1. Log in
2. Unit 4
   - Writing Methods
   - Recursion
   - Classes and Objects

Mar 9-11:28 AM

Object-Oriented Programming ...

Objects - Things that are created in memory!

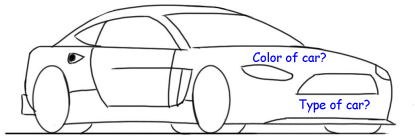
Every program creates and/or manipulates some sort of object. We now need to begin to create and manipulate our own objects!

- The creation and manipulation of actual virtual objects.
- Think of a car, say we want to build a car and ...

Type of car? Color of car? Brake! Turn on lights! Accelerate! Turn on wipers! Start the car!

Apr 21-1:43 PM

Our digital car starts to look like this ...



Running?
Brake?
Lights?
Wipers?

Before we can build our first car, let's review something we should know ...

Apr 21-1:43 PM

Scanner Object Creation!

```
Scanner getNumber = new Scanner(System.in);
```

type of object (a scanner) | name of the object (scanner name) | required to make a new scanner object | parameter = stuff handed to the object

* This is how we built a Scanner object.

* We can create our own objects ... like a car! BUT ... JAVA has a Scanner "builder" for us, we have to make our own "car-builder" first!

May 18-1:29 PM

How to make our "Car Builder"

1. Make a new Class called Car ...

```
public class Car
{

}
```

NO public static void main ... for now! this is our Class

This Car Class is going to be the class that we will use to build and manipulate all of our cars!

Apr 21-1:43 PM

How to make our "Car Builder"

2. Create the variables you want to track on each car ...

```
public class Car
{
    public int year;
    public String type;

}
```

These are called INSTANCE VARIABLES

Every car object we build will contain this specific information about itself!

Notice: We will just store the year and type for now.

This code IS NOT in the main method ... not a list of directions, it is object info!

Apr 21-1:43 PM

How to make our "Car Builder"

2. Create the variables you want to track on the car ...

```
public class Car
{
    public int year;
    public String type;

}
```

Variables are "public" because we want all other parts of our program and any other classes/methods that we create to be able to access them.
Public = open for anything to access/use

Apr 21-1:43 PM

How to make our "Car Builder"

3. Create the constructor method that will build the cars for us!

```
public class Car
{
    public int year;
    public String type;

    public Car( )
    {
        year = ?;  //not known yet
        type = ?;  //not known yet
    }
}
```

Remember that a method is like a "mini-program" that does something for us? We need to make the method that builds the cars for us!

For now each car will be built with a year and type label attached to it.

Apr 21-1:43 PM

How to make our "Car Builder"

3. Create the constructor method that will build the cars for us!

```
public class Car
{
    public int year;
    public String type;

    public Car( )
    {
        year = ???
        type = ???
    }
}
```

The constructor method name MUST have the same name as the class name!!!

Apr 21-1:43 PM

How to make our "Car Builder"

4. Add the parameters (what is the car's information???)

```
public class Car
{
    public int year;
    public String type;

    public Car(int carYear, String carType)
    {
        year = carYear;
        type = carType;
    }
}
```

Notice: The Car method needs to be told what the carYear and carType is, it will then build the car and store the information into the car!!!

The car method needs 2 parameters - year and type!

Apr 21-1:43 PM

Congratulations ... you've made a "Car Builder"!

```
public class Car
{
    public int year;
    public String type;

    public Car(int carYear, String carType)
    {
        year = carYear;
        type = carType;
    }
}
```

* This is now called our Car class.
* It contains a Car constructor method that builds cars for us.
* Currently the only information we will have about any car we make will be the year and type.

Apr 21-1:43 PM

Review How to Make a Scanner Object One More Time ...

```
Scanner getNumber = new Scanner(System.in);
```

- type of object (a scanner)
- name of the object (scanner name) YOU NAME IT!
- required to make a new scanner object
- parameter(s) = stuff handed to the object

* We are ready to make a Car Object!!!

