Java Polymorphism

Multiple Forms

web explorations

Preview

You are getting deeper into the project helping the city maintain their vehicles. And they have a lot of different kinds.
Autos, police cars, buses, trucks, snow plows.

Each of these vehicles has common features such as a horn and an engine and a driver.
And the different types of vehicles have unique features and capabilities.

You've decided to use inheritance create a general Vehicle class as well as child classes for each different type of vehicle.

With inheritance you have the power of polymorphism allowing you to keep track of the type of vehicle based on the subclass.

In this tutorial we will be tracking which type of vehicle is parked in which parking area of the city garage.

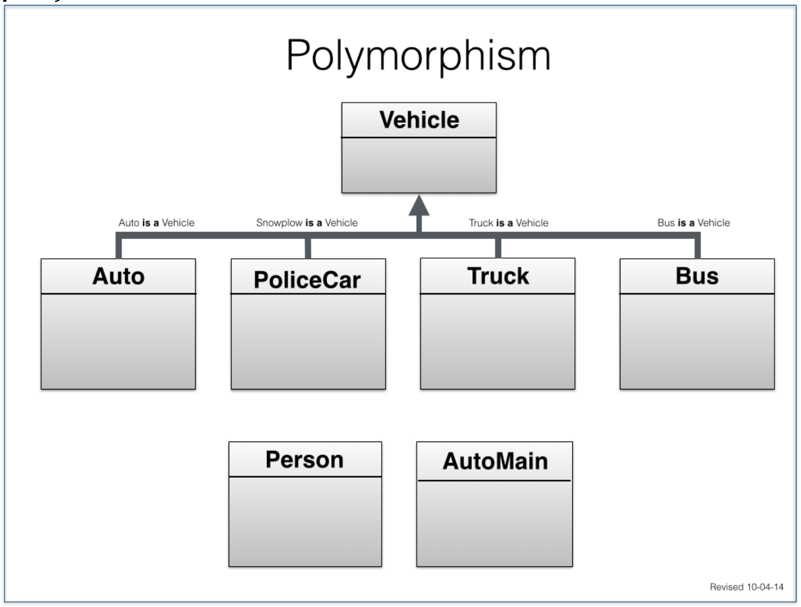
You will be storing the information in an ArrayList and then walking through the array using sound files and text to designate which type of vehicle is parked in each parking area.

Here are some of the skills you will be practicing:

- Using sound files in Java with the InputStream, AudioStream, and AudioPlayer classes.
- Writing a generic method in the parent class.
- Overriding methods in the child classes.
- Using the concept of polymorphism to track different types of objects.

UML of the Project

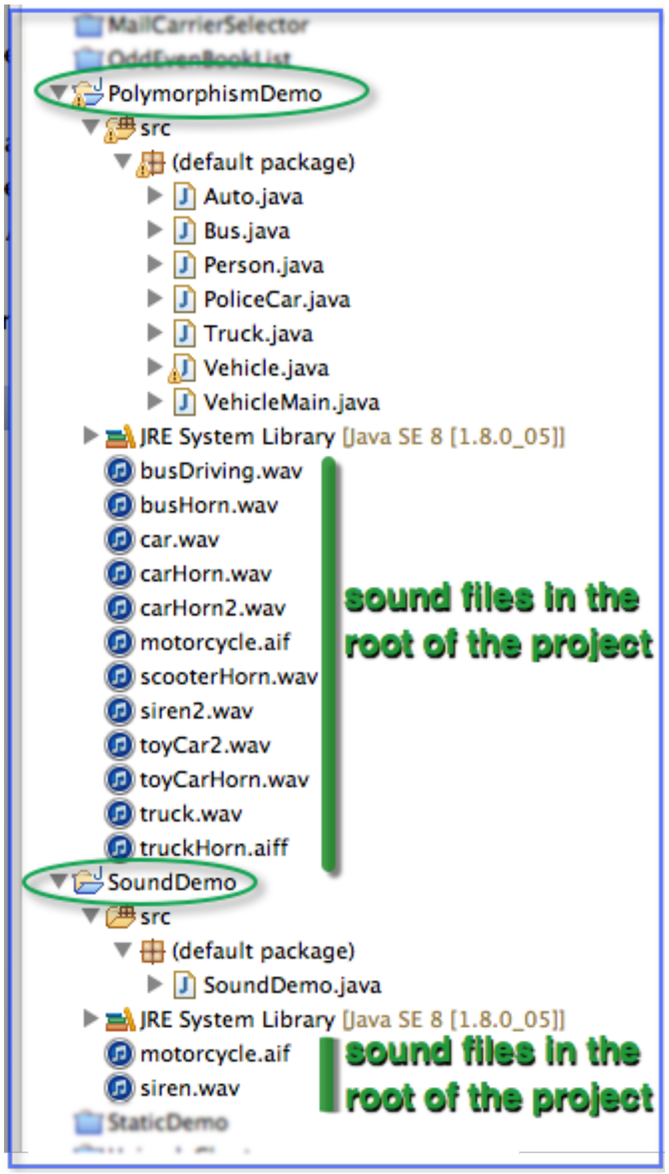
Here is a UML diagram showing how the classes will be organized for this project.



The parent class, Vehicle, will have a playSound( ) method as well as a generic drive( ) and playSound( ) method.
(The parent will have toy car sounds just to keep us honest.)

Each child class will have a unique sound for its drive( ) and playSound( ) method.

Here is what your Package Explorer will look like in Eclipse:


MailCarrierSelector
OddEvenBookList
PolymorphismDemo
src
(default package)
Auto.java
Bus.java
Person.java
PoliceCar.java
Truck.java
Vehicle.java
VehicleMain.java
JRE System Library [Java SE 8 [1.8.0_05]]
busDriving.wav
busHorn.wav
car.wav
carHorn.wav
carHorn2.wav
motorcycle.aif
scooterHorn.wav
siren2.wav
toyCar2.wav
toyCarHorn.wav
truck.wav
truckHorn.aiff
sound files in the root of the project
SoundDemo
src
(default package)
SoundDemo.java
JRE System Library [Java SE 8 [1.8.0_05]]
motorcycle.aif
siren.wav
sound files in the root of the project
StaticDemo

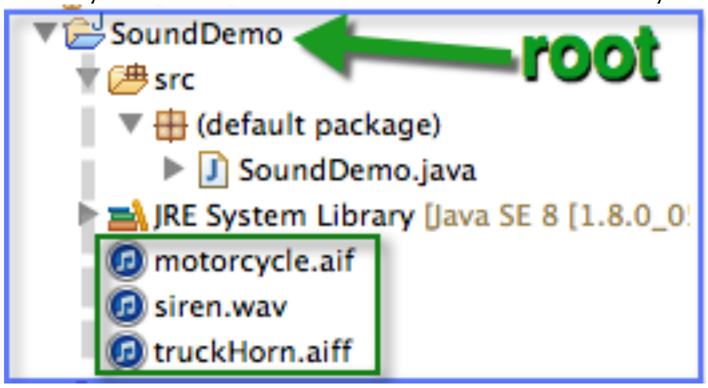
Playing Sound Files

For this application we want to play sound files designating the different types of vehicles. Can we do that with Java?

A quick Web search (java play sound files) shows us that there are lots of different ways to do it.
A simple one that doesn't require any outside libraries (such as JavaFX or commercial packages) uses InputStream, AudioStream, and AudioPlayer classes.

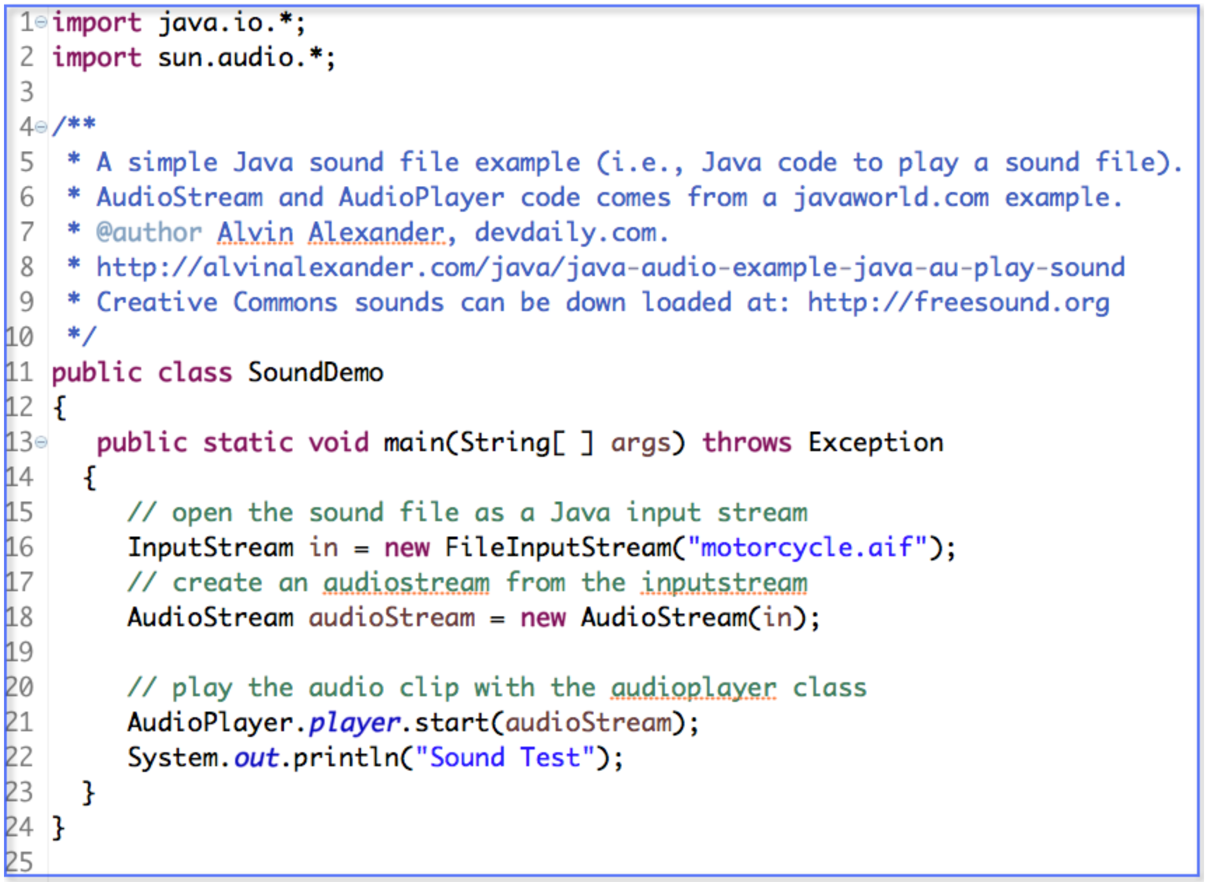
You can also download sound effect files from http://www.freesound.org.
These files are licensed under various Creative Commons making them free and available for you to use as long as you give attribution (credit) to the recording artist.
Java supports .aiff, .aif, .au, and .wav files. (.wav files are uncompressed and are usually much bigger than desired for apps being used on a mobile network.)

