

Something new:
Make a Class...

Mr. Neat
Java

What if we wanted 2
Cars? Or 3, 4, 10,...

Wouldn't it be nice if?....

Car fred;

fred = new Car(x,y);

We need to make the class Car!

Make a Class

So far we have USED existing classes:

- Rectangle
- Text
- EasyReader
- String

Somebody else wrote all of these classes

Make a Class

The way you know you are using a Class is the reserved word *new*

- *new* Rectangle(10,12,14,16);
- *new* Text(12, 14, “pizza”);
- *new* EasyReader();
- *new* String(“cheese”);

Ta - Da, now we are
going to make a Class!

- Classes are in separate files
- the name of the class is the name
of the file

Ta - Da, now we are going to make a Class!

in a separate file called Car.java, (in the same folder)

```
public class Car
{

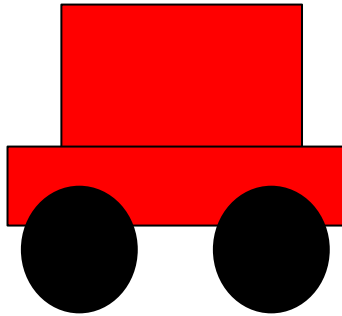
}

```

Ta - Da, now we are going to make a Class!

- the class definition has three parts:
 - 1) fields (also called global variables or instance variables)
 - 2) constructors
 - 3) methods


What are some variables that define the Car?



Ta - Da, now we are going to make a Class!

in a separate file called Car.java, (in the same
folder)

```
public class Car
{
    private Rectangle upperBody;
}
```



reserved word meaning
only can use this
variable in this class

Ta - Da, now we are going to make a Class!

in a separate file called Car.java, (in the same
folder)

```
public class Car
{
    private Rectangle upperBody;
    private Rectangle lowerBody;
    private Ellipse wheel1;
    ...
}
```

Ta - Da, now we are going to make a Class!

- the class definition has three parts:
 - 1) fields (also called global variables or instance variables)
 - 2) constructors
 - 3) methods

Constructors.....

Now dealing with two files!

One file **USES** the Car

The other file **MAKES** the Car

Constructors.....

In the file that **USES** the Car:

```
public class starter...  
{
```

```
    public static void main(String args[])  
    {
```

```
        Car fred = new Car(17.0, 23.0);
```

these values have to get to the
Car.java file!



Constructors.....

In the file that **MAKES** the Car:

```
public class Car  
{  
    // fields up here
```

```
    public Car(double ex, double why)  
    {  
        // create Car parts here
```

the constructor
definition



Shows Both Files...

USES - starter.java

```
Car fred = new Car(17.0, 23.0);
```

MAKES - Car.java

no void!

same name as class
& file

```
public Car(double x, double y)
{
}
}
```

“formal parameters”

Constructors.....

In the file that **MAKES** the Car:

```
public class Car  
{
```

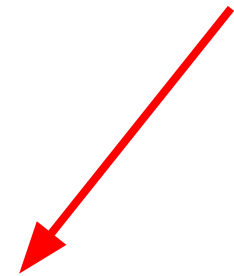
```
    private Rectangle upperBody;
```

```
    public Car(double ex, double why)  
    {
```

```
        upperBody = new Rectangle(ex, why,...);
```

```
        ...
```

make the Car!



Ta - Da, now we are going to make a Class!

- the class definition has three parts:
 - 1) fields (also called global variables or instance variables)
 - 2) constructors
 - 3) methods

briefly...


Shows Both Files...

USES - starter.java

```
Car fred = new Car(17.0, 23.0);  
fred.draw();
```

MAKES - Car.java

```
public class Car  
{  
    // fields and constructor  
    public void draw()  
    {  
  
    }  
}
```



define method draw()

Make the draw() method...

Draw each part of the Car

```
public void draw()  
{  
    upperBody.draw();  
    ...  
}
```

Now in the starter.java class,
you can construct a
Car,...or a bunch of
Cars!

```
new Car(50.,5.,canvas);  
new Car(32.,12.,canvas);  
new Car(43.,21.,canvas);  
new Car(3.,17.,canvas);  
new Car(57.,32.,canvas);
```

Lab

- Complete your Car constructor & draw() method
- Construct a bunch of Cars at different locations (test it)
- Add more features to your Car (spoilers, windows,...) if you want and have time