May 18-1:29 PM

Scanners

```
Scanner getNumber = new Scanner(System.in);
```

type of object (a scanner) | name of the object (scanner name) | required to make a new scanner object | parameter = stuff handed to the object

Cars

```
Car my1stCar = new Car(an integer, a String);
```

type of object (a Car) | name of the object (Car name) | required to make a new Car object | parameters = stuff needed to build the Car

May 18-1:29 PM

We can now use the Car class to construct cars!!!

Step #1:

Make a new class called:

elmoYouCanDriveMyCar

```
package Lesson15;
/**
 * @author Matthew_Henderson
 */
public class elmoYouCanDriveMyCar {

}
```

Apr 21-1:43 PM

Step #2: *This is our main program now ... we will just need to access the Car class here!*

```
package Lesson15;

/**
 * @author Matthew_Henderson
 */
public class elmoYouCanDriveMyCar {
    public static void main(String[] args) {

    }
}
```

Apr 21-1:43 PM

Step #3: *Create a Car!*

```
package Lesson15;

/**
 * @author Matthew_Henderson
 */
public class elmoYouCanDriveMyCar {
    public static void main(String[] args) {
        Car myFirstCar = new Car(1989, "Escort");
    }
}
```

instance of car class | Build a new car called myFirstCar | Make it a 1989 Escort | BOOM! We have a 1989 Escort in memory!

Apr 21-1:43 PM

Step #4: *Create a 2nd Car ... 2015 Porsche!*

```
package Lesson15;

/**
 * @author Matthew_Henderson
 */
public class elmoYouCanDriveMyCar {
    public static void main(String[] args) {
        Car myFirstCar = new Car(1989, "Escort");
        Car myCurrentCar = new Car(2015, "Porsche");
    }
}
```

another instance | Build a new car called myCurrentCar | Make it a 2015 Porsche | BOOM! We have a 2015 Porsche in memory!

Apr 21-1:43 PM

NOTICE: We have written 2 different programs ...

Car Constructor Class

```
package Lesson15;

/**
 * @author Matthew_Henderson
 */
public class Car
{
        public int year;
        public String type;

        public Car(int carYear, String carType)
        {
                year = carYear;
                type = carType;
        }
}
```

elmoYouCanDriveMyCar Class
The Main program!

```
package Lesson15;

/**
 * @author Matthew_Henderson
 */
public class elmoYouCanDriveMyCar {
    public static void main(String[] args) {
        Car myFirstCar = new Car(1989, "Escort");
        Car myCurrentCar = new Car(2015, "Porsche");
    }
}
```

The main program will use the Car constructor class when needing to build a Car object.

Apr 21-1:43 PM

EXAMINE THE MAIN METHOD:

* It builds 2 cars but it doesn't do or show us anything!!!

```
package Lesson15;

/**
 * @author Matthew_Henderson
 */
public class elmoYouCanDriveMyCar {
   public static void main(String[] args) {
      Car myFirstCar = new Car(1989, "Escort");
      Car myCurrentCar = new Car(2015, "Porsche");
   }
}
```

So, let's just write a simple program that asks the user which car theywant to view.

Apr 21-1:43 PM

Our Simple Program

Step #1: Ask the user which car they want to view.

```
package Lesson15;

/**
 * @author Matthew_Henderson
 */
public class elmoYouCanDriveMyCar {
   public static void main(String[] args) {
      Car myFirstCar = new Car(1989, "Escort");
      Car myCurrentCar = new Car(2015, "Porsche");
      System.out.println("Which car do you want to view (1 or 2): ");
   }
}
```

Apr 21-1:43 PM

Step #2: Set up a Scanner to get the user input.

```
package Lesson15;
import java.util.*;

/**
 * @author Matthew_Henderson
 */
public class elmoYouCanDriveMyCar {
   public static void main(String[] args) {
      Car myFirstCar = new Car(1989, "Escort");
      Car myCurrentCar = new Car(2015, "Porsche");
      System.out.println("Which car do you want to view (1 or 2): ");
      Scanner getUserNumber = new Scanner(System.in);
   }
}
```