Be smart and set up a separate project to demonstrate using sound files. You might want to call this project and class: **SoundDemo**
Save your sound files in the root directory of the project.



```
SoundDemo
  src
    (default package)
      SoundDemo.java
  JRE System Library [Java SE 8 [1.8.0_0
  motorcycle.aif
  siren.wav
  truckHorn.aiff
```

Type in this code:



```java
import java.io.*;
import sun.audio.*;

/**
 * A simple Java sound file example (i.e., Java code to play a sound file).
 * AudioStream and AudioPlayer code comes from a javaworld.com example.
 * @author Alvin Alexander, devdaily.com.
 * http://alvinalexander.com/java/java-audio-example-java-au-play-sound
 * Creative Commons sounds can be down loaded at: http://freesound.org
 */
public class SoundDemo
{
   public static void main(String[ ] args) throws Exception
  {
     // open the sound file as a Java input stream
     InputStream in = new FileInputStream("motorcycle.aif");
     // create an audiostream from the inputstream
     AudioStream audioStream = new AudioStream(in);

     // play the audio clip with the audioplayer class
     AudioPlayer.player.start(audioStream);
     System.out.println("Sound Test");
  }
}
```

When you run this program you should hear the sound clip play.
Notice how smart we were to also include a System.out.println( ) statement so we know the program has run!

Vehicle

Now that you have sound system up and running lets set up the parent class with the methods we need.

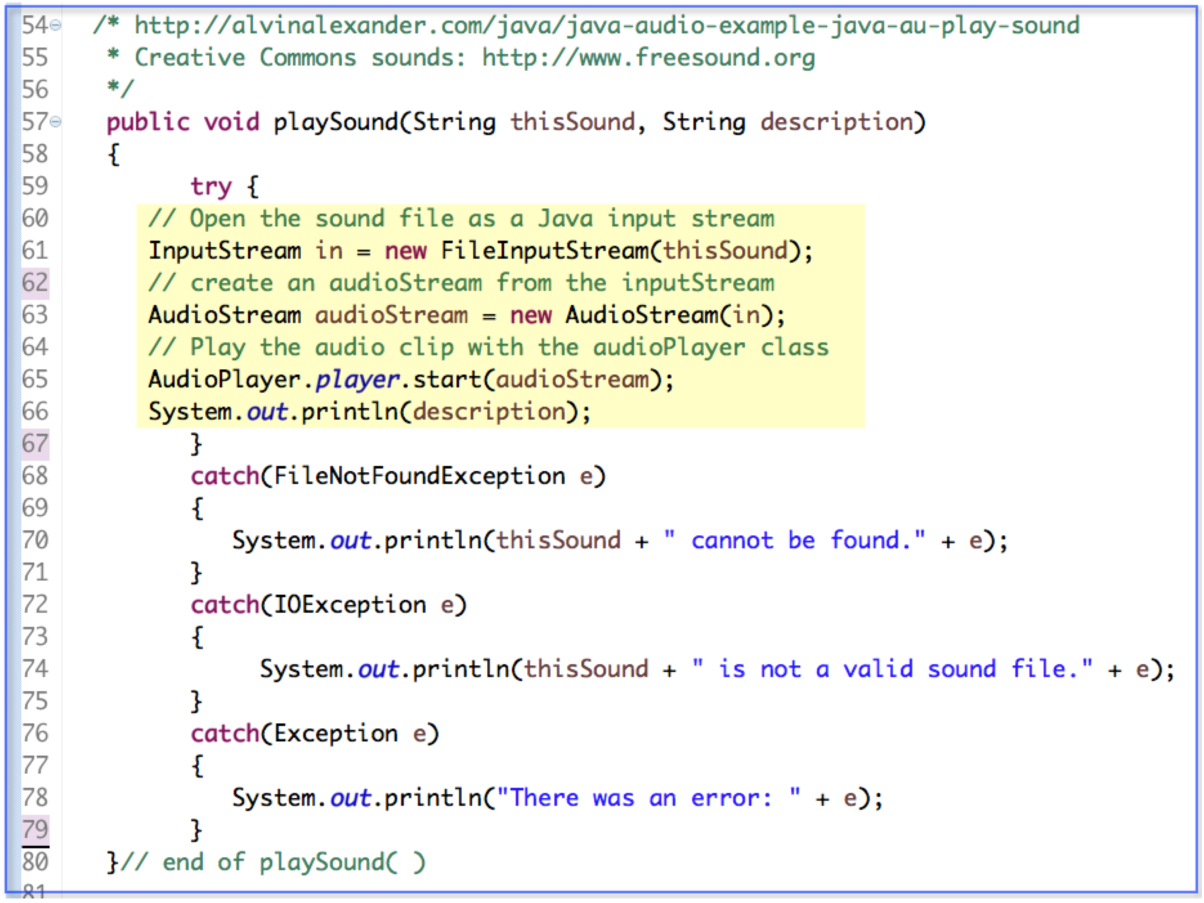
Open up a new project. You might want to call it: **PolymorphismDemo**

In your project create a new class called Vehicle.
Use the Vehicle code from earlier labs to save typing time. Just copy and paste the existing code into your new class.
You can access the original Vehicle class code here: http://webexplorations.com/book/java/InheritanceDemo.zip

What about the Person class that Vehicle requires?
There is also a Person class in the InheritanceDemo.zip file.
Or you can comment out the lines in Vehicle that declare a Person object.
The Person object is not essential to this tutorial.

We are going to add three new methods to this class, playSound( ), soundHorn( ), and drive( );
First the playSound( ) method which the other two will use.

Add this method to Vehicle (inside your PolymorphismDemo). The best location would be right after the toPrint( ) method and before the setters and getters.



```
/* http://alvinalexander.com/java/java-audio-example-java-au-play-sound
 * Creative Commons sounds: http://www.freesound.org
 */
public void playSound(String thisSound, String description)
{
        try {
    // Open the sound file as a Java input stream
    InputStream in = new FileInputStream(thisSound);
    // create an audioStream from the inputStream
    AudioStream audioStream = new AudioStream(in);
    // Play the audio clip with the audioPlayer class
    AudioPlayer.player.start(audioStream);
    System.out.println(description);
        }
        catch(FileNotFoundException e)
        {
           System.out.println(thisSound + " cannot be found." + e);
        }
        catch(IOException e)
        {
             System.out.println(thisSound + " is not a valid sound file." + e);
        }
        catch(Exception e)
        {
           System.out.println("There was an error: " + e);
        }
}// end of playSound( )
```

The method is set up using the test code from the SoundDemo project. We added two parameters so we can specify different sounds files and display different messages depending on the sound file that is playing.

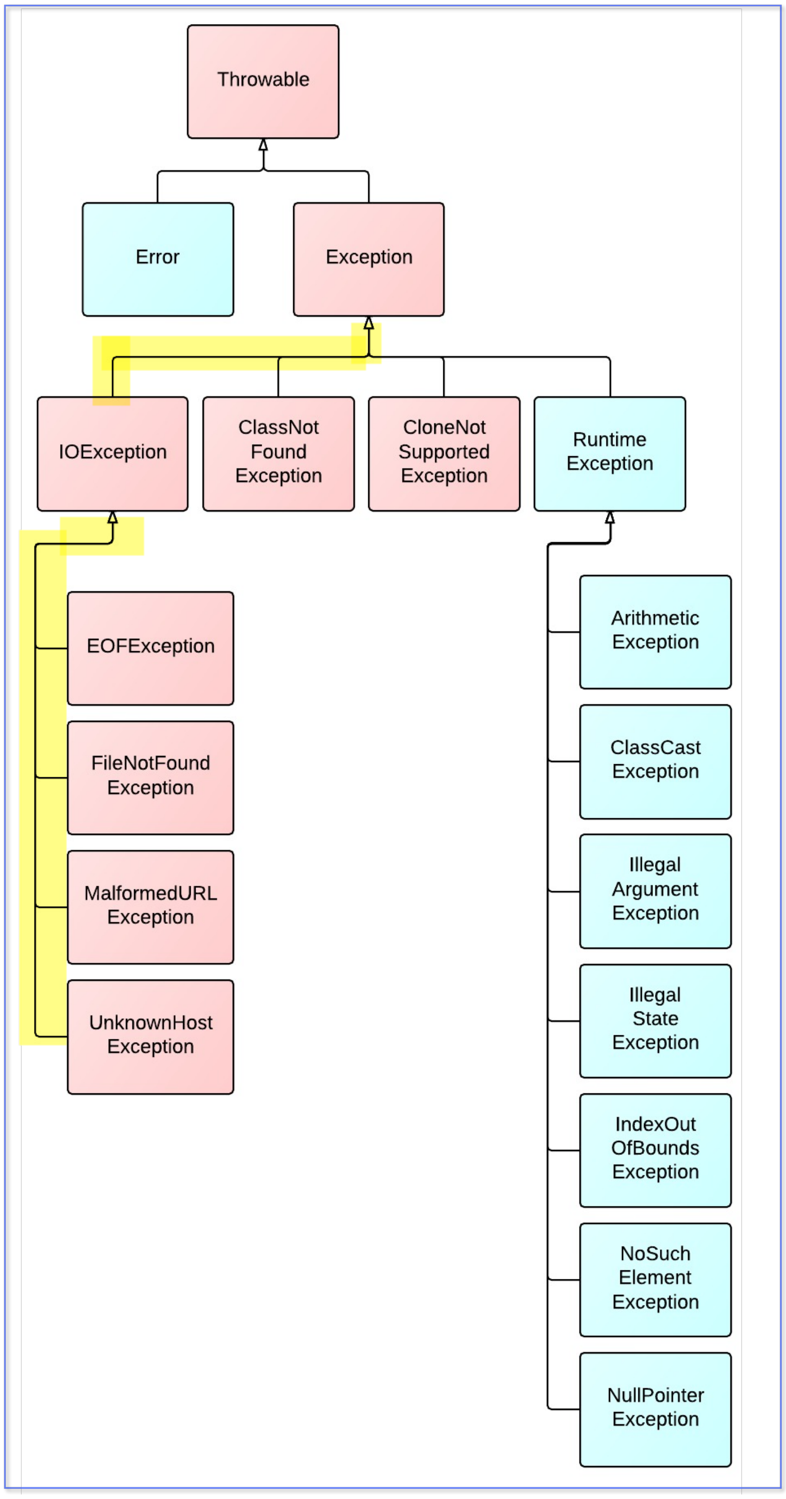
The main sound generating code has been highlighted in yellow. Notice how much more code is added in order to handle the possible exceptions.

