Advanced Programming Using Java - Lesson 4 (Chapter 2 Part B)

GOALS: Learn about and use public & static keywords, methods, and scope

Add an overloaded constructor (an overloaded method is one with different parameters—the default constructor created above has no parameters, and the overloaded one below has three). Copy&paste the constructor then make an overloaded constructor by changing it to look like the one below:



Add a method that determines damage:

* After the vehicle constructor in Vehicle.java, add a static method (a static method doesn’t require an object in order to be used, so it is great for affecting two at once and ensuring they exist):



(notice that the value -1 is returned only if one of the objects is null due to a programming error. Therefore, in the code above, -1 could be called a “flag” value since it signals to the programmer that a certain situation has occurred)

* + Click the bulb by the Random line to add import for java.util.Random

Go back to the MainForm file. In the MainForm’s Source tab, add the variables that create the ships (when you create code that explains the actual data used in the view, that is called creating the *model*). Put these variables in class MainForm (such as before the constructor public MainForm), using the overloaded constructor to define the ships:

(You can just type the first statement then copy it and change 1 to 2 for the object names, displayName, and for each of the 3 widget names. The statement is split into 4 lines just so it fits in the recommended width [red line] to make editing easier)



Also add a model for the list so we can control the view from our program (the log will display all of the events in the simulation):



* By DefaultListModel click the bulb then import java.swing.DefaultListModel
* Double-click each ship variable and logModel then click the bulb then “move initializer to Constructor”
* In the MainForm constructor, you should now see both of those “new” statements. Move the initComponents() line so it comes first (before both “new” statements), so that the constructors will not cause a crash by trying to use values from the form before they exist.
* Add another statement after initComponents that will allow the form to use the data from the new model:



Advanced Programming Using Java - Lesson 4 (Chapter 2 Part B)

Now make an attack method inside the MainForm class (such as after the MainForm constructor) that will use the damage. Notice that Vehicle (capitalized) is used instead of a vehicle object, since the getDamage method is static. Notice that the attack message will be displayed even if there is no damage, just to indicate to the player that they did try to attack. Notice that parry is disabled no matter who attacks, so taking evasive action is nullified by being targeted or by attacking, but parrying becomes an available choice (make sure you look at the screen while typing, because the parameters vTarget and vAttacker will be filled in automatically):



Now call the attack method when the buttons are pressed:

* Go back to Design view
* Double-click Ship 1’s Attack button (if you don’t see the ship1AttackbuttonActionPerformed method, scroll upward) then add the following line: 
* Go back to Design View
* Double-click Ship 2’s attack button then add the same line but switch which ship is first (the target)
* Go back to Design View
* Double-click Ship 1’s Parry button then add the following:



* Go back to Design View
* Double-click Ship 2’s parry button and add the same code except change the ship1 to ship2