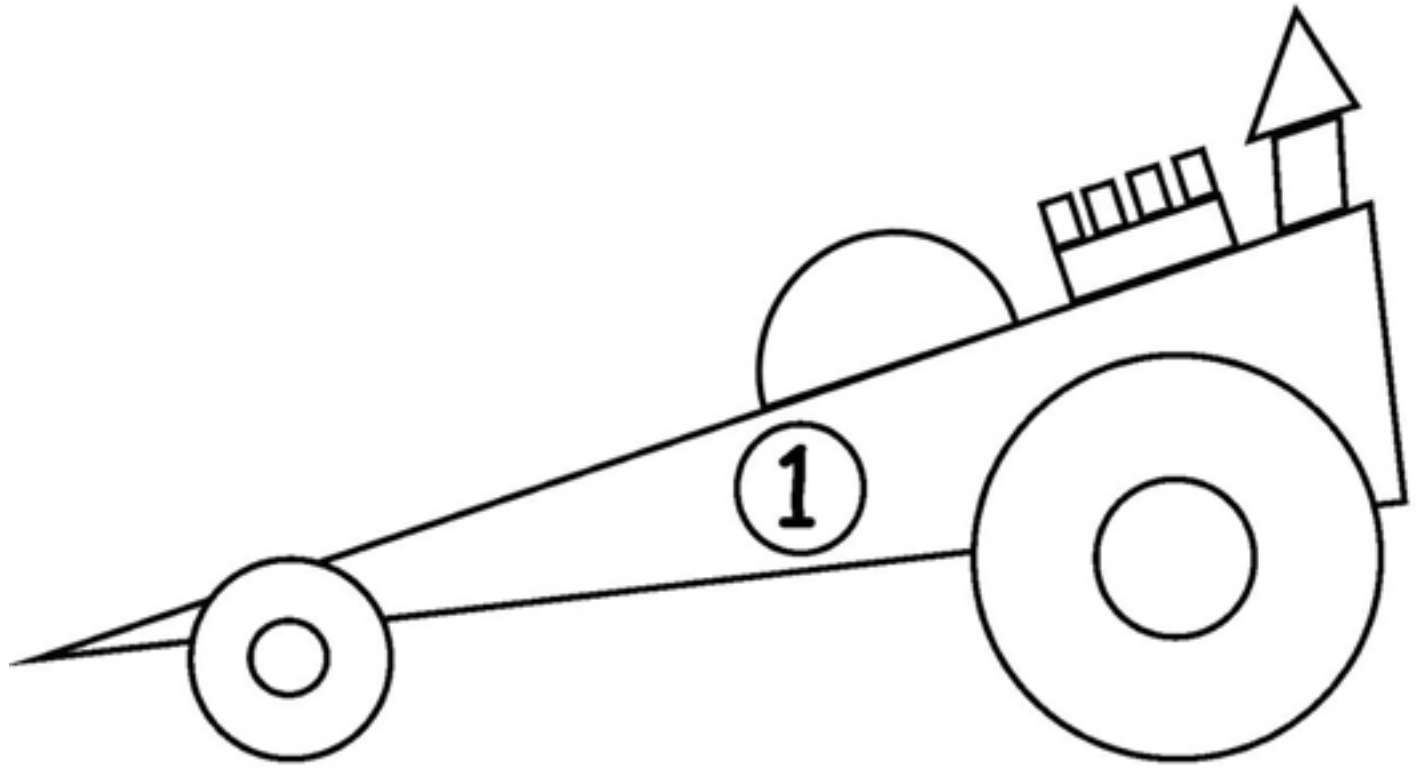
Object Oriented Programming (OOP)

UC Berkeley Graduate School of Journalism

Why we call them "objects"



Why we call them "objects"

Properties (adjectives)

Methods (verbs)

Events (triggers)

Why we call them "objects"

```
car.color = "blue"
```

Properties (adjectives)

Methods (verbs)

Events (triggers)

Why we call them "objects"

```
car.color = "blue"
```

Properties (adjectives)

```
car.start()
```

Methods (verbs)

Events (triggers)

Why we call them "objects"

```
car.color = "blue"
```

Properties (adjectives)