We could have simply used the catch(Exception e) { } block but that is too general. (Remember your own frustration on obscurely worded error messages?)
Instead we check for various types of errors and give the appropriate message in English as well as the information created by Java by printing out the exception object named "e".

The order of exception handling
Errors must be handled in the appropriate order, moving from specific to general.
FileNotFoundException is more specific than an IOException. And both of

these are more specific than the most general Exception class.
If you move the IOException catch block before the FileNotFoundException block Eclipse will give you an error message: "FileNotFound handled by the IOException.

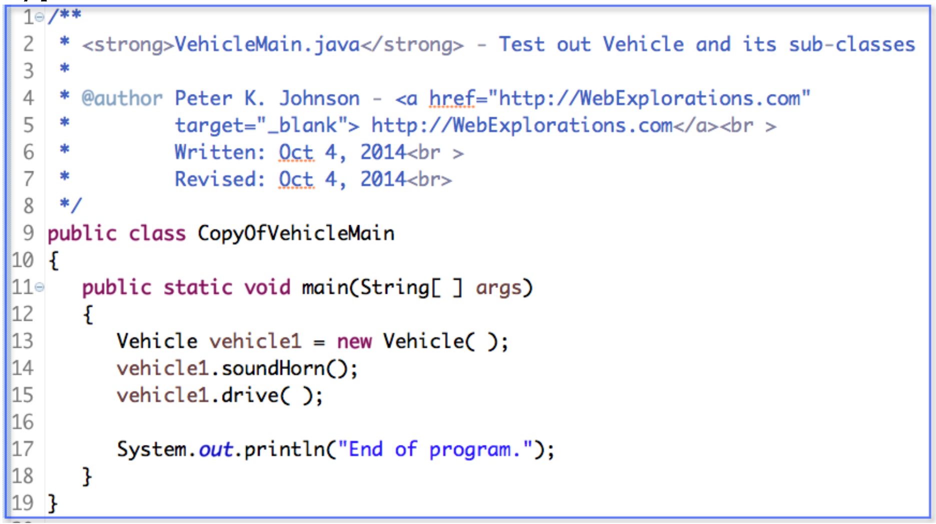
Here is a UML diagram showing the Exception handling hierarchy.
The file exceptions have been highlighted in yellow.


Throwable
Error
Exception
IOException
ClassNot Found Exception
CloneNot Supported Exception
Runtime Exception
EOFException
FileNotFound Exception
MalformedURL Exception
UnknownHost Exception
Arithmetic Exception
ClassCast Exception
Illegal Argument Exception
Illegal State Exception
IndexOut OfBounds Exception
NoSuch Element Exception
NullPointer Exception

Testing with VehicleMain

Create a new class named VehicleMain to test the Vehicle base class. (Parent, Base, Super all refer to the same thing. They are synonymous.)

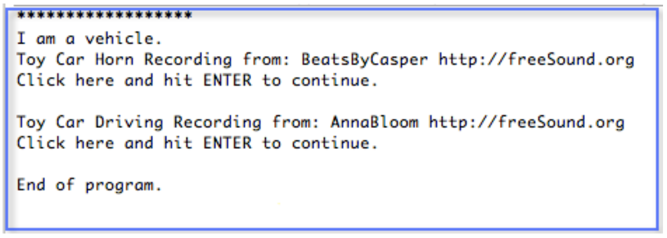
Type in this code:



```
/**
 * <strong>VehicleMain.java</strong> - Test out Vehicle and its sub-classes
 *
 * @author Peter K. Johnson - <a href="http://WebExplorations.com"
 *         target="_blank"> http://WebExplorations.com</a><br >
 *         Written: Oct 4, 2014<br >
 *         Revised: Oct 4, 2014<br>
 */
public class CopyOfVehicleMain
{
   public static void main(String[ ] args)
   {
      Vehicle vehicle1 = new Vehicle( );
      vehicle1.soundHorn();
      vehicle1.drive( );

      System.out.println("End of program.");
   }
}
```

We create a new Vehicle type object named vehicle1.
Then we use it to sound the horn and drive.

Even if you can't hear the sound files you'll be able to see the output in the console:



```
******************
I am a vehicle.
Toy Car Horn Recording from: BeatsByCasper http://freeSound.org
Click here and hit ENTER to continue.

Toy Car Driving Recording from: AnnaBloom http://freeSound.org
Click here and hit ENTER to continue.

End of program.
```

Notice how useful that simple
`System.out.println("End of program.");`
turns out to be.

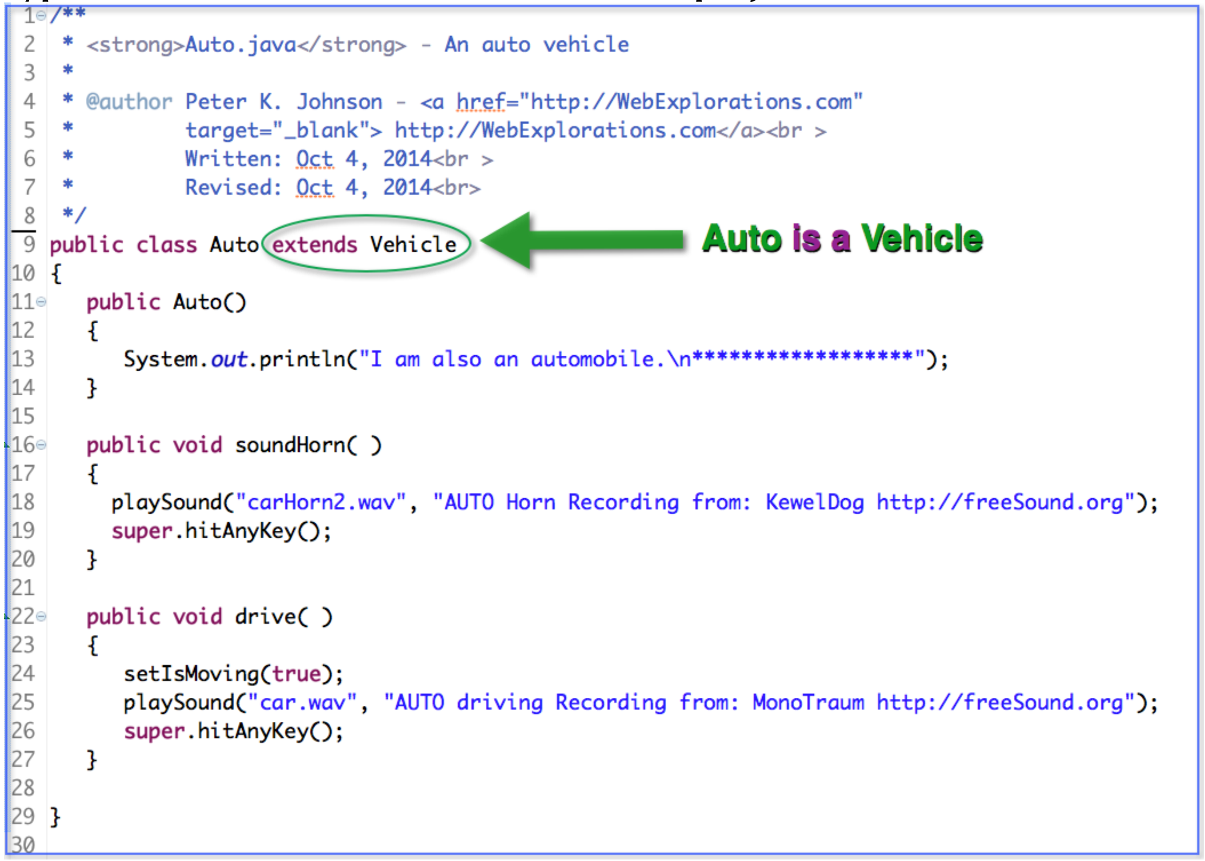
So far so good. Now we just have to focus on the children...

Auto

In order to use polymorphism you need two things in place:
(1) You need to have at least two child classes.
(2) Those child classes have to have methods that override what is in the parent class.

We are going to create an Auto class as a child of Vehicle using the keyword "extends".
Inside our child class let's override the soundHorn( ) and drive( ) methods, giving our Auto a more specific "auto" horn and motor sound.

Type in this code in a new class inside the project:

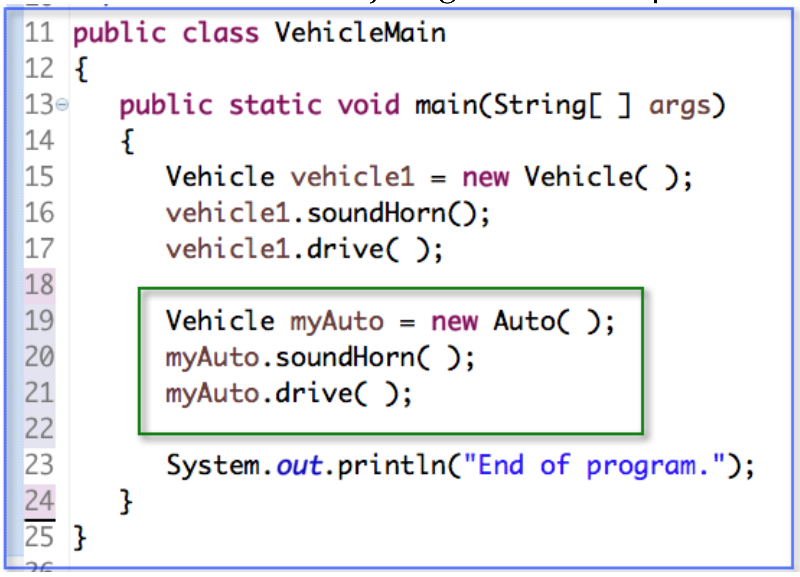


```
 1 /**
 2  * <strong>Auto.java</strong> - An auto vehicle
 3  *
 4  * @author Peter K. Johnson - <a href="http://WebExplorations.com"
 5  *         target="_blank"> http://WebExplorations.com</a><br >
 6  *         Written: Oct 4, 2014<br >
 7  *         Revised: Oct 4, 2014<br>
 8  */
 9 public class Auto extends Vehicle
10 {
11    public Auto()
12    {
13       System.out.println("I am also an automobile.\n*****************");
14    }
15 
16    public void soundHorn( )
17    {
18      playSound("carHorn2.wav", "AUTO Horn Recording from: KewelDog http://freeSound.org");
19      super.hitAnyKey();
20    }
21 
22    public void drive( )
23    {
24       setIsMoving(true);
25       playSound("car.wav", "AUTO driving Recording from: MonoTraum http://freeSound.org");
26       super.hitAnyKey();
27    }
28 
29 }
30 
```

Notice that the method playSound( ) is part of Vehicle so it is available to this class.
Alternatively, you could use the keyword "super" to access methods (and properties) in the parent class like we did on line #26 with super.hitAnyKey( );

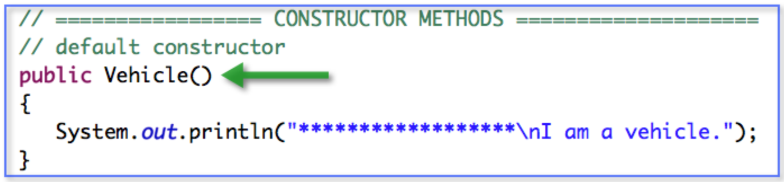
To test out the new class, add the following code to VehicleMain.

