****

**King Abdulaziz University  
Faculty of Computing and Information Technology  
Fall 2021 – 1st – Term (2021)**

|  |  |
| --- | --- |
| **Course Code: CPCS 203** | **Course Name: Programming II** |

**Assignment #3 (Object Oriented Students Information System)**

|  |  |
| --- | --- |
| **Assigned Date** | **Sunday 07 /11/2021** |
| **Delivery Date and Time** | **Tuesday 23/11/2021 at 11:00 PM** |

**WARNING**:

* This program must ONLY be submitted on the Blackboard!
* This project worth 10% of the overall module marks (100%).
* NO assignment will be accepted after 11:59 pm for any reasons.
* Students can submit their assignments between 11 and 11:59 PM but in this case it, will be considered as late submission, and they will lose 2 points from the total mark of the assignment.
* For discussion schedule, check the teacher’s 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).

**Description**

**Uber System**

What is Uber System?

Uber System is simulation software to simulate a variety of Uber oriented functionalities both for rider and captain. The system is capable to add records for riders, captains, system administrators, rides, payment, and bonuses for rider. Moreover, the system has ability to allocate ride for the rider, associate captain and system administrator to the rider, and record payment and bonus for the rider.

Furthermore, this system print report for each rider in a separate file mentioning details of the rider, such as wallet balance, pickup/ drop location, distance covered, rating, total payment till date and so on.

The system read all data from a given input file [input.txt] and generate result in output file [RiderWrite.txt] and rider reports in several output file like [22001Ism\_Rider\_Report.txt, 22012Ham\_Rider\_Report.txt etc].

**More Details are as follows:**

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 exists or display a message that the file does not exist.

The file consists of 6 integers to determine the size of the Captain, UberAdmin, Ride, RiderBonus, Payment and Rider [see the input file for details]:

1. The first number (10) in the file refers to the number of **Captains** in the System [ means system will accept ONLY TEN captain records details]
2. The second number (5 ) refers to the number of **UberAdmins** in the system [ means system will accept ONLY FIVE UberAdmin records details]
3. The third number ( 20 ) refers to the number of **Rides** in the system [ means system will accept ONLY TWENTY rides records details]
4. The fourth number ( 10 ) refers to the number of **RiderBonus** in the system [ means system will create ONLY TEN rider bonus records]
5. The fifth number ( 10 ) refers to the number of **Payments** in the system [ means system will create ONLY TEN payment records details]
6. The sixth number ( 25 ) refers to the number of **Riders** in the system [ means system will accept ONLY TWENTY FIVE rider records details]

**The commands you will have to implement are as follows:**

* **Add\_Captain** – Creates a new captain which is added to the system. The command will be followed by the following information, ALL on the same line:

An int representing **drivingLicenseNo** of a captain; an int representing **totalRides** of a captain; a double representing **rating** of a captain; an int representing **empId** of a captain; a String representing **empDesc**; a Boolean representing **onLeave** flag; an int representing **id** of a captain; a String representing **name** of a captain; a String representing **email** of a captain; an int representing **Year** of a birth of captain; an int representing **Month** of birth of captain; an int representing **Day** of a birth of captain; a char representing **gender** of a captain; an int representing **Phone** of a captain; a string representing **address** of a captain; as shown in the underneath example. [see input.txt]

**Add\_Captain 32154254 155 4.2 3215 Full\_time true 501 Rida\_Ahmad\_Zahrani rida.a@gmail.com 1973 10 12 M 534528754 Rehab**

in above command

**drivingLicenseNo** **=** 32154254**, totalRides** **=** 155**, rating** **=** 4.2**, empId** **=** 3215 **empDesc =** Full\_time**,** **onLeave** **=** true**, id =** 501, **name =** Rida\_Ahmad\_Zahrani, **email =** rida.a@gmail.com**, Year =** 1973**, Month** **=** 10**, Day** **=**12 **,**  **gender** = M**, Phone** **=** 534528754**,** and **address =** Rehab