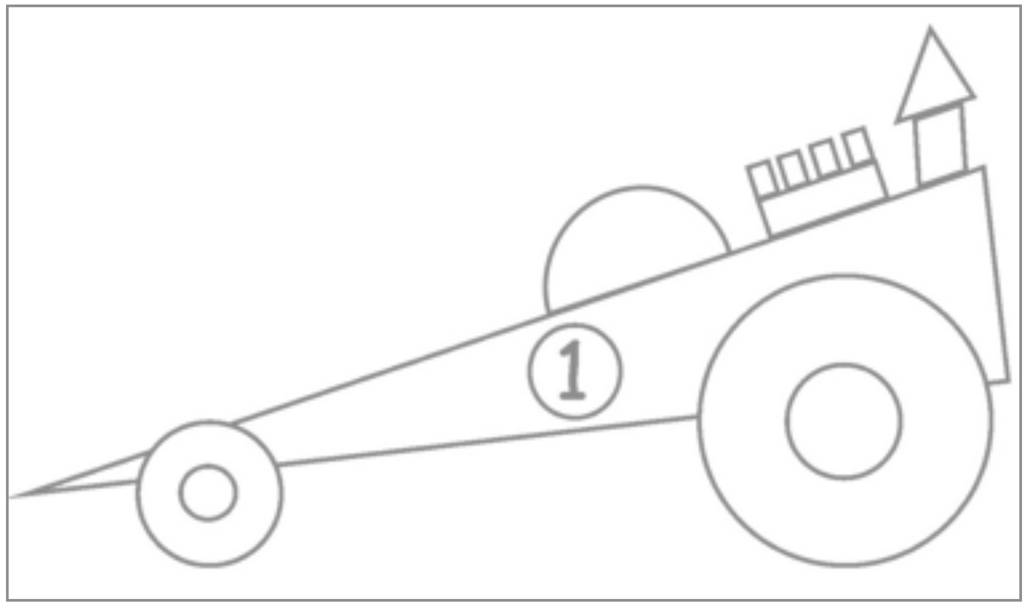
```
car.start()
```

Methods (verbs)

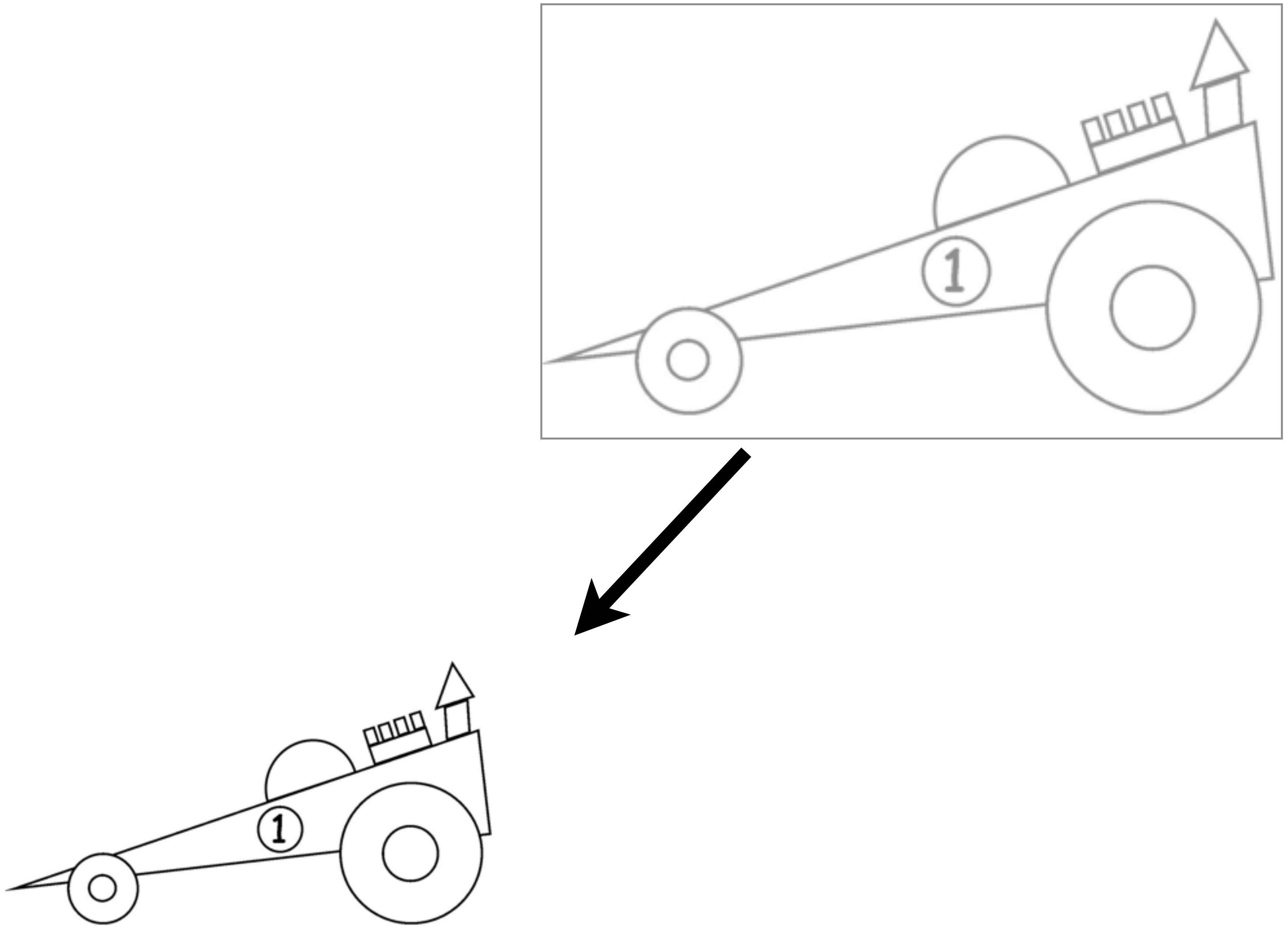
```
car.addEventListener("move", doSomething())
```

