAuth[i].getAge();................................................ 1
                            j++; ................................................ 1
                     }
              }
              if(j != 0) ................................................ 1
                     return sum/j; ................................................ 1
              return 0;
       }
       public Student[] youngStudent() { ................................................ /8
              Student[] s = new Student[nbA]; ................................................ 1
              int j = 0; ................................................ 1
              int avg = averageAgeOfStudents();

              for(int i=0; i<nbA; i++) { ................................................ 1
                     if(arAuth[i] instanceof Student && arAuth[i].getAge() <= avg) .................... 1 + 1
                            s[j++]=(Student) arAuth[i]; ................................................ 1 + 1
              }
              Return s; ................................................ 1
       }
       public int getNbWords() { ................................................ 1
              Return nbWords;
       }
}
```

Ex.2

```
import java.io.*;

publicclass Conference { ............................................  /28
	private String name;
	private String location;
	private Paper arPap[];............................................ 1
	private int nbP; ............................................ 1

	public Conference(String name, String location) { ............................................ /2
		this.name = name;
		this.location = location;
		arPap = new Paper[50]; ............................................ 1
		nbP = 0; ............................................ 1
	}

	public int readPapersFromFile(String filename) throws IOException { ............................................ /12
		File f = new File(filename); ............................................ 1
		FileInputStream fs = newFileInputStream(f); ............................................ 1
		ObjectInputStream os = newObjectInputStream(fs); ............................................ 1
		int j = 0; ............................................ 1

		try { ............................................ 1
			while(true) { ............................................ 1

				arPap[nbP++] = (Paper) os.readObject();............................................ 1 + 1 + 1
				j++; ............................................ 1
			}
		}
		catch (EOFException e){}
		catch (IndexOutOfBoundException e){} ............................................ 1

		os.close();
		return j; ............................................ 1
	}

	public int readPapersFromFile(String filename) throws IOException { ............................................ /12
		File f = new File(filename); ............................................ 1
		FileInputStream fs = newFileInputStream(f); ............................................ 1
		ObjectInputStream os = newObjectInputStream(fs); ............................................ 1

		int j = 0; ............................................ 1
		Paper p;

		try { ............................................ 1
			while(true) { ............................................ 1
				p = (Paper) os.readObject();............................................ 1
				if (nbP < arPap.length) { ............................................ 1
					arPap[nbP++] = p; ............................................ 1 + 1
					j++; ............................................ 1
				}
			}
		}
		catch (EOFException e){os.close();}

		return j; ............................................ 1
	}
```

```
    public Paper[] savePapersIntoFile(String filename) throws IOException { ................................ /12
        File f = newFile(filename); ............................................... 1
        FileOutputStream fs = newFileOutputStream(f); ............................................... 1
        ObjectOutputStream os = new ObjectOutputStream(fs); ............................................... 1

        Paper p[] = new Paper[50]; ............................................... 1
        int j=0; ............................................... 1

        for(int i=0; i<nbP; i++) { ............................................... 1
            if(arPap[i].getNbWords() >= 500) ............................................... 1
                os.writeObject(arPap[i]); ............................................... 1
            else
                p[j++] = arPap[i]; ............................................... 1 + 1
        }

        os.close();............................................... 1
        return p; ............................................... 1
    }
}
```

Ex.3

```
public class LinkedListOfPapers { ................................................ /22

	private Node head;

	public LinkedListOfPapers() { ................................................ /1
		head = null; ................................................ 1
	}

	public void insertAtBack(Paper p) { ................................................ /8
		Node newN = new Node(p); ................................................ 1

		if(head == null) { ................................................ 1
			head = newN; ................................................ 1
		}
		else { ................................................ 1
			Node current = head; ................................................ 1
			while(current.getNext() != null) ................................................ 1
				current = current.getNext();................................................ 1

			current.setNext(newN); ................................................ 1
		}
	}

	public int getTotalNbWords() { ................................................ /6
		int sum = 0; ................................................ 1
		Node current = head; ................................................ 1
		while(current != null) { ................................................ 1
			sum += current.getData().getNbWords();................................................ 1
			current = current.getNext();................................................ 1
		}

		return sum; ................................................ 1
	}

	public LinkedListOfPapersgetPapers(int n) { ................................................ /7
		LinkedListOfPapersresult = new LinkedListOfPapers(); ................................................ 1

		Node current = head; ................................................ 1
		while(current != null) ................................................ 1
		{
			if(current.getData().getNbWords() == n) ................................................ 1
				result.insertAtBack(current.getData());................................................ 1

			current = current.getNext();................................................ 1
		}
		return result; ................................................ 1
	}

}
```