Create the classes, as specified, and write Java code for a vehicle rental system. The company rents both automobiles and boats (5 each). However, they have different rental fees – the fee to rent a boat is **$115/1hr, $225/2hr, $345/4hr, $495/8hr**, and the fee to rent an automobile is **$60/day + 25 cents/mile for all miles over 500 (maximum days - 7, maximum miles - 5000).**

The code for ***each*** class should be well documented. Java documentation should be specified at the top of each class describing the class and its functionality (see lab 0 for an example) – be sure to include the **@author** block tag for your name.
In addition, Java documentation should be specified before each method header describing the purpose of the method –, **@param** and **@return** block tags should indicate the return types and parameters expected by the method.
Your program must display correct output with no runtime errors and must be well-documented, in order to receive full credit. You will only receive credit for submission, programming style, and documentation criteria, if the code for the classes relate to the program specified.

Projects Grading Rubric:

Submission Criteria

Correct project submission zip file, correctly-named project (as specified) in zip file, on time	5%

Programming Style Criteria

Appropriate choice of variable names and data types	5%
Readability/Neatest (code indentation)	5%

Program Documentation Criteria

Header comment (Description, author, date due, etc.)	5%
Documentation (class and method documentation as necessary)	10%

Program Correctness Criteria

Correct algorithm/logic, computes without run-time errors	15%
Correct results	40%
Output formatting quality	5%
Efficiency	10%

Classes for the Project:

Define a ***Vehicle class*** ***(base class)*** with the following properties:

Attributes: ***manufacturer*** *(string),* ***model*** *(string, the first letter, A or B, indicates the type of vehicle),* ***customerId*** *(String),* ***available*** (*protected boolean* indicating whether or not the vehicle is available)
count (*static int* for the number of vehicles)

Methods:

A *Default Constructor & a parameterized Constructor* to initialize the customer Id, manufacturer and model.
Accessors to return the availability, manufacturer, model and the customer Id.
Modifiers to change the manufacturer, model and the customer Id, availability (**protected**)
A method (*rentVehicle*) to rent a vehicle. **Note:** the *customerId* is required to rent a Vehicle.
An abstract method (*returnVehicle*) to compute the fee when the customer returns the Vehicle. The *hours* are required to return a boat, and the *days* and *milesDriven* are required to return an automobile.
A *toString* method to return the state of a *Vehicle* object.

Define an ***Automobile class*** - subclass derived from the ***Vehicle class*** with the following properties:

Attributes: `int mileage, static count`

Methods:

A *Default Constructor & a parameterized Constructor* to initialize the manufacturer and model.
A Method (*returnVehicle*) to compute the fee when the Automobile returns.
Note: *milesDriven* is used to update the mileage, when an Automobile is returned.
A *toString* method to return the state of an Automobile object.

Define a ***Boat class*** - subclass derived from the ***Vehicle class*** with the following properties:

Attributes: `static count`

Methods:

A *Default Constructor & a Constructor* to initialize the manufacturer and model.
A Method (*returnVehicle*) to compute the fee when the Boat returns.
A *toString* method to return the state of a Boat object.

Define a ***VehicleApp class*** ***(application class)*** with the following properties:

Attributes: `Vehicle[] v`

Methods:

`getVehicles(), transactions()`

PHASE I: Algorithm to solve the problem

PHASE II: Define UMLs for each class (Vehicle, Automobile, Boat)
Create Pseudocode
Write Code for each class (Vehicle, Automobile, Boat)

PHASE III: Create the code for the application class (VehicleRentalApp):

```
Define a reference to an array of Vehicle objects
Open a file (I:\grahamf\cmp211\vehicles.txt) and store the vehicles in the array.

Open a file (I:\grahamf\cmp211\rental.txt) and process each rental transaction.
Display the availability of each Vehicle after every transaction
```