Events (triggers)

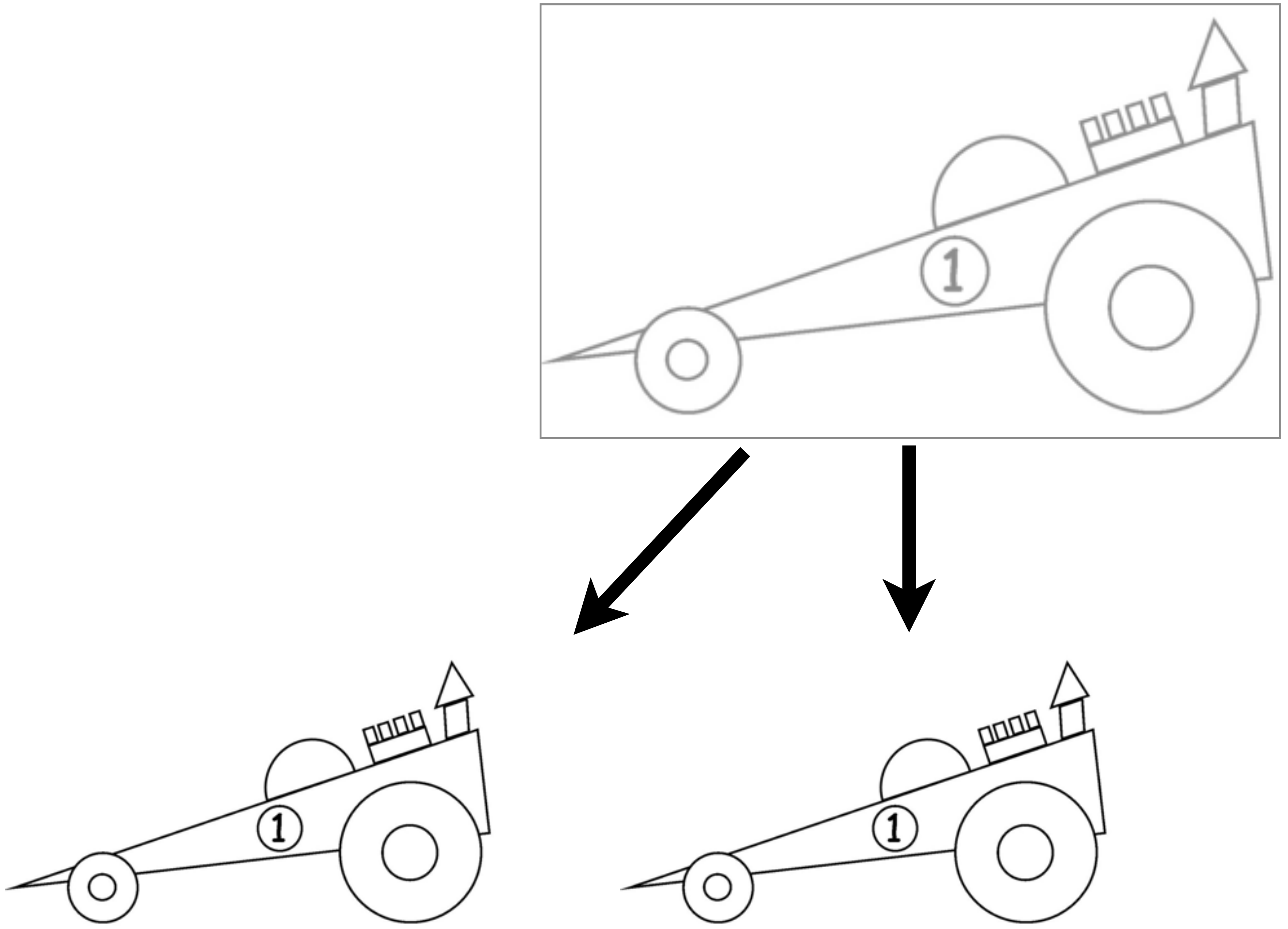
Classes (like a blueprint)



