**Example Inheritance Program**

This example involves three classes and demonstrates overriding and overloaded methods in a subclass.

The initial code for a Vehicle superclass, a Car subclass, and a TestCar2 test class is shown below.

/\*\*   
 Vehicle class  
   
 This class is used to create Vehicle objects.  
 \*/  
   
 public class Vehicle  
 {  
 // Vehicle class instance fields  
 private boolean moving; // whether or not the vehicle is currently moving  
 private double speed; // speed in mph of the vehicle  
 private char bearing; // direction vehicle headed ('N','E','S', or 'W')  
   
 public Vehicle() // Vehicle class no-arg constructor  
 {  
 moving = false; // assume not moving  
 speed = 0.0; // not moving  
 bearing = 'N'; // assume 'N'orth  
 System.out.println("Created a vehicle (no-arg)");   
 }  
   
 public Vehicle (double initialSpeed) // Vehicle 1-arg constructor  
 {  
 bearing = 'W';  
 speed = initialSpeed;  
 if (speed > 0.0)  
 {  
 moving = true;  
 }  
 System.out.println("Created a vehicle (1-arg)");   
 }   
   
 public Vehicle (double initialSpeed, char initialBearing) // Vehicle 2-arg constructor  
 {  
 bearing = initialBearing;  
 speed = initialSpeed;  
 if (speed > 0.0)  
 {  
 moving = true;  
 }  
 System.out.println("Created a vehicle (2-arg)");  
 }   
  
 public void start(double initialSpeed, char initialBearing)  
 {  
 moving = true;  
 if (initialSpeed >= 5.0 && initialSpeed <= 20.0)  
 {  
 speed = initialSpeed; // valid expected range  
 }  
 else if (initialSpeed >= 0.0 && initialSpeed < 5.0)  
 {  
 speed = 5.0; // minimum  
 }  
 else if (initialSpeed < 0.0)  
 {  
 speed = 0.0; // assume no movement  
 moving = false;  
 }  
 else if (initialSpeed > 20.0)  
 {  
 speed = 20.0; // maximum allowed  
 }  
 switch(initialBearing)  
 {  
 case 'N':  
 bearing = initialBearing;  
 break;  
 case 'E':  
 bearing = initialBearing;  
 break;  
 case 'S':  
 bearing = initialBearing;  
 break;  
 case 'W':  
 bearing = initialBearing;  
 default:  
 System.out.println("invalid bearing " + initialBearing +  
 " set to N"); // additional user notification  
 bearing = 'N';  
 }  
 }  
   
 public double getSpeed() // get and return current speed in mph  
 {  
 return speed;  
 }  
   
 public void setSpeed(double newSpeed) // set new speed in mph  
 {  
 speed = newSpeed;  
 }  
   
 public char getBearing() // get and return current bearing  
 {  
 return bearing;  
 }  
   
 public void speedUp(double mphSteps, int numSteps)  
 {  
 int counter = 0;  
 while (counter < numSteps)  
 {  
 speed += mphSteps;  
 System.out.println("counter= " + counter + ", " + this.toString());  
 counter++;  
 }  
 }  
   
 public String toString()  
 {  
 return "From toString(): speed= " + getSpeed() +  
 " mph and bearing= " + getBearing();  
 }   
}

/\*\*   
 Car class  
   
 This class is used to create Car objects and inherits from the Vehicle class.  
 \*/  
   
 public class Car extends Vehicle  
 {  
 // Car class instance fields  
 private String color; // color of the Car  
 private int doors; // number of doors of the Car  
 private double hp; // engine horsepower  
   
 public Car(String carColor, int numDoors,   
 double horsePower, double startingSpeed) // 4-arg constructor  
 {  
 super(startingSpeed);  
 color = carColor;  
 doors = numDoors;   
 hp = horsePower;   
 System.out.println("Created a car");   
 }  
   
 public String getColor()  
 {  
 return color;  
 }  
   
 public int getDoors()  
 {  
 return doors;  
 }  
   
 public double getHp()  
 {  
 return hp;  
 }  
   
 public String toString()  
 {  
 return "From Car toString(): color= " + getColor() +  
 " doors= " + getDoors() +   
 " hp= " + getHp() +  
 " speed= " + getSpeed() +  
 " mph and bearing= " + getBearing();  
 }   
}