Apr 21-1:43 PM

Step #3: Get the user input.

```
package Lesson15;
import java.util.*;

/**
 * @author Matthew_Henderson
 */
public class elmoYouCanDriveMyCar {
   public static void main(String[] args) {
      Car myFirstCar = new Car(1989, "Escort");
      Car myCurrentCar = new Car(2015, "Porsche");
      System.out.println("Which car do you want to view (1 or 2): ");
      Scanner getUserNumber = new Scanner(System.in);
      int userNumber = getUserNumber.nextInt( );
   }
}
```

Apr 21-1:43 PM

Step #4: Use an else-if statement to output correct information.

```
package Lesson15;
import java.util.*;

/**
 * @author Matthew_Henderson
 */
public class elmoYouCanDriveMyCar {
   public static void main(String[] args) {
      Car myFirstCar = new Car(1989, "Escort");
      Car myCurrentCar = new Car(2015, "Porsche");
      System.out.println("Which car do you want to view (1 or 2): ");
      Scanner getUserNumber = new Scanner(System.in);
      int userNumber = getUserNumber.nextInt( );
      if(userNumber==1)
         System.out.println("You are viewing a "+myFirstCar.year+" "+myFirstCar.type);
      else if(userNumber==2)
         System.out.println("You are viewing a "+myCurrentCar.year+" "+myCurrentCar.type);
      else
         System.out.println("That is not a valid choice!");
   }
}
```

These are called **Object References** Variables that represent a specific object

Apr 21-1:43 PM

myFirstCar.year

which object?

What do you want to know about it?

myCurrentCar.type

which object?

What do you want to know about it?

What will these return???

Nov 6-9:56 AM

A CRUCIAL COMPARISON!!!

```
getUserNumber.nextInt( );
```
look at the scanner object called getUserNumber

have it run the nextInt() method that it contains

```
myFirstCar.year
```
look at the Car object called myFirstCar

grab the year variable about that car

```
myFirstCar.type
```
look at the Car object called myFirstCar

grab the type variable about that car

```
objectName.whatToLookAt
```
What object do you want to look at?

What variable or method in that object do you want to access?

Apr 21-1:43 PM

Time to upgrade our Car objects!!!

```
package Lesson15;

/**
 * @author Matthew_Henderson
 */
public class Car
{
        public int year;
        public String type;

        public Car(int carYear, String carType)
        {
                year = carYear;
                type = carType;
        }
}
```

Notice: Our cars are pretty boring. All we can do is make them and keep track of the year they ware made and what type of car they are.

WE NEED SOME ACTION!!!

May 19-10:59 AM

We should be able to turn the car on and off, shouldn't we?

```
package Lesson15;

/**
 * @author Matthew_Henderson
 */
public class Car
{
        public int year;
        public String type;
        public boolean running;

        public Car(int carYear, String carType)
        {
                year = carYear;
                type = carType;
                running = false;
        }
}
```

We need a new variable that will keep track of whether or not the car is running!

boolean is a good choice here:

true = car is running
false = car is not running

Notice: All cars that are created will not be running, so we don't need a parameter for it! We don't want our newly created cars driving away randomly on their own :-)

May 19-10:59 AM

So now every car that we create has a "running status" ...

```
package Lesson15;
import java.util.*;

/**
 * @author Matthew_Henderson
 */
public class elmoYouCanDriveMyCar {
   public static void main(String[] args) {
      Car myFirstCar = new Car(1989, "Escort");
      Car myCurrentCar = new Car(2015, "Porsche");
      System.out.println("Which car do you want to view (1 or 2): ");
      Scanner getUserNumber = new Scanner(System.in);
      int userNumber = getUserNumber.nextInt( );
      if(userNumber==1)
         System.out.println("You are viewing a "+myFirstCar.year+" "+myFirstCar.type);
      else if(userNumber==2)
         System.out.println("You are viewing a "+myCurrentCar.year+" "+myCurrentCar.type);
      else
         System.out.println("That is not a valid choice!");
   }
}
```

Remember our main method (program)?