Notice the key difference: The type of myAuto is Vehicle, but the constructor used is Auto( );
This makes the new object general and specific at the same time.



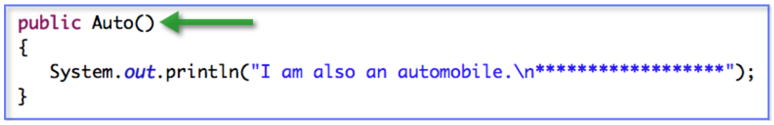
```
public class VehicleMain
{
   public static void main(String[ ] args)
   {
      Vehicle vehicle1 = new Vehicle( );
      vehicle1.soundHorn();
      vehicle1.drive( );

      Vehicle myAuto = new Auto( );
      myAuto.soundHorn( );
      myAuto.drive( );

      System.out.println("End of program.");
   }
}
```

When you run the VehicleMain you should see something like this (as well as hear the appropriate sounds).
When the new object is made the Vehicle constructor runs first: "I am a vehicle."



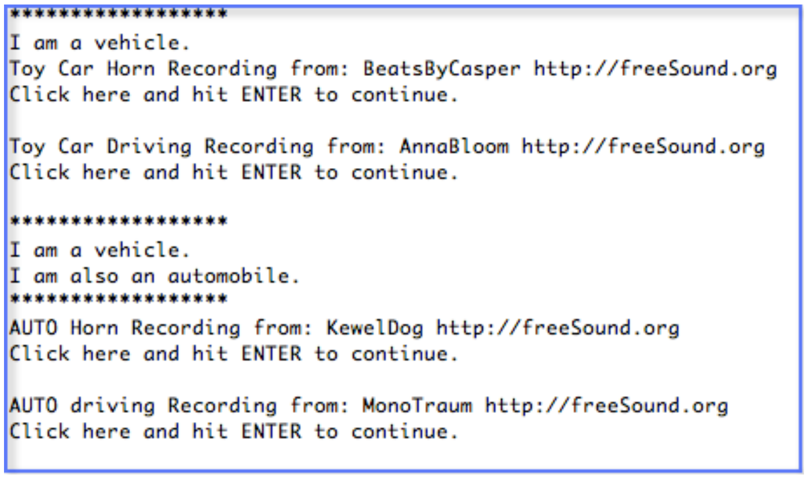
```
// ================ CONSTRUCTOR METHODS ===================
// default constructor
public Vehicle()
{
   System.out.println("*****************\nI am a vehicle.");
}
```

And then the child constructor runs: "I am also an automobile."



```
public Auto()
{
   System.out.println("I am also an automobile.\n*****************");
}
```

In Java-speak: Objects made from Auto "is a" Vehicle.

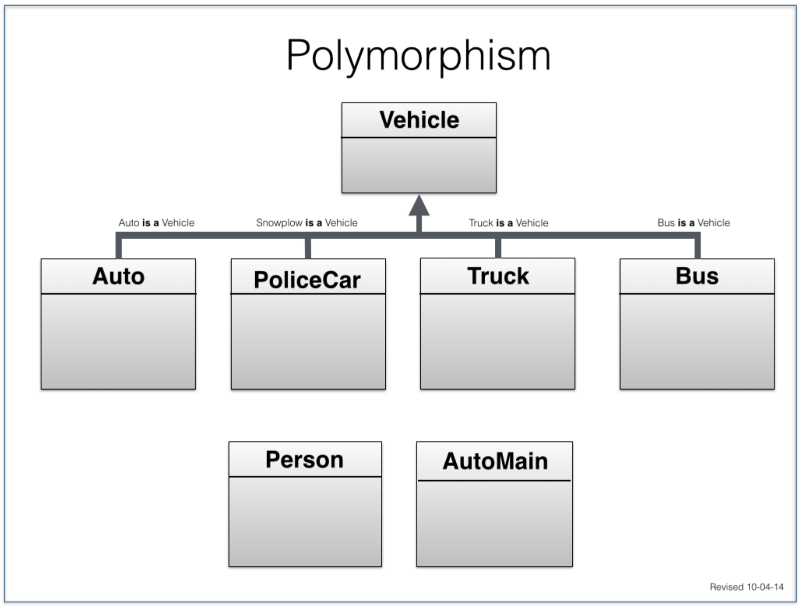


```
******************
I am a vehicle.
Toy Car Horn Recording from: BeatsByCasper http://freeSound.org
Click here and hit ENTER to continue.

Toy Car Driving Recording from: AnnaBloom http://freeSound.org
Click here and hit ENTER to continue.

******************
I am a vehicle.
I am also an automobile.
******************
AUTO Horn Recording from: KewelDog http://freeSound.org
Click here and hit ENTER to continue.

AUTO driving Recording from: MonoTraum http://freeSound.org
Click here and hit ENTER to continue.
```

This is the key to polymorphism. But, in order to have "many forms" we need some other child classes.

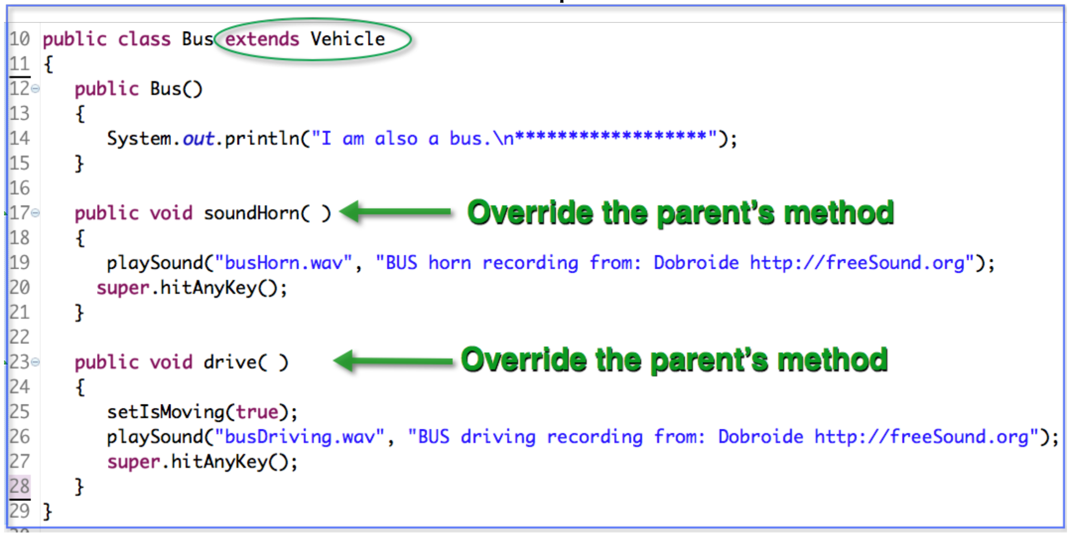
Truck, Bus, PoliceCar

Looking at the UML diagram we can see that there will be several sub classes: PoliceCar, Truck, and Bus.

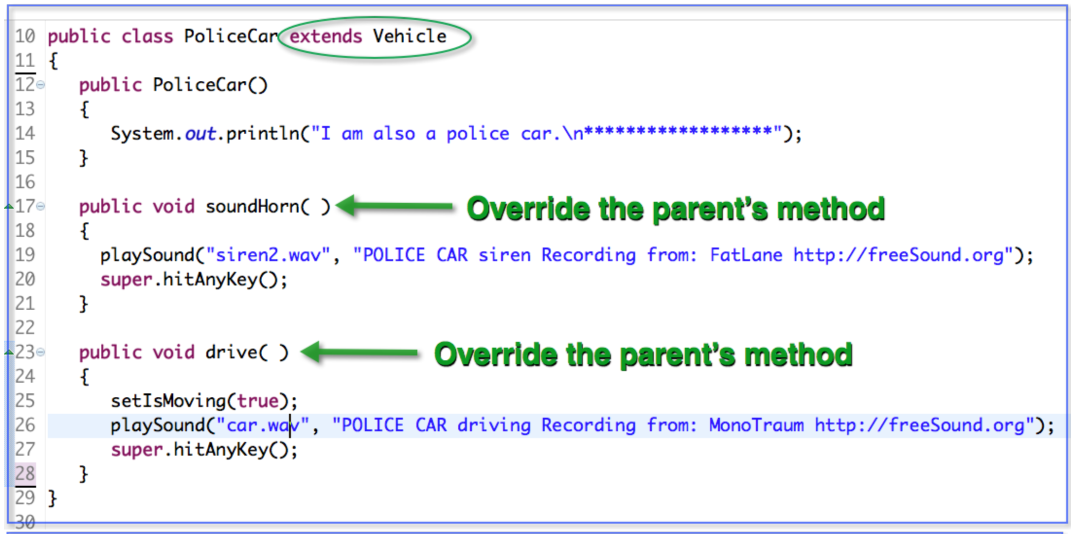


Make each of these a sub class of Vehicle using the keyword "extends". Include an overriding method for soundHorn( ) and drive( ).

Type in this code as separate classes for each subclass:
You can copy and paste the Auto class file in Eclipse. The only difference is the sound file name and the description.

