Worksheet W.A.13.1 Name_____

Inheritance

1. Find the output to the following program. Note that the toString() method overriding is handled differently for the Car and Truck classes.

See page 3 for exercise #2

```
import apcslib.*;// for formatting

class Vehicle
{
  protected String myBrand;       // brand name
  protected int myPrice;          // price of vehicle
  protected double myGasMileage;  // vehichle's gas mileage
  protected double myGasPrice;    // gas price per gallow
  protected int myYearlyMiles;    // miles driven per year

  // constructor
  public Vehicle(String brand, int price, double mileage,
      double gasPrice, int miles)
  {
    myBrand = brand; myPrice = price; myGasMileage = mileage;
    myGasPrice = gasPrice; myYearlyMiles = miles;
  }

  public String toString()
  {
    return "Your $" + myPrice + " " + myBrand + " costs $"
            + Format.right(myYearlyMiles / myGasMileage * myGasPrice, 6, 2)
            + " in gas each year!";
  }
}

//-------------------- End of Vehicle class --------------------//


class Car extends Vehicle
{
  protected String myCarType;     // type of car
  protected int myNumPassengers;  // number of passengers
  protected int myNumDoors;       // number of car doors

  // constructor
  public Car(String brand, String type, int price, double mileage,
      double gasPrice, int miles, int passengers, int numDoors)
  {
    // uses Vehicle's constructor
    super(brand, price, mileage, gasPrice, miles);

    // initializes what's new to Car
    myCarType = type; myNumPassengers = passengers; myNumDoors = numDoors;
  }

  // overloads toString() method
```



```
  public String toString()
  {
    return "Your $" + myPrice + " " + myBrand + " " + myCarType
           + " costs $" + Format.right(myYearlyMiles / myGasMileage
           * myGasPrice, 6, 2) + " in gas each year!\nIt has " + myNumDoors
           + " doors and carries " + myNumPassengers + " passengers.";
  }
}

//-------------------- End of Car class --------------------//


class Truck extends Vehicle
{
  protected String myTruckType;   // type of truck
  protected int myHaulingPounds;  // hauling capacity
  protected int myTowingPounds;   // towing capacity

  // constructor
  public Truck(String brand, String type, int price, double mileage,
      double gasPrice, int miles, int haul, int tow)
  {
    // uses Vehicle's constructor
    super(brand, price, mileage, gasPrice, miles);

    // initializes what's unique to Truck
    myTruckType = type; myHaulingPounds = haul; myTowingPounds = tow;
  }

  // overloads toString() method() using super
  public String toString()
  {
    return super.toString() + "\nThis " + myTruckType + " can carry "
           + myHaulingPounds + " pounds and can tow " + myTowingPounds
           + " pounds.";
  }
}

//-------------------- End of Truck class --------------------//


class Driver
{
  public static void main (String args[])
  {
    Vehicle standard = new Vehicle("Ford", 20000, 23, 1.95, 15000);
    System.out.println(standard.toString());
    // The previous line could have been written as follows:
    // System.out.println(standard);
    // When just the object is used with println(), the toString() method is
    // automatically called.
    System.out.println();
    Car family = new Car("Lexus", "convertible", 45000, 24, 2.05, 15000, 5, 4);
    System.out.println(family.toString());
    System.out.println();
    Truck rig = new Truck("Chevy", "4X4", 32000, 12, 1.95, 15000, 1500, 10000);
    System.out.println(rig.toString());
  }
}
```


2. Write your own SportsCar class that extends the Car class. This new class should have the following unique protected instance variables: myColor, MyEngine that stores its engine size in liters, mySuspension - e.g., firm, soft, touring, etc., and myTires - e.g., regular, wide, low profile, etc. To utilize this SportsCar class, add the following lines to the Driver class. Note that the last four parameters ("red", 3.0, "firm", and "low profile") are the unique additions to the SportsCar class. It may not be practical or realistic to use so many parameters to instantiate a single object; however, this exercise gives you practice working with multiple generations of inheritance.

```
SportsCar sporty = new SportsCar("Porsche", "Boxster", 55000, 19, 2.05,
15000, 2, 2, "red", 3.0, "firm", "low profile");
System.out.println(sporty.toString());
```

The resulting output from adding the SportsCar class and this code to the Driver class should look as follows:

```
Your $55000 Porsche Boxster costs $1618.42 in gas each year!
It has 2 doors and carries 2 passengers.
Your cool red sports car has a 3.2 liter engine, firm suspension and low
profile tires.
```