May 19-10:59 AM

So now every car that we create has a "running status" ...

```
public class elmoYouCanDriveMyCar {
   public static void main(String[] args) {
      Car myFirstCar = new Car(1989, "Escort");
      Car myCurrentCar = new Car(2015, "Porsche");
      System.out.println("Which car do you want to view (1 or 2): ");
      Scanner getUserNumber = new Scanner(System.in);
      int userNumber = getUserNumber.nextInt( );
      if(userNumber==1) {
         System.out.println("You are viewing a "+myFirstCar.year+" "+myFirstCar.type);
         if (myFirstCar.running==true)
            System.out.println("This car is running!");
         else
            System.out.println("This car is not running.");
      }

      else if(userNumber==2)
         System.out.println("You are viewing a "+myCurrentCar.year+" "+myCurrentCar.type);
      else
         System.out.println("That is not a valid choice!");
   }
}
```

We should communicate whether or not the car is running!

May 19-10:59 AM

So, we have cars that are not running ... how boring!

How can we start one of our cars?

1. We need to manipulate the car!
2. We need to change the running status!
3. We need a method (mini-program) that does it for us!
4. We need a method added to our Car class!

May 19-10:59 AM

Here's the method ... it is simple!

```
public void startCar( )
{
        running = true;
}
```

This method simply turns the car to a "running" state! Turns it on!

Public - Remember it simply means this method is open for all to use!

void - For now, void means that we are making changes ... not having the method output anything. (in other words nothing is being returned by startCar() ... just changing the state of running)

startCar( ) - Simply the name of our method (no parameters needed!). We made our own method!

May 19-10:59 AM

Our startCar( ) method needs to be in the Car class!

```
package Lesson15;
/**
 * @author Matthew_Henderson
 */
public class Car
{
        public int year;
        public String type;
        public boolean running;

        public Car(int carYear, String carType)
        {
                year = carYear;
                type = carType;
                running = false;
        }

        public void startCar( )
        {
                running = true;
        }
}
```

This method simply turns the car to a "running" state! Turns it on!

May 19-10:59 AM

Now is the time! Does the user want to start the car?

```
public class elmoYouCanDriveMyCar {
   public static void main(String[] args) {
      ...
   int userNumber = getUserNumber.nextInt( );
   if(userNumber==1) {
      System.out.println("You are viewing a "+myFirstCar.year+" "+myFirstCar.type);
      if (myFirstCar.running==true)
         System.out.println("This car is running!");
      else {
         System.out.println("This car is not running. Would you like to start it (1=Yes, 2=No): ");
         int userStart = getUserNumber.nextInt();
         if(userStart==1)
         {
            myFirstCar.startCar();
            if (myFirstCar.running==true)
               System.out.println("The "+ myFirstCar.type+" is now running!");
         }
      }
   ...
```

We should communicate whether or not the car is running!

object reference!

May 19-10:59 AM

Now, if a car is running ... we need to be able to turn it off!

Do you think you could do this on your own now?

Can you write the method called turnCarOff?

May 19-10:59 AM

Here's the method ... it is simple!

```
public void turnCarOff( )
{
        running = false;
}
```

This method simply turns the car to a "no-running" state! Turns it off!

Public - Remember it simply means this method is open for all to use!

void - For now, void means that we are making changes ... not having the method output anything. (in other words nothing is being returned by startCar() ... just changing the state of running)

turnCarOff( ) - Simply the name of our method (no parameters needed!). We made our own method!

May 19-10:59 AM

Our startCar( ) method needs to be in the Car class!

```
package Lesson15;
public class Car
{
        public int year;
        public String type;
        public boolean running;

        public Car(int carYear, String carType) {
                year = carYear;
                type = carType;
                running = false;
        }

        public void startCar( ) {
                running = true;
        }

        public void turnCarOff( ) {
                running = false;
        }

}
```

This method simply turns the car to a "non-running" state! Turns it off!

May 19-10:59 AM

Things to do ...

1. Do you have Unit 4 WS01-04 completed?
2. Begin work on Unit 04 WS05 - Classes & Objects 1

May 19-2:31 PM