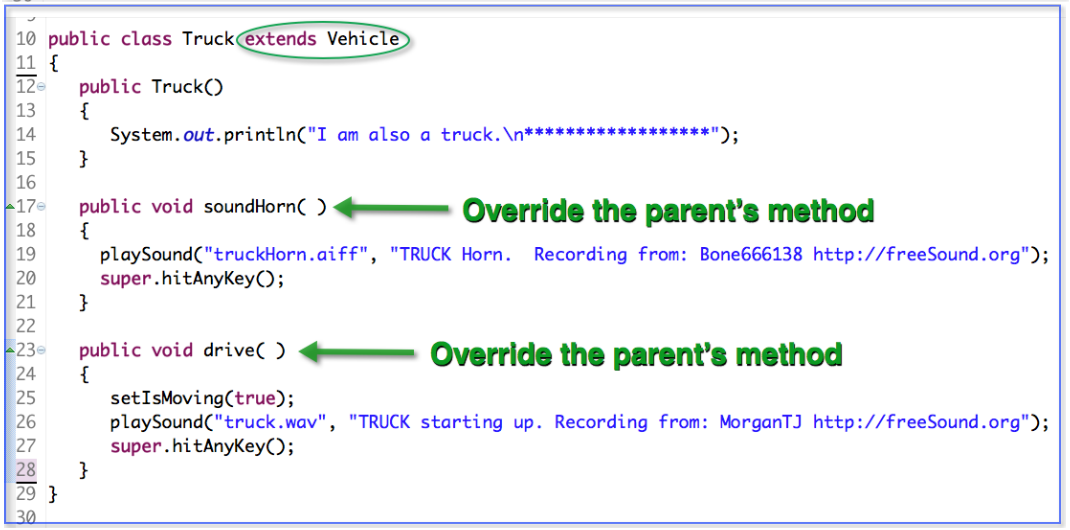


```java
public class Bus extends Vehicle
{
    public Bus()
    {
        System.out.println("I am also a bus.\n*****************");
    }

    public void soundHorn( )
    {
        playSound("busHorn.wav", "BUS horn recording from: Dobroide http://freeSound.org");
      super.hitAnyKey();
    }

    public void drive( )
    {
        setIsMoving(true);
        playSound("busDriving.wav", "BUS driving recording from: Dobroide http://freeSound.org");
        super.hitAnyKey();
    }
}
```



```java
public class PoliceCar extends Vehicle
{
   public PoliceCar()
   {
      System.out.println("I am also a police car.\n******************");
   }

   public void soundHorn( )
   {
     playSound("siren2.wav", "POLICE CAR siren Recording from: FatLane http://freeSound.org");
     super.hitAnyKey();
   }

   public void drive( )
   {
      setIsMoving(true);
      playSound("car.wav", "POLICE CAR driving Recording from: MonoTraum http://freeSound.org");
      super.hitAnyKey();
   }
}
```

Override the parent's method
Override the parent's method



```java
public class Truck extends Vehicle
{
   public Truck()
   {
      System.out.println("I am also a truck.\n******************");
   }

   public void soundHorn( )
   {
     playSound("truckHorn.aiff", "TRUCK Horn.  Recording from: Bone666138 http://freeSound.org");
     super.hitAnyKey();
   }

   public void drive( )
   {
      setIsMoving(true);
      playSound("truck.wav", "TRUCK starting up. Recording from: MorganTJ http://freeSound.org");
      super.hitAnyKey();
   }
}
```

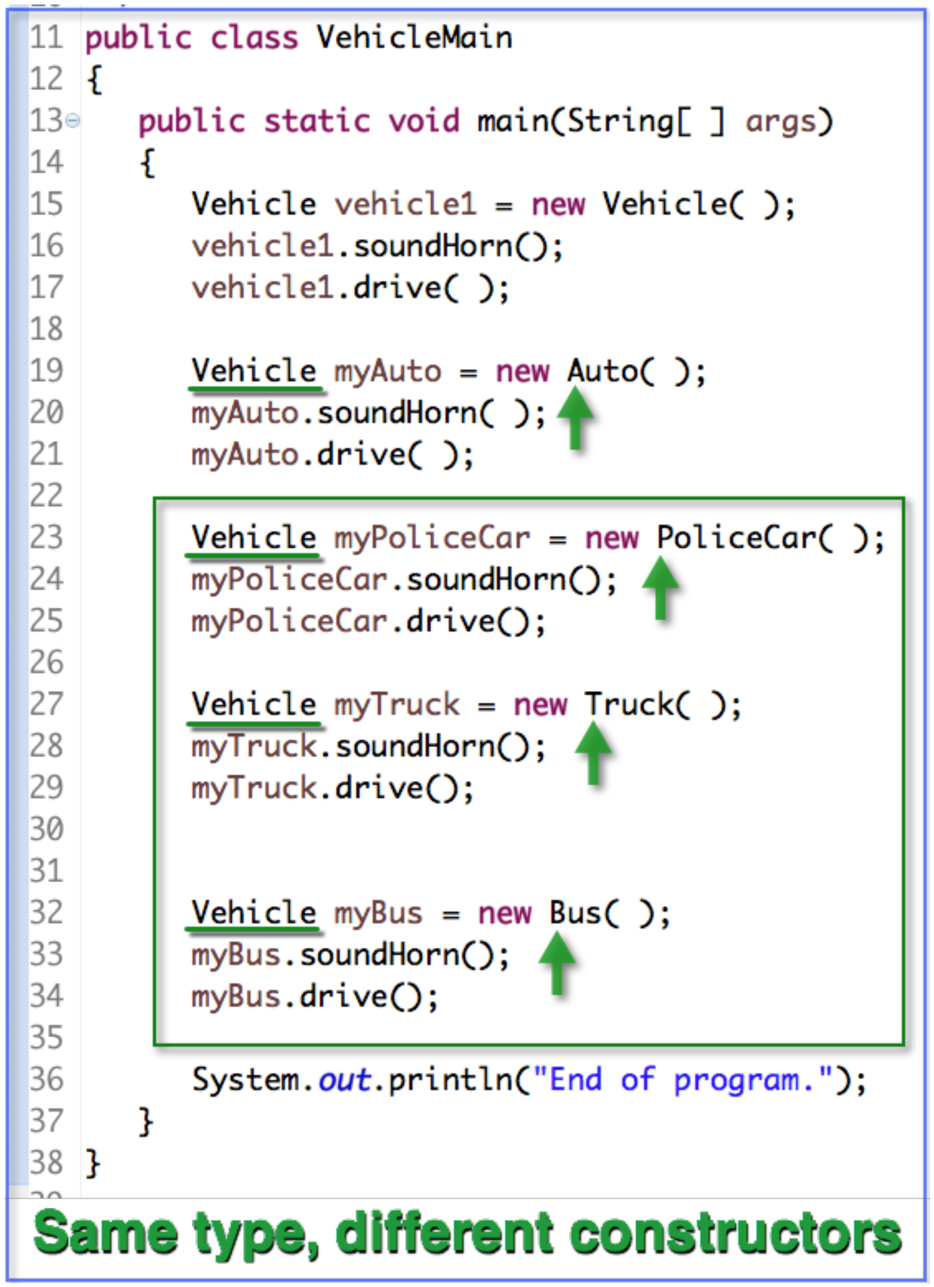
Override the parent's method
Override the parent's method

Testing with VehicleMain

Okay, let's test out our new derived or child classes.

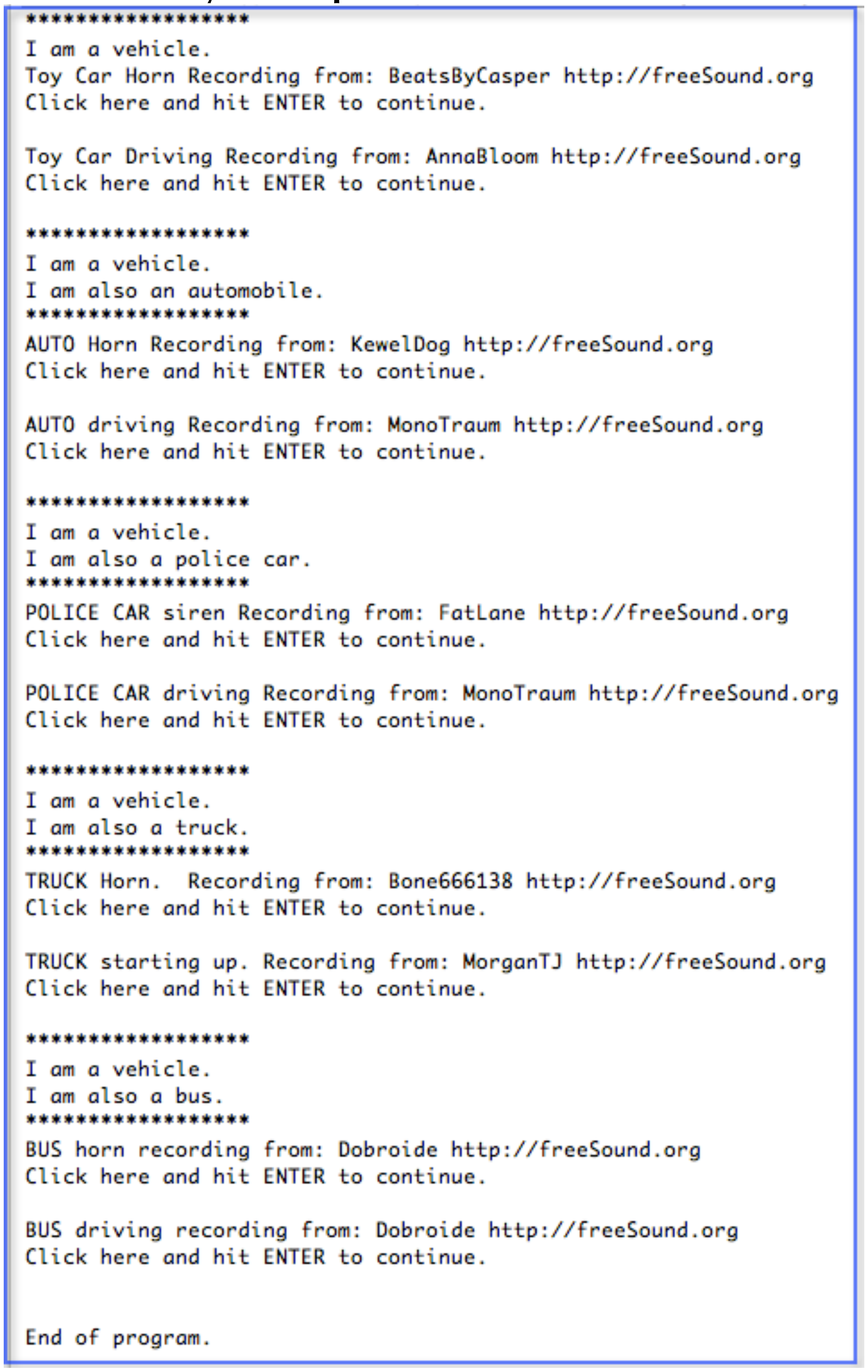
Create some new objects in VehicleMain:

They will all be the same type (Vehicle) but each one will use its own constructor:



```
public class VehicleMain
{
    public static void main(String[ ] args)
    {
        Vehicle vehicle1 = new Vehicle( );
        vehicle1.soundHorn();
        vehicle1.drive( );

        Vehicle myAuto = new Auto( );
        myAuto.soundHorn( );
        myAuto.drive( );

        Vehicle myPoliceCar = new PoliceCar( );
        myPoliceCar.soundHorn();
        myPoliceCar.drive();

        Vehicle myTruck = new Truck( );
        myTruck.soundHorn();
        myTruck.drive();


        Vehicle myBus = new Bus( );
        myBus.soundHorn();
        myBus.drive();

        System.out.println("End of program.");
    }
}
```

