FACULTY OF COMPUTING
& INFORMATION TECHNOLOGY
KING ABDULAZIZ UNIVERSITY

FCIT
KAU

كلية الحاسبات
وتقنية المعلومات
جامعة الملك عبدالعزيز

KAU Management System

Assignment 3

CPCS203 Programming-II Spring 2019

Delivery Date: Sunday, March 17, 2019

Instructions

- This program must ONLY be submitted on the Blackboard!
- This project worth 6% of the overall module marks (100%).
- NO assignment will be accepted after 11:59 pm for any reason
- Students can submit their assignment between 11 and 11:59 PM but in this case it will be consider as late submission.
- For discussion schedule, check the captain name, date and time on the BlackBoard.
- ***Further information is provided in the course syllabus.***

Objectives

- Learn how to use and implement the concept of Inheritance (which supports code reusability) and dynamic binding.
- Learn how to use and implement Passing Object concepts.
- Learn to use and implement String, StringBuilder, File I/O (Reading/Writing from/to files).

How to submit your assignment?

- Submit your assignment on the Blackboard ONLY.
- Make sure to add your names / IDs / Section / Your name / Assignment number at the beginning of your program

Files provided with assignment

- Input file samples:
  - **input.txt**: This file contains all input information that needs to be entered into the system.
- Output files:
  - **StudentWrite.txt**: This output file displays all information in the system (The information in this file is read from **input.txt)**.
  - **AllReportsFor25Students (25 .txt files)**: These files are printed for all students individually to print their detailed report (Print_Report command).

Note: Please check the format of each of these files and make sure you follow this format in your assignment solution.

1.1 What is KAU Management System?

KAU Management System is simulation software which simulates various KAU management oriented functionalities. The system can register students, teachers, invigilators, exam venues, courses offered, and course labs etc. Furthermore, the system is capable to allocate exam venue to students, assign teacher and invigilator to the student, and assign courses and labs for the students.

Moreover, this system print report for each student in a separate file mentioning details of the students, such as current semester, courses, labs, GPA, total hours etc.

System Read ALL data from a given input file [input.txt] and generate result and reports in several output file like [StudentWrite.txt , Sa17001_Student_Report.txt, Za17002_ Student_Report.txt etc].

For a more detailed description of the system and commands, please follow the next section to understand how to develop the KAU Management System.

System Details

The Initial Procedure of the Program: Your program will use File I/O to read input from a given file name ***input.txt***. Make sure the file exist or display a message that the file does not exist. The first line of input file consists of ***6 integers*** to determine the array size of the Teacher, Invigilator, ExamVenue, Courselab, Course and Student respectively [***see the input file***]:

1) The first number (10) in the file refers to the number of ***Teachers*** in the System [means system will accept ONLY TEN teachers records details]
2) The second number (5) refers to the number of ***Invigilator*** in the system [means system will accept ONLY FIVE invigilators records details]
3) The third number (20) refers to the number of ***ExamVenues*** in the system [means system will accept ONLY TWENTY examvenues records details]
4) The fourth number (10) refers to the number of ***Course lab*** in the system [means system will create ONLY TEN course lab records]
5) The fifth number (10) refers to the number of ***Course*** in the system [means system will create ONLY TEN course records details]
6) The sixth number (25) refers to the number of ***Students*** in the system [means system will accept ONLY TWENTY FIVE Students records details]

1.1 Command: Add_Teacher

This command adds a new teacher to the system. The command will be followed by the following information, ALL on the same line:

Command Example
Add_Teacher PhD IT 13 Associate_Professor 478 false 51478 Alaa_Ahmad_Zahrani Saudi 1973 10 12 M 534528754 Rehab

Field name	Type
degree	String
department	String
teachingHours	double
jobTitle	String
officeNumber	int
onLeave	Boolean
id	int
name	String
nationality	String
Year *(of birth)*	int
Month *(of birth)*	int
Day *(of birth)*	int
gender	char
phone	int
address	String

Note: Each teacher record must be saved in the StudentWrite.txt file as per given sample ***StudentWrite.txt*** file.

1.2 Command: Add_Invigilator

This command adds a new invegilator to the system. The command will be followed by the following information, ALL on the same line:

Command Example
Add_Invigilator 2 Remote_Invigilation Secretary 124 false 55475 Ali_Alghamdi Saudi 1990 5 8 M 547871054 Kandara

Field name	Type
invigilatonExperienceYears	int
invigilatonSkill	String
jobTitle	String
officeNumber	int
onLeave	boolean
id	int
name	String
nationality	String
Year *(of birth)*	int
Month *(of birth)*	int
Day *(of birth)*	int
gender	char
Phone	int
address	String

Note: Each invegilator record must be saved in the StudentWrite.txt file as per given sample ***StudentWrite.txt*** file.