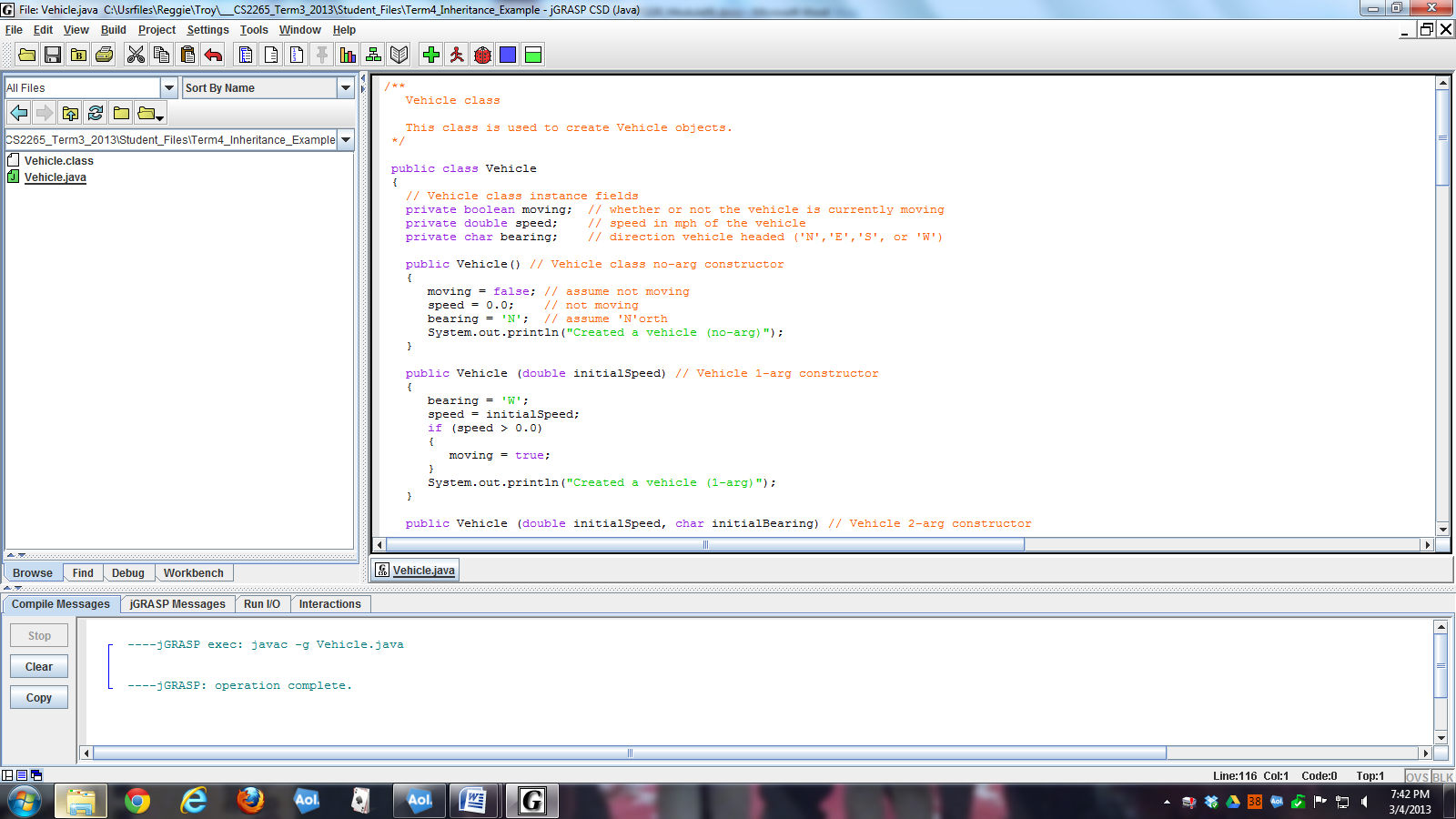
/\*\*  
 TestCar2 class  
   
 This class is used to create a Car, start it up, and then speed it up.  
 \*/  
   
public class TestCar2  
{  
 public static void main(String[] args)  
 {   
 // create, start, and speed up a Car  
 Car myCar2 = new Car("blue", 4, 300., 10.0); // blue car, 4 doors, 300. hp  
 // w/ initial speed of 10 mph  
 System.out.println(myCar2.toString());  
 myCar2.speedUp(5.0, 2);   
 }  
}