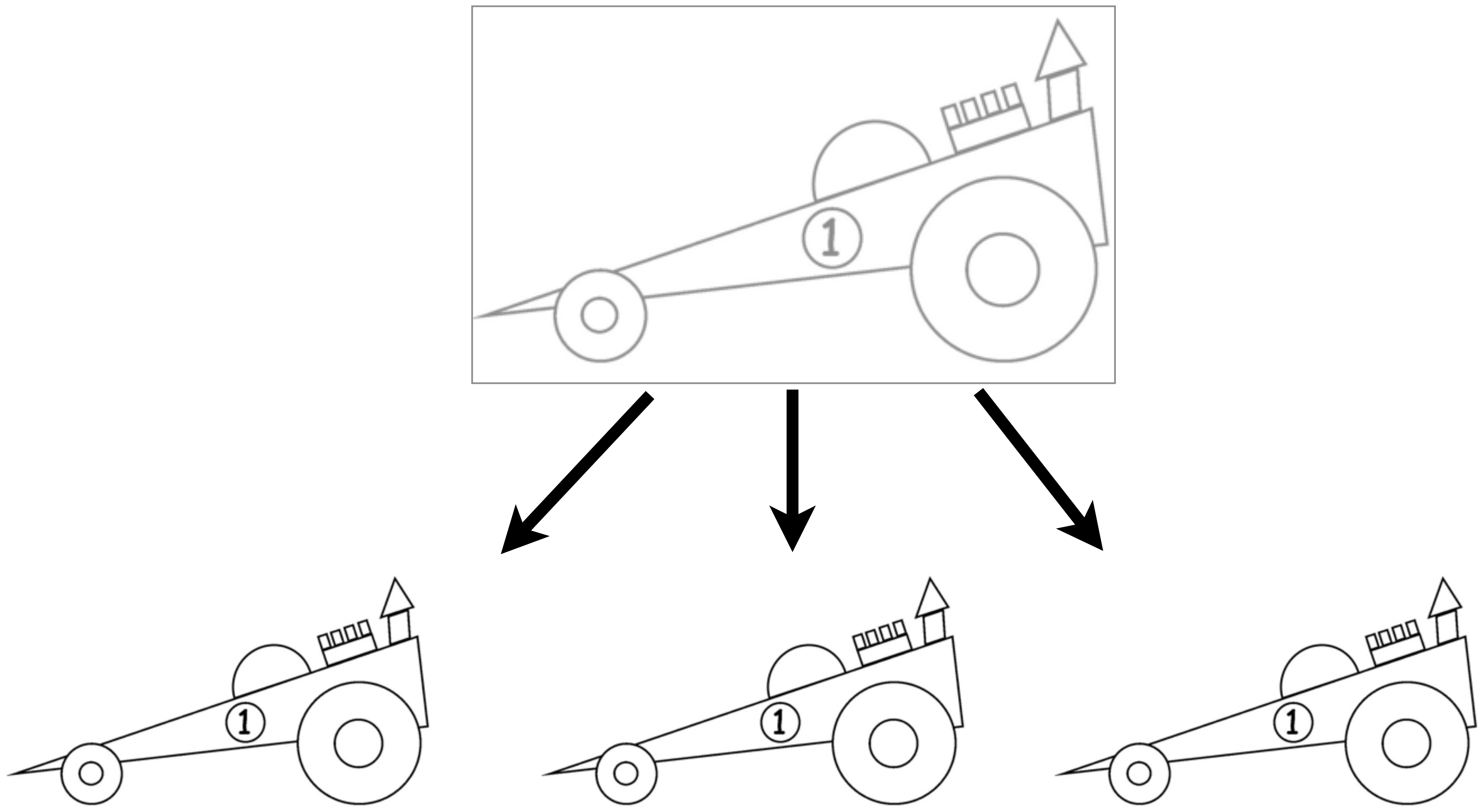
Classes (like a blueprint)