1.3 Command: Add_ ExamVenue

This command adds a new Exam Venue to the system. The command will be followed by the following information, ALL on the same line:

Command Example
Add_Examvenue 1001 Second 31

Field name	Type
examvenueNo	int
floor	String
buildingNo	String

Note: Each exam venue record must be saved in the StudentWrite.txt file as per given sample ***StudentWrite.txt*** file.

1.4 Command: Add_CourseLab

This command adds a new Course Lab to the system. The command will be followed by the following information, ALL on the same line:

Command Example
Add_CourseLab 501 Graphics_Lab 1.5

Field name	Type
labID	int
name	String
hours	double

Note: Each Course Lab record must be saved in the StudentWrite.txt file as per given sample ***StudentWrite.txt*** file.

1.5 Command: Add_ Course

This command adds a new Course to the system. The command will be followed by the following information, ALL on the same line:

Command Example
Add_Course 201 Computer_Fundamentals 4.0

Field name	Type
courseID	int
courseTitle	String
hours	double

Note: Each Course record must be saved in the StudentWrite.txt file as per given sample ***StudentWrite.txt*** file.

1.6 Command: Add_Student

This command adds a new Student to the system. The command will be followed by the following information, ALL on the same line:

Command Example
Add_Student Computer_Science 4 3.54 2013 5 5 17001 Moukhled_AlOutaibi Saudi 1998 4 8 M 12412222 Jeddah 6 4

Field name	Type
department	String
semester	int
cgpa	double
Year *(of enrollment)*	int
Month *(of enrollment)*	int
Day *(of enrollment)*	int
id	int
name	String
nationality	String
Year *(of birth)*	int
Month *(of birth)*	int
Day *(of birth)*	int
gender	char
Phone	int
address	String
totalCourse	int
totalCourseLab	int

Note: Each student record must be saved in the StudentWrite.txt file as per given sample ***StudentWrite.txt*** file.

1.7 Command: Assign_Teacher_Student

This command will be used to assign a teacher to a student. The command will be followed by the following information, ALL on the same line:

Command Example
Assign_Teacher_Student 44254 17001

Field name	Type
teacherID	int
studentID	int

Note: Each Teacher-Student record must be saved in the StudentWrite.txt file as per given sample ***StudentWrite.txt*** file.

1.8 Command: Assign_ExamVenue_Student

This command will be used to assign an exam venue to a student. The command will be followed by the following information, ALL on the same line:

Command Example
Assign_Examvenue_Student 1005 17005

Field name	Type
ExamvenueNumber	int
studentID	int

Note: Each ExamVenue-Student record must be saved in the StudentWrite.txt file as per given sample ***StudentWrite.txt*** file.

1.9 Command: Assign_Invigilator_Student

This command will be used to assign an invigilator to a student. The command will be followed by the following information, ALL on the same line:

Command Example
Assign_Invigilator_Student 55475 17001

Field name	Type
invigilatorID	int
studentID	int

Note: Each Invigilator-Student record must be saved in the StudentWrite.txt file as per given sample ***StudentWrite.txt*** file.

1.10 Command: Assign_CourseLab_Student

This command will be used to assign a lab to a student. The command will be followed by the following information, ALL on the same line:

Command Examples
Assign_ CourseLab _Student 17001 501 503 506 509 Assign_ CourseLab _Student 17009 509 506

Field name	Type
studentID	int
labID	int

Important Note
We are already providing the maximum number of taken labs as **totalCourseLab** respectively with each **Add_Student** command to initialize the array size.

Note: Each Lab-Student record must be saved in the StudentWrite.txt file as per given sample ***StudentWrite.txt*** file.

1.11 Command: Assign_Course_Student

This command will be used to assign a course to a student. The command will be followed by the following information, ALL on the same line:

Command Examples
Assign_Course_Student 17001 201 391 403 351 202 353 Assign_Course_Student 17020 353 202 201

Field name	Type
studentID	int
courseID	int

Important Note
We are already providing the maximum number of taken courses as **totalCourse** respectively with each **Add_Student** command to initialize the array size.

Note: Each Course-Student record must be saved in the StudentWrite.txt file as per given sample ***StudentWrite.txt*** file.

1.12 Command: Print_Report

This command will have no other information on the line.
This command will be used to print ALL 25 students' complete report details in a separate file for each Student. Since there are twenty five students, therefore you have to generate twenty five student report files, each report in a separate file.