**The Assignment**

1. Start by compiling the code for all 3 classes.
2. Then modify the Car class below to provide a start() method that overrides the Vehicle class start() method and a 1-parameter speedUp() method that overloads the Vehicle class’s 2-parameter speedup() method.
3. Do not modify the Vehicle class code!
4. In a new test class called TestCar3 have the Car class’s overriding start() method start a car in a southerly direction at zero speed. Then have the Car class’s overloaded speedup() method accept a value of 50 as the additional miles per hour that the car should start moving from its current speed.

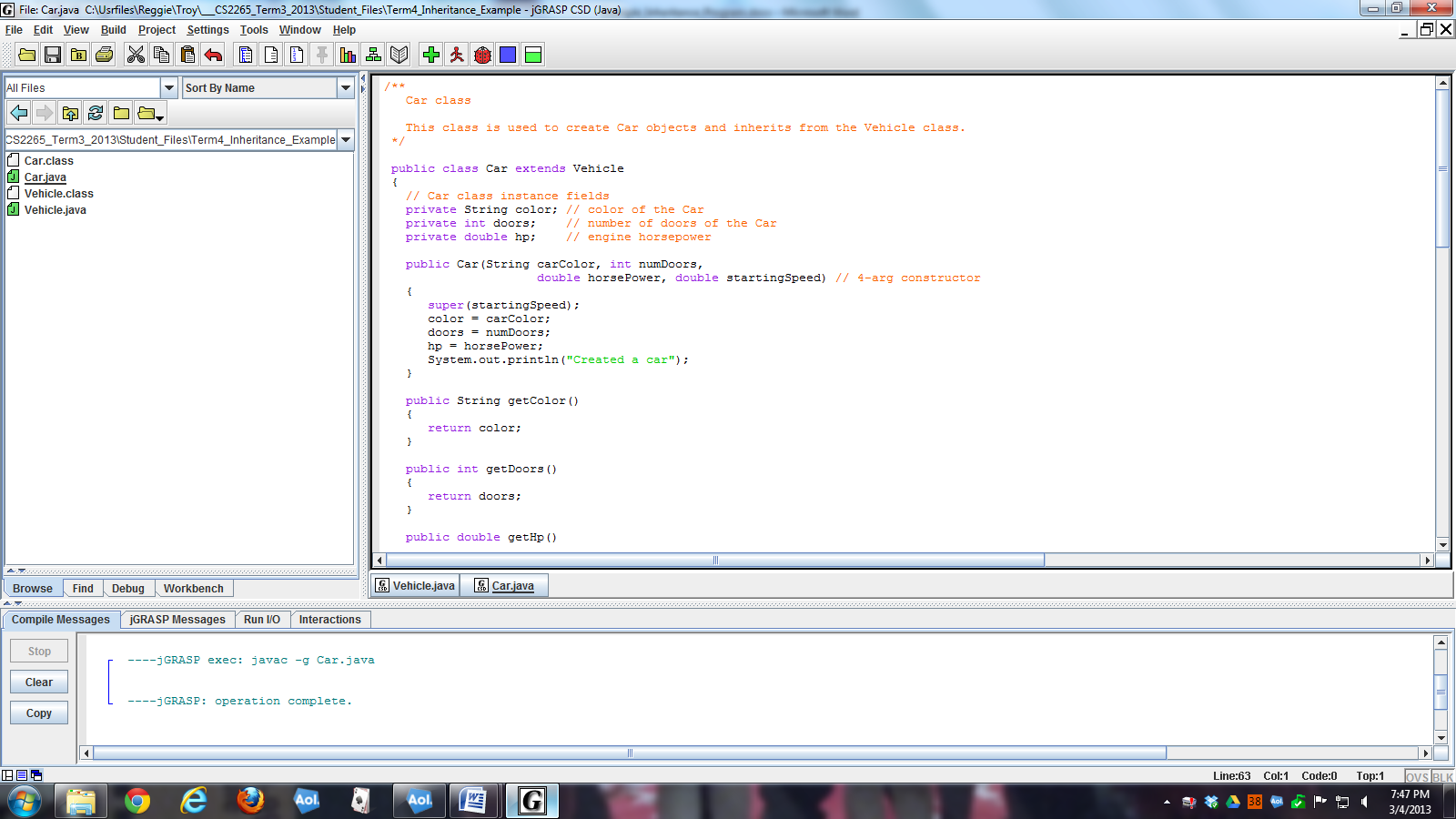
**The Solution**

1. The unmodified Vehicle class is first compiled.

****

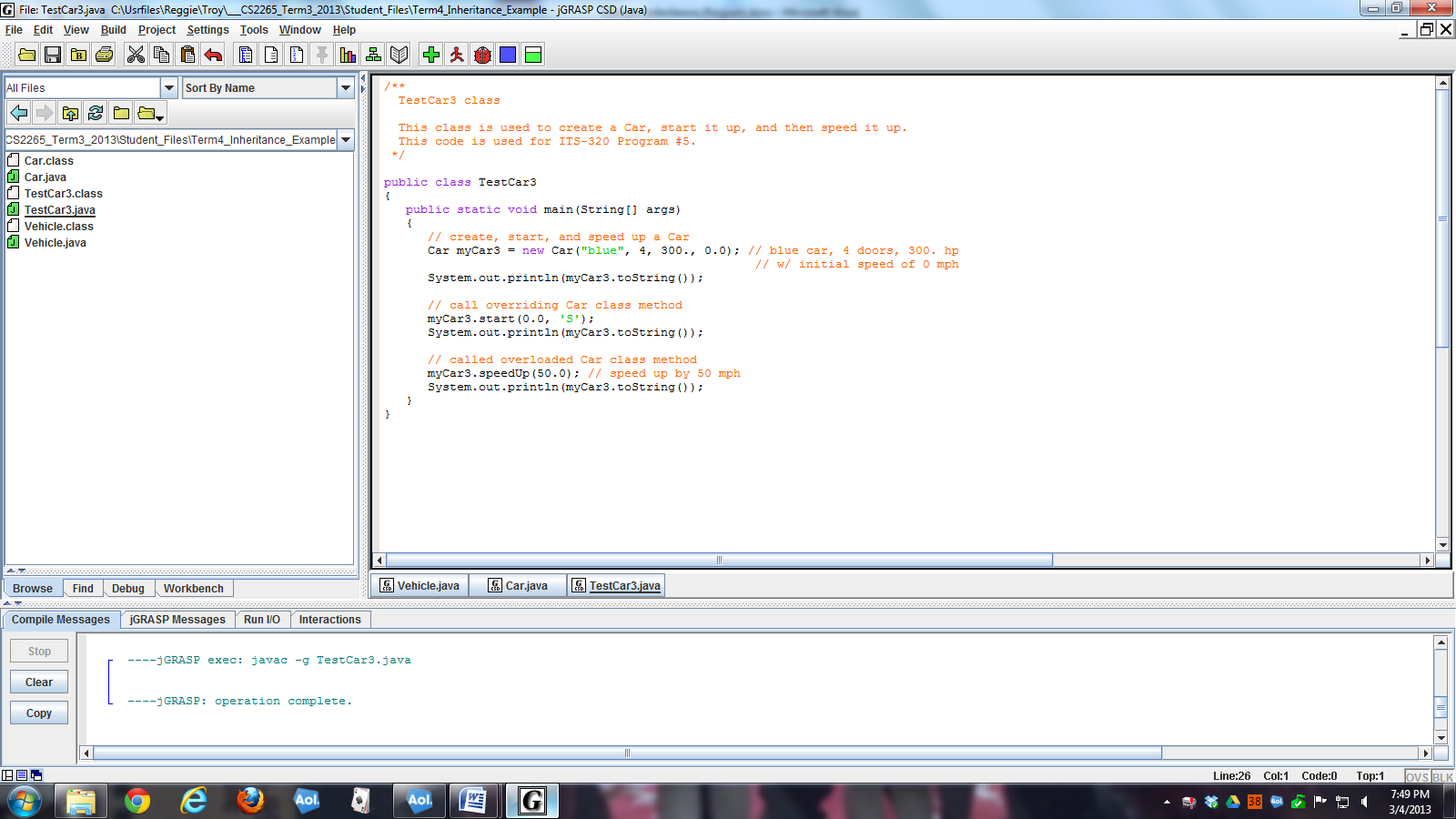
1. The Car class is then modified and compiled.

