Name:									

ISTE-120 Lab 11: Inheritance

Exercise 1 – Creating the super class (3 points)

Step 1: Create a Boat class that contains information and methods common to all boats. It will be used as a super class in subsequent tasks. It has the attributes and methods shown in the table below. Note carefully, that the parameterized constructor, does <u>not</u> make direct assignments to instance variables (attributes). Rather, it must initialize them using the appropriate mutators.

Attributes	Data type
color	A String value for the boat color.
length	An int value for the boat length.
Method	Description
Boat	Default constructor that initializes color to white and length to 20.
Boat	A constructor with 2 parameters: a string value for the boat color
	that initializes color using setColor, and an integer value for the boat
	length that initializes length using setLength.
setColor	A mutator that tests the input parameter to insure it is white, red,
	blue or yellow. If so, it sets color and returns true. For all other
	values, color is not set and false is returned.
getColor	An accessor that returns the color.
setLength	A mutator that tests the input parameter to ensure it is in the range
	of 20 to 50 inclusive. If so, it sets length and returns true . For all
	other values, length is not set and false is returned.
getLength	An accessor that returns the length.
toString	Returns a string using String.format: Color = color Length = length

Although not strictly necessary, writing a TestBoat test class to verify the design of Boat is recommended. Note in the sample output below, each mutator was tested with both good and bad data.

Step 2: Create a SailBoat class using the Boat class as the super class and having the following attributes and methods:

Attributes	Data type
numSails	An int that holds the number of sailboat sails
Method	Description
SailBoat	The Boat class default constructor that sets the numSails to 1
SailBoat	Constructor with 3 parameters: a String for the boat color, an int for
	the boat length and an int for the number of sails. The color and the
	length are passed to the Boat class 2-parameter constructor, while
	the number of sails is initialized using the setNumSails mutator.
setNumSails	A mutator that tests the input parameter to ensure it is in the range
	of 1 to 4 inclusive. If so, it sets numSails and returns true . For all
	other values, numSails is not set and false is returned.
getNumSails	Returns the int value of numSails
calcPrice	Calculates the price of the SailBoat as follows:
	length * 1000 + numSails * 2000
toString	Returns a string using String.format: super.toString() + "Number of
	Sails = " + numSails + " Price = " + calcPrice

Step 3: Create a test class, TestSailBoat, that creates an object of the SailBoat class and prints the output of the toString method. Test the mutators for errors and print appropriate error messages.

Command Pro	ompt								
dkpvcs> java Sailboat: Color = red		Number Sails = 2	Price = \$ 29,000.00						
Changing Sailboat's color to "purple" Error: Boats can only be white, red, blue and yellow. Setting Boat's color to "yellow"									
Changing Sailboat's length to 55. Error: Boats can only be between 20 and 50 feet, inclusively. Changing Boat's length to 50.									
Sailboat: Color = yello	w Length = 50	Number Sails = 2	Price = \$ 54,000.00						
dkpvcs>									

Exercise 2 – Create another subclass (3 points)

Step 1: Create a PowerBoat as a subclass of the Boat class with the following attributes and methods:

Attributes	Data type
engineSize	An int that holds the engine size as engine horsepower
Method	Description
PowerBoat	Default constructor that calls the default constructor of the
	Boat class and sets the size of the engine to 5.
PowerBoat	Constructor with 3 parameters: a String for the boat color, an
	int for the boat length and an int for the engine size
	(horsepower). The color and the length are passed to the 2-
	parameter constructor in the Boat class. The engineSize is
	initialized using setEngineSize.
setEngineSize	Returns true if input is in the range 5 – 350. For all other
	values, do not set engineSize and return false.
getEngineSize	Returns the size of the engine.
calcPrice	Calculates the price of the PowerBoat as follows:
	5000 + length * 300 + engineSize of * 20
toString	Returns the string using String.format: super.toString +
	"Engine Size = " + engineSize + " Price = " + calcPrice.

Step 2: Create a TestPowerBoat that will create an object of the PowerBoat class and print the output of the toString method. Test the mutators for errors and print appropriate error messages. Sample output is as follows:

Command Prompt									
dkpvcs> java TestPowerBoat Powerboat:									
Color = yellow Length = 20 Engine Size =	20 Price = \$ 11,400.00								
Changing Powerboat's color to "purple" Error: Boats can only be white, red, blue and yellow. Setting Boat's color to "white"									
Changing Powerboat's engine size to 500. Error: That engine is too powerful. The engine size must be between 1 and 350, inclusively. Changing PowerBoat's engine size to 350.									
Powerboat:									
Color = white	350 Price = \$ 18,000.00								
dkpvcs>									

Exercise 3 – Create a Collection (Inventory) class (4 points)

Step 1: Create an Inventory test class. Begin by creating an ArrayList that holds objects of any type of Boat. Add the sailboats and powerboats listed below to the collection. The boats can be defined directly with a constructor eliminating the need to write a substantial amount of code that would be required to read in the values of the boats' attributes.

- A 22 ft. blue power boat with a 60 horsepower engine
- An 18 ft. white sail boat with 1 sail
- A 42 ft. red sail boat with a 3 sails
- A 35 ft. yellow power boat with an 80 horsepower engine
- A 50 ft. red power boat with a 120 horsepower engine
- A 33 ft. blue sail boat with 2 sails
- A 14 ft. white power boat with a 10 horsepower engine

A for/each loop must be used to print out all of the sailboats and powerboats as shown below:

```
Command Prompt
dkpvcs> java Inventory
Printing all boats:
Color = blue Length = 22
                                   Engine Size = 60
                                                             Price = $ 12,800.00
Color = white Length = 20 Number Sails = 1
Color = red Length = 42 Number Sails = 3
Color = yellow Length = 35 Engine Size = 80
                                                             Price = $ 22,000.00
                                                             Price = $ 48,000.00
                                                             Price = $ 17,100.00
Engine Size = 120
                                                             Price = $ 22,400.00
Color = blue Length = 33
                                   Number Sails = 2
                                                             Price = $ 37,000.00
Color = white Length = 20
                                                             Price = $ 11,200.00
                                   Engine Size = 10
```

Step 2: Add functionality to the test that calculates the total price of all sailboats and powerboats in the collection, finds the most expensive boat and prints the toString information about that boat.

Note: A String.format statement applied to all toString returns is required in order to insert commas in the appropriate position.

Signature:	
Jigilatai C.	