File name must be given as:

Student name's First two Letter + Student Id+"_Student_Report" + ".txt"

For example
Ab17002_Student_Report.txt

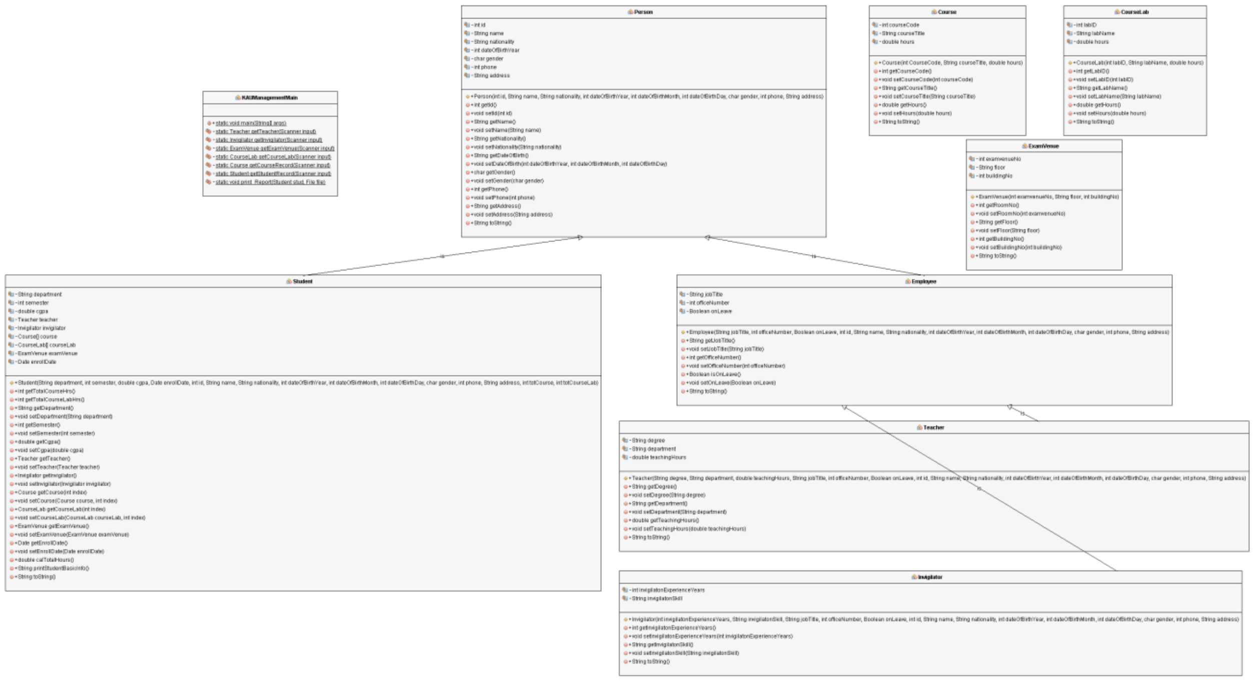
[***see ALL 25 Sample output Student Report files***]

Further Details are as follows:

You have to create **NINE classes** in this program.

- **Person** class is super class of ALL the classes of this assignment.
- **Employee** class is a sub-class of Person.
- **Teacher** class is a sub-class of Employee.
- **Invigilator** class is a sub-class of Employee.
- **Student** class is a sub-class of Person.
- **ExamVenue** class to store Examvenue details.
- **Course** class to store course details.
- **Courselab** class to store course lab details.
- **Tester class (BA1487412P3_KAUManagementSystem)** to create objects and invoke appropriate methods for program to execute successfully.