/\*\*   
 Car class  
   
 This class is used to create Car objects and inherits from the Vehicle class.  
 \*/  
   
 public class Car extends Vehicle  
 {  
 // Car class instance fields  
 private String color; // color of the Car  
 private int doors; // number of doors of the Car  
 private double hp; // engine horsepower  
   
 public Car(String carColor, int numDoors,   
 double horsePower, double startingSpeed) // 4-arg constructor  
 {  
 super(startingSpeed);  
 color = carColor;  
 doors = numDoors;   
 hp = horsePower;   
 System.out.println("Created a car");   
 }  
   
 public String getColor()  
 {  
 return color;  
 }  
   
 public int getDoors()  
 {  
 return doors;  
 }  
   
 public double getHp()  
 {  
 return hp;  
 }  
   
 // overrides the Vehicle start() method  
 public void start(double initialSpeed, char initialBearing)  
 {  
 System.out.println("in overriding Car class start() method");  
 super.start(0.0, 'S'); // will start at 5 mph minimum  
 super.setSpeed(0.0); // this puts it to zero mph  
 }  
   
 // overloads the Vehicle speedUp() method  
 public void speedUp(double extraSpeed)  
 {  
 System.out.println("in overloaded Car class speedUp() method");  
 this.setSpeed( this.getSpeed() + extraSpeed );  
 }  
   
 public String toString()  
 {  
 return "From Car toString(): color= " + getColor() +  
 " doors= " + getDoors() +   
 " hp= " + getHp() +  
 " speed= " + getSpeed() +  
 " mph and bearing= " + getBearing();  
 }   
}

****

1. A new test class is then created to demonstrate the new overridden and overloaded Car class methods.

/\*\*  
 TestCar3 class  
   
 This class is used to create a Car, start it up, and then speed it up.  
 This code is used for ITS-320 Program #5.  
 \*/  
   
public class TestCar3  
{  
 public static void main(String[] args)  
 {   
 // create, start, and speed up a Car  
 Car myCar3 = new Car("blue", 4, 300., 0.0); // blue car, 4 doors, 300. hp  
 // w/ initial speed of 0 mph  
 System.out.println(myCar3.toString());  
   
 // call overriding Car class method   
 myCar3.start(0.0, 'S');   
 System.out.println(myCar3.toString());  
   
 // called overloaded Car class method  
 myCar3.speedUp(50.0); // speed up by 50 mph  
 System.out.println(myCar3.toString());   
 }  
}



1. The TestCar3 class is then executed.

ÏÏ«Ï ----jGRASP exec: java TestCar3  
ÏÏ§Ï  
ÏÏ§ÏCreated a vehicle (1-arg)  
ÏÏ§ÏCreated a car  
ÏÏ§ÏFrom Car toString(): color= blue doors= 4 hp= 300.0 speed= 0.0 mph and bearing= W  
ÏÏ§Ïin overriding Car class start() method  
ÏÏ§ÏFrom Car toString(): color= blue doors= 4 hp= 300.0 speed= 0.0 mph and bearing= S  
ÏÏ§Ïin overloaded Car class speedUp() method  
ÏÏ§ÏFrom Car toString(): color= blue doors= 4 hp= 300.0 speed= 50.0 mph and bearing= S  
ÏÏ§Ï  
ÏÏ©Ï ----jGRASP: operation complete.