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

* **Add\_UberAdmin –** Creates a new UberAdmin which is added to the system. The command will be followed by the following information ALL on the same line:

an String representing **department** of an UberAdmin; a integer representing **empId** of an UberAdmin, a String representing **empDesc** of an UberAdmin; a boolean representing **onLeave** flag of an UberAdmin; an int representing **id** of a UberAdmin; a String representing **name** of a UberAdmin; an String representing **email** of a UberAdmin; an int representing **Year** of a birth of an UberAdmin; an int representing **Month** of birth of an UberAdmin; an int representing **Day** of birth of an UberAdmin; a char representing **gender** of a UberAdmin; an int representing **Phone** of a UberAdmin; a string representing **address** of a UberAdmin as shown in the underneath example. [see input.txt]

**Add\_UberAdmin OperationX 1021 Supervisor false 805 Basher\_Bandar basher@uber.com 1975 4 28 M 554803421 Kandara**

in above command

* **department** **=**OperationX**, empId** **=**1021**, empDesc** **=** Supervisor**, onLeave =**false**, id =**805**, name =** Basher\_Bandar**, email =** basher@uber.com**, Year** 1975, **Month** **=**4**, Day** **=2**8**, gender** =M**, phone =**554803421**, address =**Kandara

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

* **Add\_Ride –** Creates a new ride which is added to the system. The command will be followed by the following information ALL on the same line:

an integer representing **rideID**; a String representing **rideType** of a ride; an integer representing **noOfSeats** in the ride; a String representing **registrationNo** of n ride. [see input.txt]

**Add\_Ride 5001 UberX 4 LRA6452**

in above command

**rideID =**5001 **, rideType =** UberX**, noOfSeats =** 4, **registrationNo =** LRA6452

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

* **Add\_RiderBonus**

Creates a new RiderBonus which is added to the system. The command will be followed by the following information ALL on the same line:

an integer representing **bonusId** of aRiderBonus; a String representing **bonusDesc** of a RiderBonus; an double representing **bonusCredit** of a RiderBonus as shown in the underneath example. [see input.txt]

**Add\_RiderBonus 906 Monthly\_Promotion 2**

in above command

**bonusId =**906 , **bonusDesc =** Monthly\_promotion, **bonusCredit:** 2

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

* **Add\_Payment**

Creates a new Payment which is added to the system. The command will be followed by the following information ALL on the same line:

an integer representing **paymentId** of aPayment; a String representing **paymentDesc** of a Payment; an double representing **paymentAmount** of a Payment as shown in the underneath example. [see input.txt]

**Add\_Payment 305 Cash 15**

in above command

**paymentId** **=**305, **paymentDesc** **=**Cash, **hours =**15

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

* **Add\_Rider**

Creates a new Rider which is added to the system. The command will be followed by the following information ALL on the same line:

A double representing **walletBalance** of a Rider; a String representing **pickupFrom** of a Rider; a String representing **dropTo**; a double representing **distance** travelled; an int representing **Year** of joining of a Rider; an int representing **Month** of a joining of a Rider; an int representing **Day** of a joining of a Rider; a double of a Rider **rating** of a Rider, an int representing **id** of a Rider; a String representing **name** of a Rider; a String representing **email** of a Rider; an int representing **Year** of birth of a Rider; an int representing **Month** of a birth of a Rider; an int representing **Day** of a birth of a Rider; a char representing **gender** of a Rider; an int representing **Phone** of a Rider; a String representing **address** of a Rider; an int representing **nosOfPayment** of a Rider; and an int representing **nosOfBonus** of a Rider; as shown in the underneath example. [see input.txt]

**Add\_Rider 3.41 Balad Fayha 5.7 2016 5 27 3.8 22008 Hiba\_AlMasrahi nm.asrahi@gmail.com 1965 7 13 F 537874123 Madina 3 2**

**Note:** The last two parameters of the Add\_Rider command (**nosOfPayment** **=**3**, and nosOfBonus** **=**2**)** will be used to initialize the array size of Payment[] and RiderBonus[] arrays to 3 and 2 respectively. Later in the Assign\_Payment\_Rider and Assign\_RiderBonus\_Rider commands, we'll add 3 Payments to Payment[] array and 2 bonuses to the RiderBonus[] array.