Here is what your output should look like:



```
******************
I am a vehicle.
Toy Car Horn Recording from: BeatsByCasper http://freeSound.org
Click here and hit ENTER to continue.

Toy Car Driving Recording from: AnnaBloom http://freeSound.org
Click here and hit ENTER to continue.

******************
I am a vehicle.
I am also an automobile.
******************
AUTO Horn Recording from: KewelDog http://freeSound.org
Click here and hit ENTER to continue.

AUTO driving Recording from: MonoTraum http://freeSound.org
Click here and hit ENTER to continue.

******************
I am a vehicle.
I am also a police car.
******************
POLICE CAR siren Recording from: FatLane http://freeSound.org
Click here and hit ENTER to continue.

POLICE CAR driving Recording from: MonoTraum http://freeSound.org
Click here and hit ENTER to continue.

******************
I am a vehicle.
I am also a truck.
******************
TRUCK Horn.  Recording from: Bone666138 http://freeSound.org
Click here and hit ENTER to continue.

TRUCK starting up. Recording from: MorganTJ http://freeSound.org
Click here and hit ENTER to continue.

******************
I am a vehicle.
I am also a bus.
******************
BUS horn recording from: Dobroide http://freeSound.org
Click here and hit ENTER to continue.

BUS driving recording from: Dobroide http://freeSound.org
Click here and hit ENTER to continue.


End of program.
```

Okay, so that's polymorphism. Big deal.
What is it good for?

Track Parking with ArrayList

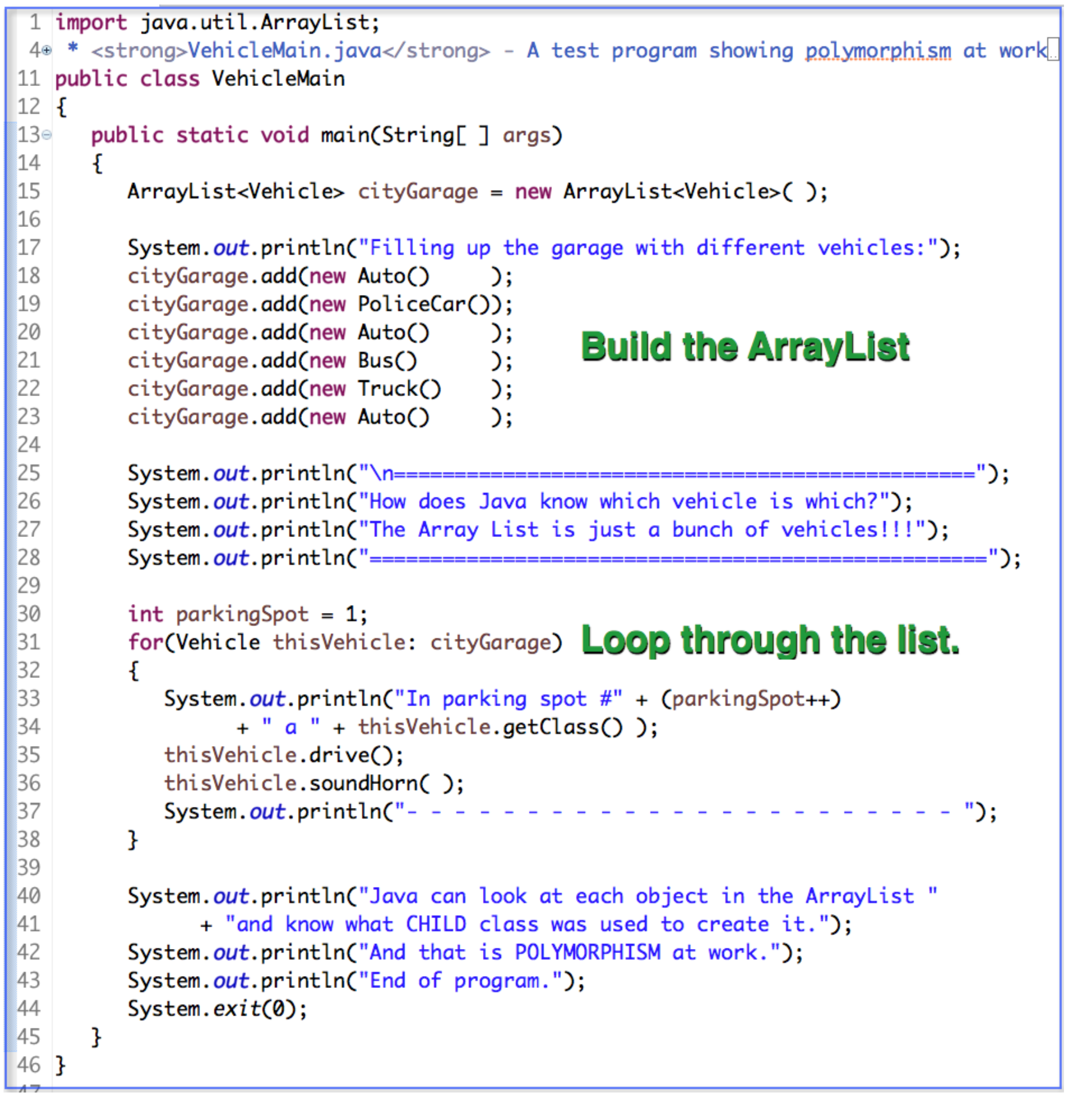
One of the benefits of having polymorphism built into a language such as Java is the ability to store different objects in a data structure such as an Array or ArrayList.
Keep in mind that only one type of object can be stored in a list.

In our program we are going to store Vehicles in the ArrayList:

```
ArrayList<Vehicle> cityGarage = new ArrayList<Vehicle>( );
```

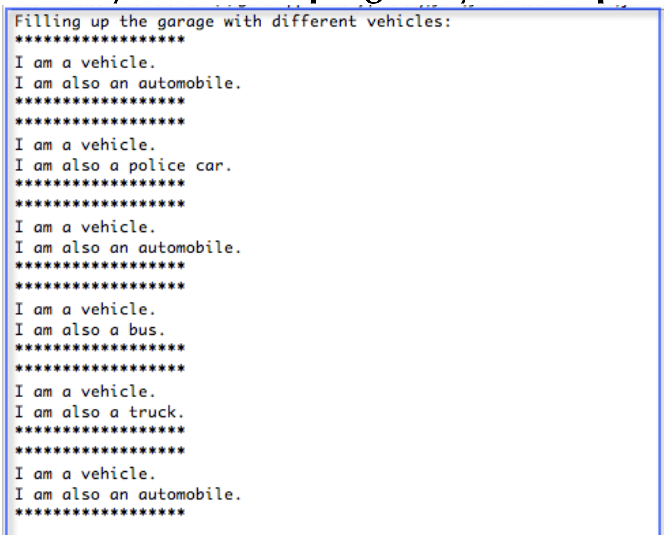
With polymorphism we can build an ArrayList of Vehicles, each one of them having its own distinct characteristics or overriding features.

Replace the code in the main( ) method of VehicleMain with this code:

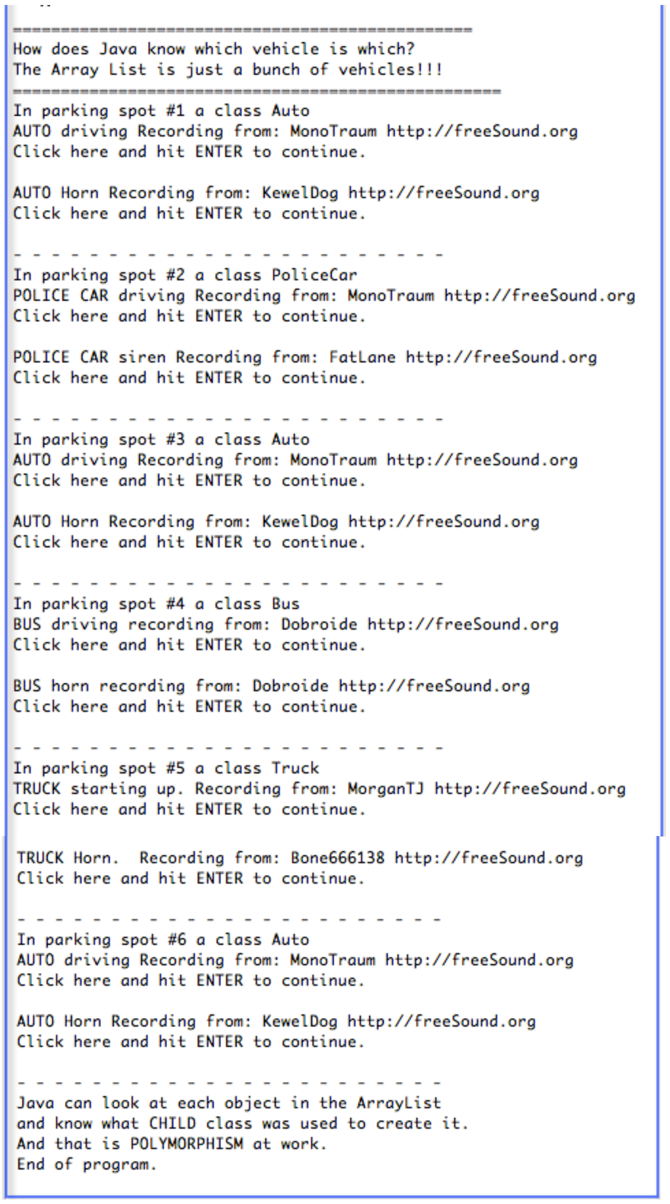


```
import java.util.ArrayList;
 * <strong>VehicleMain.java</strong> - A test program showing polymorphism at work
public class VehicleMain
{
   public static void main(String[ ] args)
   {
      ArrayList<Vehicle> cityGarage = new ArrayList<Vehicle>( );

      System.out.println("Filling up the garage with different vehicles:");
      cityGarage.add(new Auto()     );
      cityGarage.add(new PoliceCar());
      cityGarage.add(new Auto()     );
      cityGarage.add(new Bus()      );
      cityGarage.add(new Truck()    );
      cityGarage.add(new Auto()     );

      System.out.println("\n================================================");
      System.out.println("How does Java know which vehicle is which?");
      System.out.println("The Array List is just a bunch of vehicles!!!");
      System.out.println("==================================================");

      int parkingSpot = 1;
      for(Vehicle thisVehicle: cityGarage)
      {
         System.out.println("In parking spot #" + (parkingSpot++)
               + " a " + thisVehicle.getClass() );
         thisVehicle.drive();
         thisVehicle.soundHorn( );
         System.out.println("- - - - - - - - - - - - - - - - - - - - - - - ");
      }

      System.out.println("Java can look at each object in the ArrayList "
            + "and know what CHILD class was used to create it.");
      System.out.println("And that is POLYMORPHISM at work.");
      System.out.println("End of program.");
      System.exit(0);
   }
}
```