1.2 UML Class Diagram


KAUManagementMain
+static void main(String[] args)
-static Teacher getTeacher(Scanner input)
-static Invigilator getInvigilator(Scanner input)
-static ExamVenue getExamVenue(Scanner input)
-static CourseLab getCourseLab(Scanner input)
-static Course getCourseRecord(Scanner input)
-static Student getStudentRecord(Scanner input)
-static void print_Report(Student stud, File file)
Person
-int id
-String name
-String nationality
-int dateOfBirthYear
-char gender
-int phone
-String address
+Person(int id, String name, String nationality, int dateOfBirthYear, int dateOfBirthMonth, int dateOfBirthDay, char gender, int phone, String address)
+int getId()
+void setId(int id)
+String getName()
+void setName(String name)
+String getNationality()
+void setNationality(String nationality)
+String getDateOfBirth()
+void setDateOfBirth(int dateOfBirthYear, int dateOfBirthMonth, int dateOfBirthDay)
+char getGender()
+void setGender(char gender)
+int getPhone()
+void setPhone(int phone)
+String getAddress()
+void setAddress(String address)
+String toString()
Course
-int courseCode
-String courseTitle
-double hours
+Course(int CourseCode, String courseTitle, double hours)
+int getCourseCode()
+void setCourseCode(int courseCode)
+String getCourseTitle()
+void setCourseTitle(String courseTitle)
+double getHours()
+void setHours(double hours)
+String toString()
CourseLab
-int labID
-String labName
-double hours
+CourseLab(int labID, String labName, double hours)
+int getLabID()
+void setLabID(int labID)
+String getLabName()
+void setLabName(String labName)
+double getHours()
+void setHours(double hours)
+String toString()
ExamVenue
-int examvenueNo
-String floor
-int buildingNo
+ExamVenue(int examvenueNo, String floor, int buildingNo)
+int getRoomNo()
+void setRoomNo(int examvenueNo)
+String getFloor()
+void setFloor(String floor)
+int getBuildingNo()
+void setBuildingNo(int buildingNo)
+String toString()
is
is
Student
-String department
-int semester
-double cgpa
-Teacher teacher
-Invigilator invigilator
-Course[] course
-CourseLab[] courseLab
-ExamVenue examVenue
-Date enrollDate
+Student(String department, int semester, double cgpa, Date enrollDate, int id, String name, String nationality, int dateOfBirthYear, int dateOfBirthMonth, int dateOfBirthDay, char gender, int phone, String address, int totCourse, int totCourseLab)
+int getTotalCourseHrs()
+int getTotalCourseLabHrs()
+String getDepartment()
+void setDepartment(String department)
+int getSemester()
+void setSemester(int semester)
+double getCgpa()
+void setCgpa(double cgpa)
+Teacher getTeacher()
+void setTeacher(Teacher teacher)
+Invigilator getInvigilator()
+void setInvigilator(Invigilator invigilator)
+Course getCourse(int index)
+void setCourse(Course course, int index)
+CourseLab getCourseLab(int index)
+void setCourseLab(CourseLab courseLab, int index)
+ExamVenue getExamVenue()
+void setExamVenue(ExamVenue examVenue)
+Date getEnrollDate()
+void setEnrollDate(Date enrollDate)
+double calTotalHours()
+String printStudentBasicInfo()
+String toString()
Employee
-String jobTitle
-int officeNumber
-Boolean onLeave
+Employee(String jobTitle, int officeNumber, Boolean onLeave, int id, String name, String nationality, int dateOfBirthYear, int dateOfBirthMonth, int dateOfBirthDay, char gender, int phone, String address)
+String getJobTitle()
+void setJobTitle(String jobTitle)
+int getOfficeNumber()
+void setOfficeNumber(int officeNumber)
+Boolean isOnLeave()
+void setOnLeave(Boolean onLeave)
+String toString()
is
is
Teacher
-String degree
-String department
-double teachingHours
+Teacher(String degree, String department, double teachingHours, String jobTitle, int officeNumber, Boolean onLeave, int id, String name, String nationality, int dateOfBirthYear, int dateOfBirthMonth, int dateOfBirthDay, char gender, int phone, String address)
+String getDegree()
+void setDegree(String degree)
+String getDepartment()
+void setDepartment(String department)
+double getTeachingHours()
+void setTeachingHours(double teachingHours)
+String toString()
Invigilator
-int invigilatonExperienceYears
-String invigilatonSkill
+Invigilator(int invigilatonExperienceYears, String invigilatonSkill, String jobTitle, int officeNumber, Boolean onLeave, int id, String name, String nationality, int dateOfBirthYear, int dateOfBirthMonth, int dateOfBirthDay, char gender, int phone, String address)
+int getInvigilatonExperienceYears()
+void setInvigilatonExperienceYears(int invigilatonExperienceYears)
+String getInvigilatonSkill()
+void setInvigilatonSkill(String invigilatonSkill)
+String toString()

Zoom the file, if you are unable to see the above UML Class diagram.

Important Notes:

- Use of class & object, arrays of Object, passing object to method and Inheritance is mandatory.
- Use of Files, Reading/Writing from/on files and String , StringBuilder methods.
- Your program output must be exactly same as given sample output files.
- Your display should be in a readable form.
- Organize your code in separated methods.
- Repeat the program until command=Quit.
- Document your code with comments.
- Use meaningful variables.
- Use dash lines between each method.
- **Delayed submission will not be accepted and there will not be any extension of the project.**

Deliverables:

- You should submit one zip file containing all java codes: BA1587412P3_KAU.java where BA is your section, 1587412 your ID and P3 is program 3.
- **NOTE: your name, ID, and section number should be included as comments in all files!**

Input and Output Format

Your program must generate output in a similar format to the sample run provided.
Sample input: See sample input file.
Sample output: See sample output files.

Good Luck!