Possible Pseudocode:

```
======== main(String[] args) ===========
          Instantiate a VehicleRental object
          create array of Vehicles - v = new Vehicle[10];
          call getVehicles method  - getVehicles();
          call rentVehicles method - transactions()

======== void getVehicles() ===========
Purpose is to read data from the vehicles text file, create objects
(Boat or Automobile) and store them in the array of Vehicles

Open data file
while the file has more data {

     read a line and split the data (delimited by comma)
     if valid create the appropriate object and store it in
     the vehicle array and increment the count of valid vehicles
     else display a message indicating invalid data

}
```

```
======== void transactions() ===========
Open data file

read file
while the file has more {
     read a line and split the data (delimited by comma)

     if error occurs
          display message, end
     else
          get id and transaction
          if transaction is rent
               if auto
                    check the availability
                    if the availability is true
                         rent the vehicle
                         display information
                    else
                         display message: auto unavailable
               else if boat
                    check the availability
                    if the availability is true
                         rent the vehicle
                         display information
                    else
                         display message: boat unavailable
               else
                    display message: invalid vehicle type
          else
               if transaction is return
                    get amount
                    check each vehicle
                    if availability is false and customer id matches
                         return vehicle
                         display information including cost
                    if customer can't be found
                         display message: Vehicle was not rented
               else
                    display message: Invalid transaction type
}
```

Sample Output:

```
Reading vehicles data file...
Vehicle [id=A1111, manufacturer=Ford, model=Taurus, available=true,
renter=null][Automobile [mileage=0 count: 1]
Vehicle [id=A2222, manufacturer=Nissan, model=Maxima, available=true,
renter=null][Automobile [mileage=0 count: 2]
Vehicle [id=A3333, manufacturer=Honda, model=Accord, available=true,
renter=null][Automobile [mileage=0 count: 3]
Vehicle [id=A4444, manufacturer=Toyota, model=Camry, available=true,
renter=null][Automobile [mileage=0 count: 4]
Vehicle [id=A5555, manufacturer=Chevrolet, model=Malibu, available=true,
renter=null][Automobile [mileage=0 count: 5]
Vehicle [id=B1111, manufacturer=Cobalt, model=200s, available=true, renter=null][count: 1]
Vehicle [id=B2222, manufacturer=Cobalt, model=CS22, available=true, renter=null][count: 2]
Vehicle [id=B3333, manufacturer=Boston Whaler, model=230 Outrage, available=true,
renter=null][count: 3]
Vehicle [id=B4444, manufacturer=Stingray, model=182SC Deck Boat, available=true,
renter=null][count: 4]
Vehicle [id=B5555, manufacturer=Sea Ray, model=SDX 220, available=true,
renter=null][count: 5]

Reading transactions data file...
Renting vehicle...
Rented: Vehicle [id=A1111, manufacturer=Ford, model=Taurus, available=false,
renter=C0010][Automobile [mileage=0 count: 5]

Renting vehicle...
Rented: Vehicle [id=A2222, manufacturer=Nissan, model=Maxima, available=false,
renter=D0055][Automobile [mileage=0 count: 5]

Renting vehicle...
Rented: Vehicle [id=B1111, manufacturer=Cobalt, model=200s, available=false,
renter=B0029][count: 5]

Returning automobile...
Returned Automobile: Vehicle [id=A1111, manufacturer=Ford, model=Taurus, available=true,
renter=C0010][Automobile [mileage=340 count: 5]
Days : 1     Miles: 340   Cost: $60.00

Returning automobile...
Error: No record found for that auto rental...

Returning boat...
Returned Boat: Vehicle [id=B1111, manufacturer=Cobalt, model=200s, available=true,
renter=B0029][count: 5]
Hours: 4     Cost: $345.00

Returning automobile...
Returned Automobile: Vehicle [id=A2222, manufacturer=Nissan, model=Maxima, available=true,
renter=D0055][Automobile [mileage=450 count: 5]
Days : 2     Miles: 450   Cost: $120.00

Invalid transaction...

Returning automobile...
Error: No record found for that auto rental...
```