When you run the program your output should look something like this:



```
Filling up the garage with different vehicles:
******************
I am a vehicle.
I am also an automobile.
******************
******************
I am a vehicle.
I am also a police car.
******************
******************
I am a vehicle.
I am also an automobile.
******************
******************
I am a vehicle.
I am also a bus.
******************
******************
I am a vehicle.
I am also a truck.
******************
******************
I am a vehicle.
I am also an automobile.
******************
```



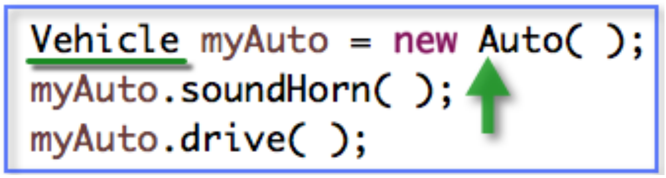
```
================================================
How does Java know which vehicle is which?
The Array List is just a bunch of vehicles!!!
===================================================
In parking spot #1 a class Auto
AUTO driving Recording from: MonoTraum http://freeSound.org
Click here and hit ENTER to continue.

AUTO Horn Recording from: KewelDog http://freeSound.org
Click here and hit ENTER to continue.

- - - - - - - - - - - - - - - - - - - - - - -
In parking spot #2 a class PoliceCar
POLICE CAR driving Recording from: MonoTraum http://freeSound.org
Click here and hit ENTER to continue.

POLICE CAR siren Recording from: FatLane http://freeSound.org
Click here and hit ENTER to continue.

- - - - - - - - - - - - - - - - - - - - - - -
In parking spot #3 a class Auto
AUTO driving Recording from: MonoTraum http://freeSound.org
Click here and hit ENTER to continue.

AUTO Horn Recording from: KewelDog http://freeSound.org
Click here and hit ENTER to continue.

- - - - - - - - - - - - - - - - - - - - - - -
In parking spot #4 a class Bus
BUS driving recording from: Dobroide http://freeSound.org
Click here and hit ENTER to continue.

BUS horn recording from: Dobroide http://freeSound.org
Click here and hit ENTER to continue.

- - - - - - - - - - - - - - - - - - - - - - -
In parking spot #5 a class Truck
TRUCK starting up. Recording from: MorganTJ http://freeSound.org
Click here and hit ENTER to continue.

TRUCK Horn.  Recording from: Bone666138 http://freeSound.org
Click here and hit ENTER to continue.

- - - - - - - - - - - - - - - - - - - - - - -
In parking spot #6 a class Auto
AUTO driving Recording from: MonoTraum http://freeSound.org
Click here and hit ENTER to continue.

AUTO Horn Recording from: KewelDog http://freeSound.org
Click here and hit ENTER to continue.

- - - - - - - - - - - - - - - - - - - - - - -
Java can look at each object in the ArrayList
and know what CHILD class was used to create it.
And that is POLYMORPHISM at work.
End of program.
```

How Does It Work?

How does Java know which vehicle is which? After all, the ArrayList is just a bunch of vehicles.

Polymorphism is not something you do. It is simply a description of how an object-oriented language handles parent and child classes.

When you create a new object such as myAuto,using a specific constructor like this:



```
Vehicle myAuto = new Auto( );
myAuto.soundHorn( );
myAuto.drive( );
```

you are telling Java, "myAuto is of type Vehicle." The constructor method is telling Java, "Use all the properties and methods in Auto. If there are duplicates the methods in Auto win (override) those in Vehicle."

For example, Java will know that myAuto will have all the Vehicle properties and methods as well as using the more specific methods such as soundHorn( ) and drive( ) that the Auto class has.

When you use the myAuto object to soundHorn( ) Java knows to use the overriding code in the child class, if it exists. Otherwise it will simply use the method found in the parent class, the generic soundHorn( ).

Other Practical Examples

There are lots of other ways you can leverage your knowledge of polymorphism.

You are writing a program with multiple forms. They all have the same header information such as the customer's name, address, phone, email, and id number.

You have a large inventory of vehicles such as the local Harley Davidson dealership. All the inventory items have a part number or serial number, description, quantity, stock date. You'll write classes to track different groups of items such as motorcycles, engine parts, accessories, clothing.

You are writing an ERP module for Human Resources (HR) and need to track different types of employees. They are all Employees (the base class) but each group of Employees will be a derived class such as Administrator, OfficeStaff, ShopFloor, Sales, IT support, and contract labor. Each type of employee gets paid differently (salary, hourly, base plus commission) and has different ways to earn PTO (Personal Time Off).

You are working on a graphic program and need to display different types of shapes and images based on what class they are.

You are working on a LMS (Learning Management System). Instructors can add information to the system but Students can't. Both are Users on the system. You can use polymorphism to automatically tell who can update the course materials and who can't by using the method: myObject.getClass( ).

Based on these examples think of two other ways that polymorphism would be useful in a software application.

How about in a video game?

Summary

Here are some of the skills you have practiced in this tutorial:

- Using sound files in Java with the InputStream, AudioStream, and AudioPlayer classes.
- Writing a generic method in the parent class.
- Overriding methods in the child classes.
- Using the concept of polymorphism to track different types of objects.

Along the way you also saw some examples of best-practices in coding:

✓ How to leverage println( ) statements to make it easier to track various parts of your program. For example, the println( ) statements in each constructor and the "End of program" statement.

✓ How to set up a "Hit any key" method in a base class that will be available to all its children.

✓ How to test out a small bit of code before adding it to your master project. (We created a SoundDemo project to learn how to process sound files. Then later we made a method out of this code using parameters to make it very reusable.)

✓ The SoundDemo project hard-coded in the sound filenames to speed testing. Later these were replaced in the method using parameters.

✓ SoundDemo handled only two sound files, a .wav and an .aif to see if either format worked. This kept testing to a manageable minimum.

✓ How to handle several errors using try/catch and following the exception hierarchy moving from specific to more general. (See: playSound( ) method)

✓ Formatting and commenting code for maximum readability, code clarity, and future maintainability.

✓ Include Web URL references for others working on this code in the future (including your future self).

✓ We followed a smart workflow:

(1) Testing specific techniques - SoundDemo - Answered the question, "Can Java do this???"
(2) General - Wrote and tested the Vehicle class.
(3) Specific - Wrote and test one child class.
(4) Duplicated and tested the other children.

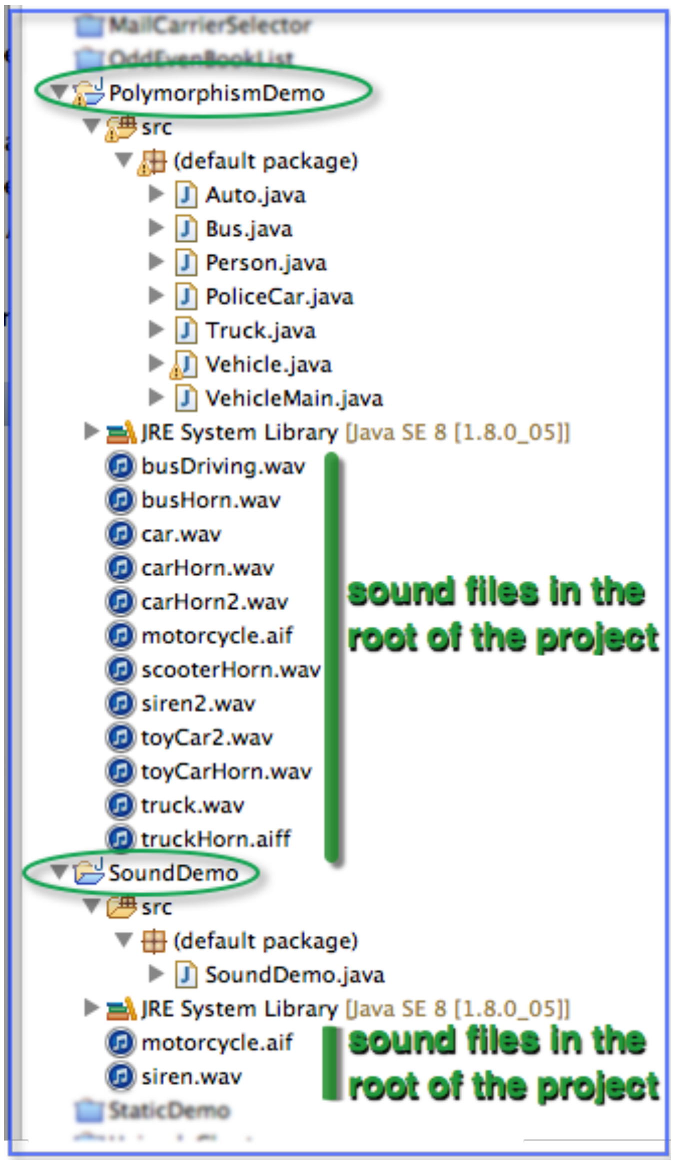
Source Code

Here is the source code for this tutorial.

Or your can download the file from webexplorations.com/book/java/PolymorphismSoundDemo.zip .
This zip file contains Auto.java, Bus.java, Person.java, PoliceCar.java, SoundDemo.java, Truck.java, Vehicle.java, and VehicleMain.java and all of the sound files.

Unzip the file and drag the files into an Eclipse project.

Your Package Explorer should look like this:



The classes are listed in alphabetical order.