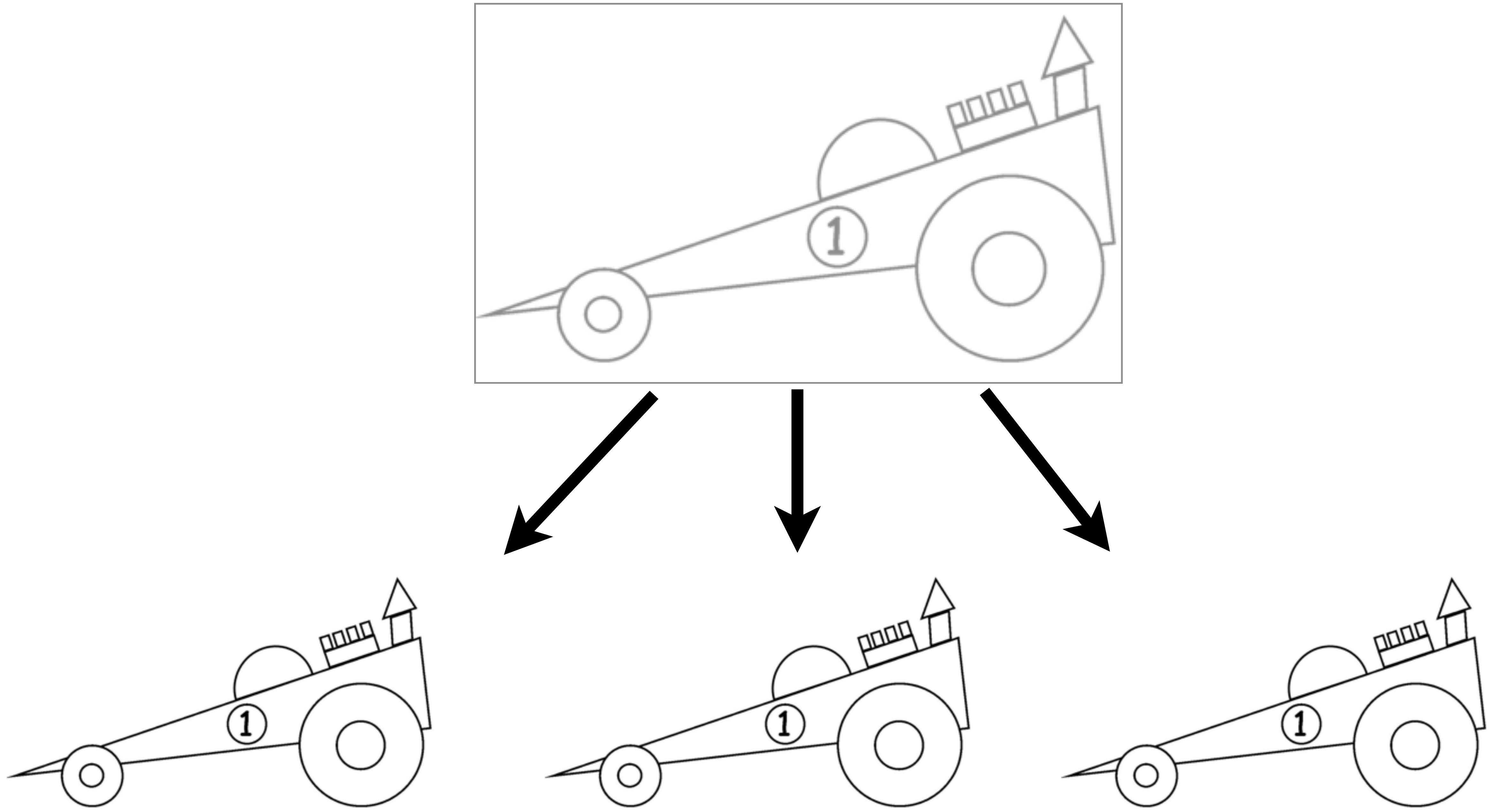
Classes (like a blueprint)