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

* **Assign\_Captain\_Rider**

This command will be used to assigned a captain to a Rider [see input.txt]

**Assign\_Captain\_Rider 501 22001**

Captain with ID: 501 is assigned to Rider ID: 22001.

**Note:** Each Captain-Rider assigned action must be saved in the **RiderWrite.txt** file as per given sample RiderWrite.txt file.

* **Assign\_Ride\_Rider**

This command will be used to allocate a ride to a Rider [see input.txt]

**Assign\_Ride\_Rider 5004 22004**

Ride with rideID: 5004 is assigned to Rider ID: 22004.

**Note:** Each Ride-Rider allocation action must be saved in the **RiderWrite.txt** file as per given sample RiderWrite.txt file.

* **Assign\_UberAdmin\_Rider**

This command will be used to assigned a UberAdmin to a Rider [see input.txt]

**Assign\_UberAdmin\_Rider 803 22008**

UberAdmin with ID: 803 is assigned to Rider ID: 22008.

**Note:** Each UberAdmin-Rider allocation action must be saved in the **RiderWrite.txt** file as per given sample RiderWrite.txt file.

* **Assign\_RiderBonus\_Rider**

This command will be used to allocate Payment Lab to a Rider [see input.txt]

**Assign\_RiderBonus\_Rider 22001 901 902**

RiderBonus with ID 901 and 902 is granted to Rider ID: 22001

**Assign\_RiderBonus\_Rider 22007 907 903 904 905**

RiderBonus with ID 907, 903, 904 and 905 is granted to Rider ID: 22007

**Note:** We are already providing the maximum number of taken labs as **nosOfBonus** respectively with each **Add\_Rider** command to initialize the array size.

**Note:** Each RiderBonus-Rider association action must be saved in the **RiderWrite.txt** file as per given sample RiderWrite.txt file.

* **Assign\_Payment\_Rider**

This command will be used to allocate Payment to a Rider [see input.txt]

**Assign\_Payment\_Rider 22001 301 302 303**

Payment with ID 301, 302, and 303 is paid by Rider ID: 22001

**Assign\_Payment\_Rider 22016 306 302 303 304**

Payment with ID 306, 302, 303 and 304 is paid by Rider ID: 22016

**Note:** We are already providing the maximum number of taken Payments as **nosOfPayment** respectively with each **Add\_Rider** command to initialize the array size.

**Note:** Each Payment- Rider association action must be saved in the **RiderWrite.txt** file as per given sample RiderWrite.txt file.

* **Print\_Report**

This command will have no other information on the line.

This command will be used to print ALL 25 Riders’ complete report details in a separate file for each Rider. Since there are twenty five Riders, therefore you have to generate twenty five Rider report files, each report in a separate file.