Auto.java

```
/**
 * <strong>Auto.java</strong> - An auto vehicle
 *
 * @author Peter K. Johnson - <a href="http://
WebExplorations.com"
 *          target="_blank"> http://WebExplorations.com</a><br >
 *          Written: Oct 4, 2014<br >
 *          Revised: Oct 4, 2014<br>
 */
```

```
public class Auto extends Vehicle
{
   public Auto()
   {
      System.out.println("I am also an automobile.
\n*******************");
   }

   public void soundHorn( )
   {
     playSound("carHorn2.wav", "AUTO Horn Recording from:
KewelDog http://freeSound.org");
     hitAnyKey();
   }

   public void drive( )
   {
      setIsMoving(true);
      playSound("car.wav", "AUTO driving Recording from:
MonoTraum http://freeSound.org");
      super.hitAnyKey();
   }

}
```

Bus.java

```
/**
 * <strong>Bus.java</strong> - A bus vehicle
 *
 * @author Peter K. Johnson - <a href="http://
WebExplorations.com"
 *          target="_blank"> http://WebExplorations.com</a><br >
 *          Written: Oct 4, 2014<br >
 *          Revised: Oct 4, 2014<br>
 */

public class Bus extends Vehicle
{
   public Bus()
   {
      System.out.println("I am also a bus.
\n*******************");
   }

   public void soundHorn( )
   {
      playSound("busHorn.wav", "BUS horn recording from:
Dobroide http://freeSound.org");
     super.hitAnyKey();
   }

   public void drive( )
   {
      setIsMoving(true);
      playSound("busDriving.wav", "BUS driving recording from:
Dobroide http://freeSound.org");
      super.hitAnyKey();
   }
}
```

Person.java

```
import java.time.LocalDate;
// Java 7: import java.util.Date;

/**
 * <strong>Person.java</strong> - human
 *
 * @author Peter K. Johnson - <a href="http://
WebExplorations.com"
 *         target="_blank"> http://WebExplorations.com</a><br >
 *         Written: Sep 29, 2014<br >
 *         Revised: Sep 29, 2014<br>
 */

public class Person
{
  private String firstName = "";
  private String lastName = "";
  private String ssn = "";
  // Java 7: private Date dob = new Date( );
  // Java 8
  private LocalDate dob = LocalDate.now( );

  // Constructors
  public Person( ) { };

  public Person(String ssn)
  {
    this.ssn = ssn;
  }
  public Person(String ssn, String firstName, String lastName,
LocalDate dob)
  {
    this.ssn = ssn;
    this.firstName = firstName;
    this.lastName = lastName;
    this.dob = dob;
  }

  /**
  * toString( ) - Show the properties of a Person object
  */
 public String toString() {
    String result = "";
    result = String.format("*******  P E R S O N
****************\n");
    result += String.format("Name:  %s %s\n", firstName,
lastName);
    result += String.format("SSN:  %s\n", ssn);
    result += String.format("Birthday: %s\n", dob.toString( ));
    result += String.format("**************************\n");
    return result;
 }

 // ================= SET METHODS ====================
  public void setFirstName(String fname) { firstName = fname; }
  public void setLastName(String lname) { lastName = lname; }
  public void setSSN(String ssn) { this.ssn = ssn; }
  public void setDOB(LocalDate dob) { this.dob = dob; }

  // ================= GET METHODS ====================
  public String getFirstName( ) { return firstName; }
  public String getLastName( ) { return lastName; }
  public String getSSN( ) { return ssn; }
  public LocalDate getDOB( ) { return dob; }
} // end of Person
```

PoliceCar.java

```
/**
 * <strong>PoliceCar.java</strong> - A police car vehicle
 *
 * @author Peter K. Johnson - <a href="http://
WebExplorations.com"
```

```
 *          target="_blank"> http://WebExplorations.com</a><br >
 *          Written: Oct 4, 2014<br >
 *          Revised: Oct 4, 2014<br>
 */

public class PoliceCar extends Vehicle
{
   public PoliceCar()
   {
      System.out.println("I am also a police car.
\n******************");
   }

   public void soundHorn( )
   {
     playSound("siren2.wav", "POLICE CAR siren Recording from:
FatLane http://freeSound.org");
     super.hitAnyKey();
   }

   public void drive( )
   {
      setIsMoving(true);
      playSound("car.wav", "POLICE CAR driving Recording from:
MonoTraum http://freeSound.org");
      super.hitAnyKey();
   }
}
```

SoundDemo.java

```
import java.io.*;
import sun.audio.*;

/**
 * A simple Java sound file example (i.e., Java code to play a
sound file).
 * AudioStream and AudioPlayer code comes from a javaworld.com
example.
 * @author Alvin Alexander, devdaily.com.
 * http://alvinalexander.com/java/java-audio-example-java-au-
play-sound
 * Creative Commons sounds can be down loaded at: http://
freesound.org
 */
public class SoundDemo
```

```
{
   public static void main(String[ ] args) throws Exception
  {
     // open the sound file as a Java input stream
     InputStream in = new FileInputStream("motorcycle.aif");
     // create an audiostream from the inputstream
     AudioStream audioStream = new AudioStream(in);

     // play the audio clip with the audioplayer class
     AudioPlayer.player.start(audioStream);
     System.out.println("Sound Test");
  }
}
```

Truck.java

```
/**
 * <strong>Truck.java</strong> - A Truck Vehicle
 *
 * @author Peter K. Johnson - <a href="http://
WebExplorations.com"
 *         target="_blank"> http://WebExplorations.com</a><br >
 *         Written: Oct 4, 2014<br >
 *         Revised: Oct 4, 2014<br>
 */

public class Truck extends Vehicle
{
   public Truck()
   {
      System.out.println("I am also a truck.
\n*****************");
   }

   public void soundHorn( )
   {
     playSound("truckHorn.aiff", "TRUCK Horn.  Recording from:
Bone666138 http://freeSound.org");
     super.hitAnyKey();
   }

   public void drive( )
   {
      setIsMoving(true);
      playSound("truck.wav", "TRUCK starting up. Recording from:
MorganTJ http://freeSound.org");
```

```
      super.hitAnyKey();
   }
}
```

Vehicle.java

```
import java.io.*;
import java.util.Scanner;
import sun.audio.*;
/**
 * <strong>Vehicle.java</strong> - A general default vehicle
 *
 * @author Peter K. Johnson - <a href="http://
WebExplorations.com"
 *          target="_blank"> http://WebExplorations.com</a><br >
 *          Written: Oct 4, 2014<br >
 *          Revised: Oct 4, 2014<br>
 */
public class Vehicle
{

   // ================== PROPERTIES ===========================
   private float direction = 0.0F; // 0=north 90=east 180=south
270=west
   private int mileage = 0;
   private int speed = 0;
   private boolean isMoving = false;
   private Person driver = new Person( );

   // ================= CONSTRUCTOR METHODS ====================
   // default constructor
   public Vehicle()
   {
      System.out.println("******************\nI am a vehicle.");
   }

   public Vehicle(int mileage)
   {
      System.out.println("******************\nI am a vehicle.");
      this.mileage = mileage;
   }

    /**
    * toString( ) - Show the properties of a vehicle object
    */
   public String toString() {
      String result = "";
      result = String.format("***** %s ********************\n",
this.getClass().getSimpleName() );
      result += String.format("Mileage:  %s\n", mileage);
      result += String.format("Speed:  %s\n", speed);
      result += String.format("Is Moving: %s\n", isMoving);
      result += String.format("Direction: %.0f degrees.\n",
direction);
      result += String.format("Driver is: %s %s.\n",
driver.getFirstName( ), driver.getLastName());
      return result;
   }

   public void soundHorn( )
   {
     playSound("toyCarHorn.wav", "Toy Car Horn Recording from:
BeatsByCasper http://freeSound.org");
     this.hitAnyKey();
   }

   public void drive( )
   {
      isMoving = true;
      playSound("toyCar2.wav", "Toy Car Driving Recording from:
AnnaBloom http://freeSound.org");
      this.hitAnyKey();
   }

  /* http://alvinalexander.com/java/java-audio-example-java-au-
play-sound
   * Creative Commons sounds: http://www.freesound.org
   */
   public void playSound(String thisSound, String description)
   {
         try {
      // Open the sound file as a Java input stream
      InputStream in = new FileInputStream(thisSound);
       // create an audioStream from the inputStream
      AudioStream audioStream = new AudioStream(in);
      // Play the audio clip with the audioPlayer class
      AudioPlayer.player.start(audioStream);
      System.out.println(description);
      }
         catch(FileNotFoundException e)
         {
            System.out.println(thisSound + " cannot be found." +
e);
         }
         catch(IOException e)
         {
              System.out.println(thisSound + " is not a valid
sound file." + e);
         }
         catch(Exception e)
         {
            System.out.println("There was an error: " + e);
         }
   }// end of playSound( )

   public void hitAnyKey( )
   {
      Scanner keyIn = new Scanner(System.in);
      System.out.println("Click here and hit ENTER to
continue.");
      keyIn.nextLine( );
   }

   // ================= GET METHODS ====================
   public float getDirection( ){ return direction; }
   public int getMileage( ){ return mileage; }
   public int getSpeed( ){ return speed; }
   public boolean getIsMoving( ){ return isMoving; }
   public Person getPerson( ){ return driver; }

   // ================= SET METHODS ====================
   public void setDirection(float direction) {this.direction =
direction;}
   public void setMileage(int mileage) {this.mileage = mileage;}
   public void setSpeed(int speed) {this.speed = speed;}
   public void setIsMoving(boolean isMoving) {this.isMoving =
isMoving;}
   public void setPerson(Person person) { driver = person; }
} // end of Vehicle
```

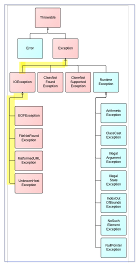
VehicleMain.java

```
import java.util.ArrayList;

/**
 * <strong>VehicleMain.java</strong> - A test program showing
polymorphism at work
```

```
 *
 * @author Peter K. Johnson - <a href="http://
WebExplorations.com"
 *         target="_blank"> http://WebExplorations.com</a><br >
 *         Written: Oct 4, 2014<br >
 *         Revised: Oct 4, 2014<br>
 */
public class VehicleMain
{
   public static void main(String[ ] args)
   {
      ArrayList<Vehicle> cityGarage = new ArrayList<Vehicle>( );

      System.out.println("Filling up the garage with different
vehicles:");
      cityGarage.add(new Auto()     );
      cityGarage.add(new PoliceCar());
      cityGarage.add(new Auto()     );
      cityGarage.add(new Bus()      );
      cityGarage.add(new Truck()    );
      cityGarage.add(new Auto()     );


System.out.println("\n=========================================
======");
      System.out.println("How does Java know which vehicle is
which?");
      System.out.println("The Array List is just a bunch of
vehicles!!!");

System.out.println("==========================================
=======");

      int parkingSpot = 1;
      for(Vehicle thisVehicle: cityGarage)
      {
         System.out.println("In parking spot #" + (parkingSpot+
+)
               + " a " + thisVehicle.getClass() );
         thisVehicle.drive();
         thisVehicle.soundHorn( );
         System.out.println("- - - - - - - - - - - - - - - - - -
- - - - - ");
      }

      System.out.println("Java can look at each object in the
ArrayList "
            + "\nand know what CHILD class was used to create
it.");
      System.out.println("And that is POLYMORPHISM at work.");
      System.out.println("End of program.");
      System.exit(0);
   }
}
```

Credits

City Vehicles Police Car http://www.mankato-mn.gov/Central-Garage


Exception Hierarchy http://www.programcreek.com/wp-content/uploads/2009/02/Exception-Hierarchy-Diagram.jpeg

Last update: 10/06/2014