**Project 2: Vehicle Information System**

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| In this project you will define some interfaces, abstract classes, and concrete classes, all placed in a specific package. You will also use the instanceof operator. Note: if you do not use interfaces, abstract classes, and/or instanceof, you will not receive full credit. Submit to user cs1302a on odin a directory called Project2 (using the following command: submit Project2 cs1302a):   * All the source code, i.e. .java files. You do not have to include .class files.  Comments are *required* in your source code! These comments should make it easy to read and follow the logic of your code.  Also, your source code should be well formatted. * A README.txt file telling us how to compile your program and how to execute it.   **Project description:**   1. Your program must be in package edu.uga.cs1302.vehicles. 2. Define an interface called Transporter with the following methods:  |  |  | | --- | --- | |  | int getMaxPassengers()  void setMaxPassengers(int maxPassengers)  int getTopSpeed()  void setTopSpeed(int topSpeed) |  1. Define an interface called Flyable with methods:  |  |  | | --- | --- | |  | int getMaxAltitude()  void setMaxAltitude(int maxAltitude)  int getMaxRange()  void setMaxRange(int maxRange) |  1. Define an interface called Floatable with methods:  |  |  | | --- | --- | |  | int getTonnage()  void setTonnage(int tonnage) |  1. Define an abstract class called Vehicle which must implement the Transporter interface. Add the representation of:    * Vehicle's *name*,    * Vehicle's *manufacturer*,    * Vehicle's *year of manufacture, and*    * *count of the instances* of the Vehicle class (using a static variable).   Use suitable qualifiers for all fields in the class.   1. Define a class called Automobile, where an Automobile *is-a* subclass of Vehicle.  Also, an Automobile has an engine with a given *horsepower*. 2. Define a class called Airplane, a subclass of Vehicle. Also, this class should implement the Flyable interface.  Furthermore, an Airplane has a given *number of engines*. 3. Define a class called Ship, a subclass of Vehicle.  Also, this class should implement the Floatable interface. Also, a Ship has a given *shipping line owner*. 4. Be creative and find more Vehicle kinds (at least *two* more). One of these new vehicle types (can be imaginary) should be able to both *fly* and *float*, and should be a direct subclass of Vehicle (for example, a *flying boat*, such as the PBY Catalina, manufactured years ago by Consolidated Aircraft). 5. Create three vehicles of each type, with different names and suitable attribute values (you may use fictitious values). 6. Store the Vehicle objects (above) in an array of type Vehicle. 7. Display a menu to the user on the screen (before implementing this part, read the notes below).    * Press 1 to see how many vehicles are in the system.    * Press 2 to see the name and kind of each vehicle.    * Press 3 to see which vehicles can fly.    * Press 4 to see which vehicles can float.    * Press 5 to see which vehicles can float AND fly.    * Press 6 to see a description of each vehicle.    * Press h to see a brief help information for your system.    * Press q to terminate the program. 8. Important notes about the menu:  * If the user enters anything other than {1, 2, 3, 4, 5, 6, h, q}, the program should print a proper error message, and ask the user to enter a number between {1, 2, 3, 4, 5, 6} or the letters  'h' (for help) or  'q' (to quit). In other words, your program should not continue (or even worse, crash) if the user enters a wrong choice. * In option 1, the result should be the total number of instances of Vehicle class created so far (use a static variable). * For option 2, use instanceof to get kind. * For option 3, use instanceof. The result should contain both name (e.g., Mustang Shelby GT350R, Queen Mary 2, Boeing 787) and type (Automobile, Ship, etc) of the vehicle. * For option 4, use instanceof. The result should contain both name and type of the vehicle. * For option 5, use instanceof. The result should contain both name and type of the vehicle. * Option 6 above could display all attribute values for a given object.  Note that the collection of properties should be suitable for a given type of a vehicle. For example:  |  |  | | --- | --- | |  | Type: Automobile  Name: Mustang Shelby GT350R  Manufacturer: Ford  Manufacture year: 2017  Engine power: 526 hp  Max passengers: 2  Top speed: 175 mph | |  | Type: Airplane  Name: Boeing 787  Manufacturer: Boeing  Manufacture year: 2014  Number of engines: 2  Max altitude: 43000 ft  Max range: 8500 mi  Max passengers: 280  Top speed: 587 mph | |  | Type: Ship  Name: Queen Mary 2  Manufacturer: Chantiers de l'Atlantique  Manufacture year: 2004  Tonnage: 148528 tons  Owner: Cunard Line  Max passengers: 2620  Top speed: 34 mph |   **Things to note:**   * Your program *must* be in package edu.uga.cs1302.vehicles. * You must provide setters and getters for all instance variables in classes. * Follow good coding style (proper variable names, indentation, etc). * Your design should be reasonably efficient, and in accordance with object oriented design principles (encapsulation, information hiding, inheritance, etc.). * If you are defining a variable as public or protected, briefly mention the reason for that. * If you are defining a variable as static or final, briefly mention the reason for that, as well. |