Worksheet A11.1

Inheritance Review

1. Find the output of the driver class. Note that the toString() method overloading is handled differently for the Car and Truck classes.

```
public class Vehicle{
  private String myBrand;       // brand name
  private int myPrice;          // price of vehicle
  private double myGasMileage;  // vehicle's gas mileage
  private double myGasPrice;    // gas price per gallon
  private int myYearlyMiles;    // miles driven per year
  Formatter formatter = new Formatter();

  // 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 getBrand(){
    return myBrand;
  }

  public int getPrice(){
    return myPrice;
  }

  public double getGasMileage(){
    return myGasMileage;
  }

  public double getGasPrice(){
    return myGasPrice;
  }

  public int getYearlyMiles(){
    return myYearlyMiles;
  }

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

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

```
public class Car extends Vehicle{
  private String myCarType;     // type of car
  private int myNumPassengers;  // number of passengers
  private 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;
  }

  public String getCarType(){
    return myCarType;
  }

  public int getNumPassengers(){
    return myNumPassengers;
  }

  public int getNumDoors(){
    return myNumDoors;
  }

  // overloads toString() method
  public String toString(){
    return "Your $" + getPrice() + " " + getBrand() + " "
    + myCarType + " costs " + formatter.format("$%.2f", getYearlyMiles()
    / getGasMileage() * getGasPrice())
    + " in gas each year!\nIt has " + myNumDoors + " doors and carries "
    + myNumPassengers + " passengers.";
  }

}

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


public class Truck extends Vehicle{
  private String myTruckType;   // type of truck
  private int myHaulingPounds;  // hauling capacity
  private 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;
  }

  public String getTruckType(){
    return myTruckType;
  }

  public int getHaulingPounds(){
    return myHaulingPounds;
  }

  public int getTowingPounds(){
    return myTowingPounds;
  }

  // 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 --------------------//


public class Driver{
  public static void main (String args[]){
    Vehicle standard = new Vehicle("Ford", 20000, 23, 1.95, 15000);
    System.out.println(standard);
    // When an 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);
    System.out.println();
    Truck rig = new Truck("Chevy", "4X4", 32000, 12, 1.95, 15000, 1500, 10000);
    System.out.println(rig);
  }
}
```

2. Write your own SportsCar class that extends the Car class. This new class should have the following unique private 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", 3.2, "firm", and "low profile") are the unique additions to the SportsCar class.

   ```
   SportsCar sporty = new SportsCar("Porsche", "Boxster", 55000, 19, 2.05,
   15000, 2, 2, "red", 3.2, "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.
   ```