Classes (like a blueprint)

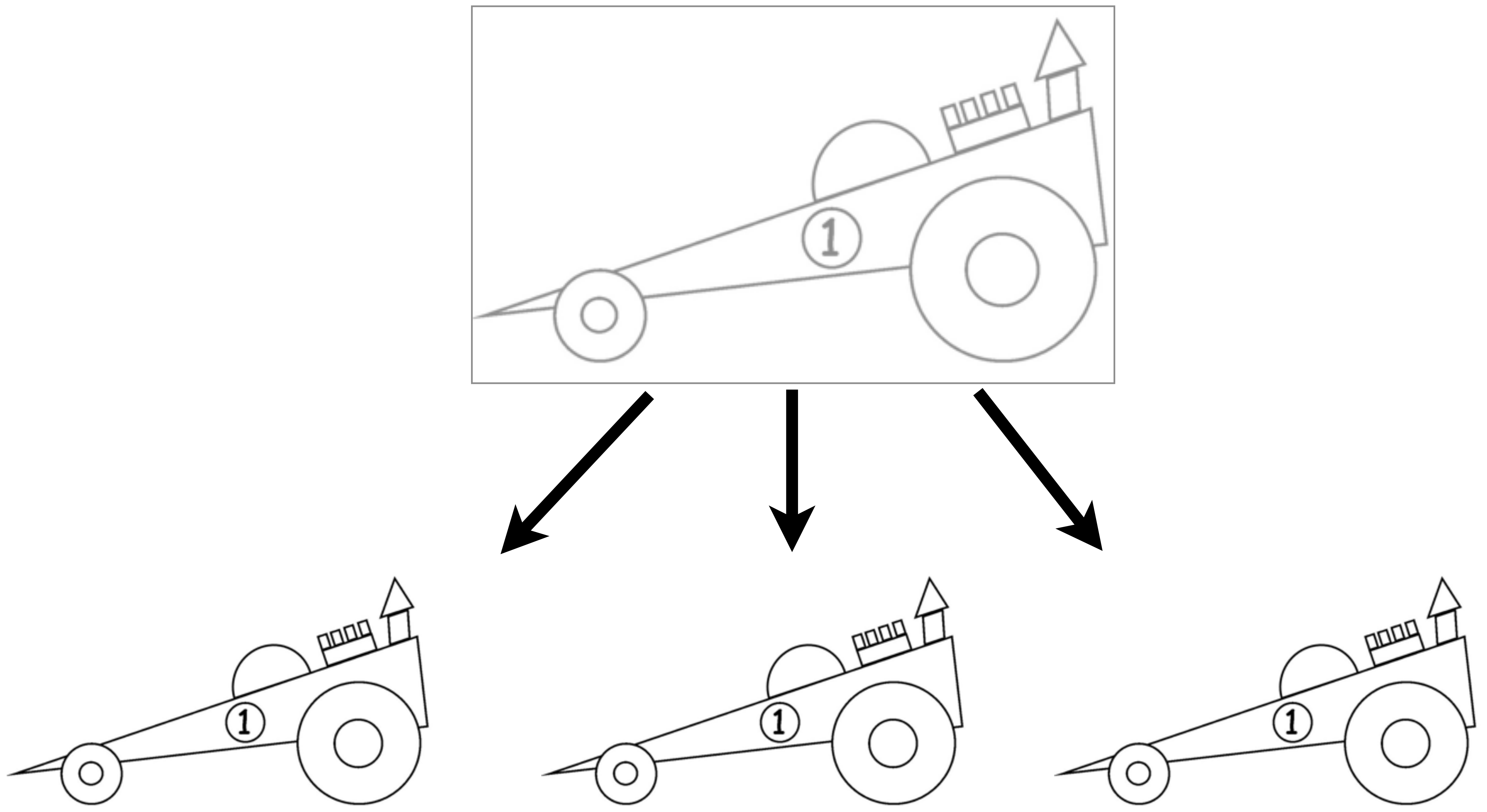


Classes (like a blueprint)