file name must be given as

**Rider Id + Rider First 3 Letters + "\_Rider\_Report" + ".txt"**

for example

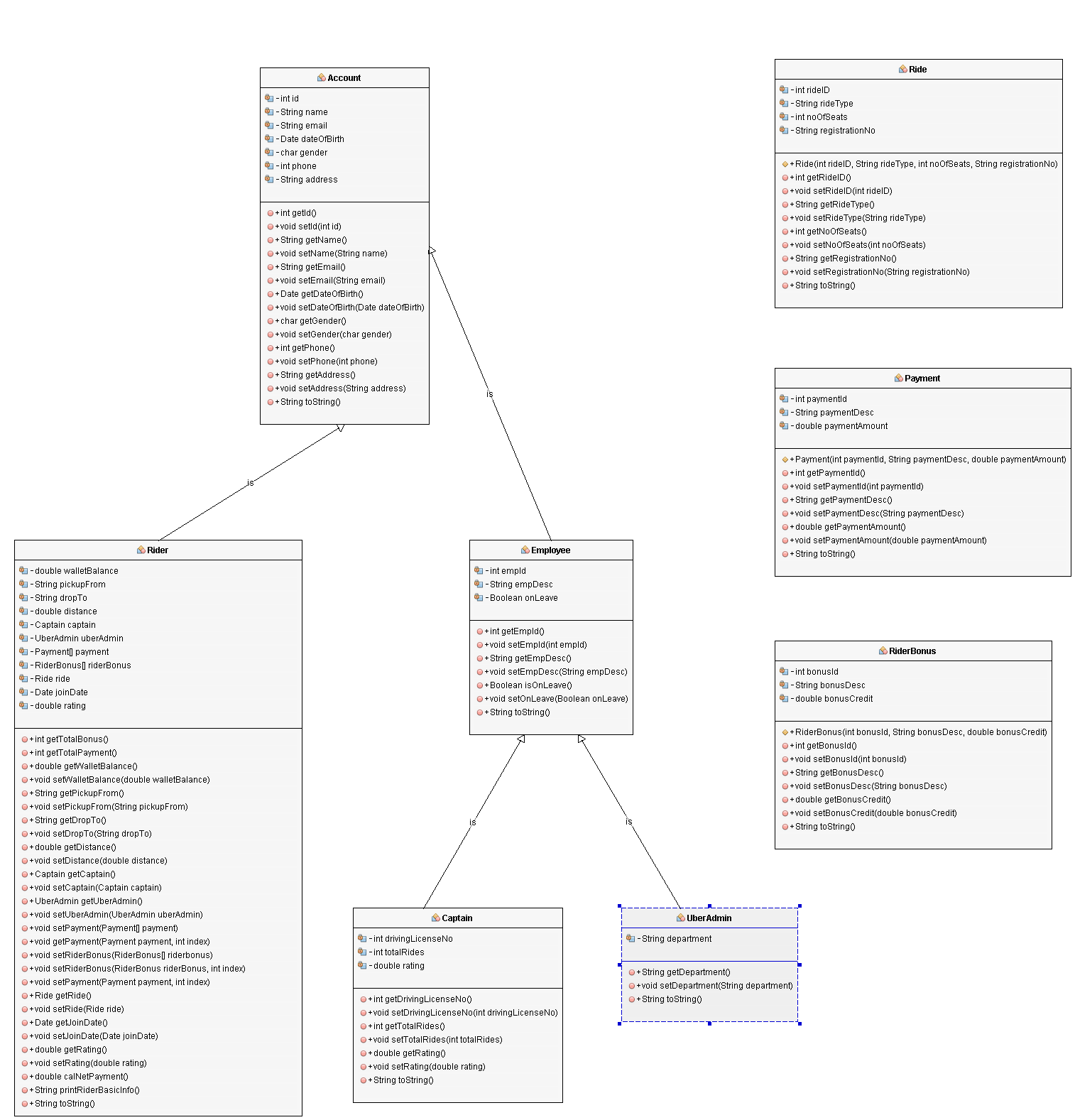
22001Ism\_Rider\_Report

[see ALL twenty five Sample output Rider Report files ]

**Further Details are as follows:**

You must create **SEVEN classes** in this program.

* **Account** class is super class of ALL the classes of this assignment.
* **Employee** class is a sub-class of **Account**.
* **Captain** class is a sub-class of **Employee**.
* **UberAdmin** class is a sub-class of **Employee**.
* **Rider** class is a sub-class of **Account**.
* **Ride** class to store Ride details.
* **Payment** class to store Payment details.
* **RiderBonus** class to store Rider Bonus details.
* **Tester class** (**BA1487412P3**\_**UberSystem**) to create objects and invoke appropriate methods for program to execute successfully.
* See the **UML Diagram underneath** to know basic class details. UML class diagram is provided to understand the concepts. Minor changes are allowed!



**UML Diagram of Uber System**

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

**Important Notes:**

* Use of class & object, arrays of Object or ArrayList, 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).**

**Deliverable**

You have to submit only the java file of your code. The file and the class names should be “**Project\_3\_YourFirstName\_YourUID**”. Where “YourFirstName“ is your first name, “YourUID“ is your university ID.

**NOTE:** your name, ID, and section number must be included as comments in the file!

**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 All sample output files.**

**Good Luck and Start Early!**