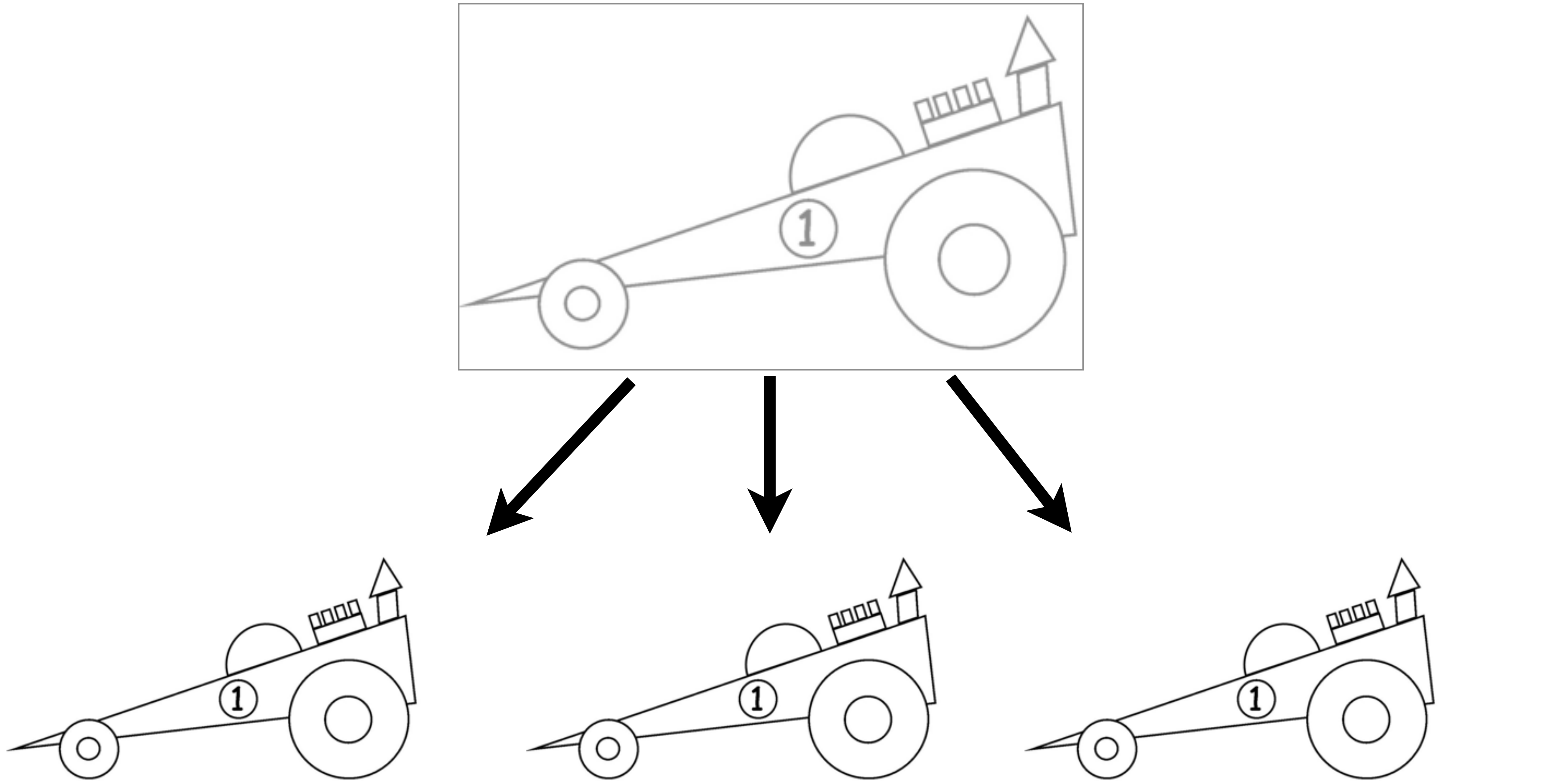
```
jrueCar.color = blue;
```

Classes (like a blueprint)



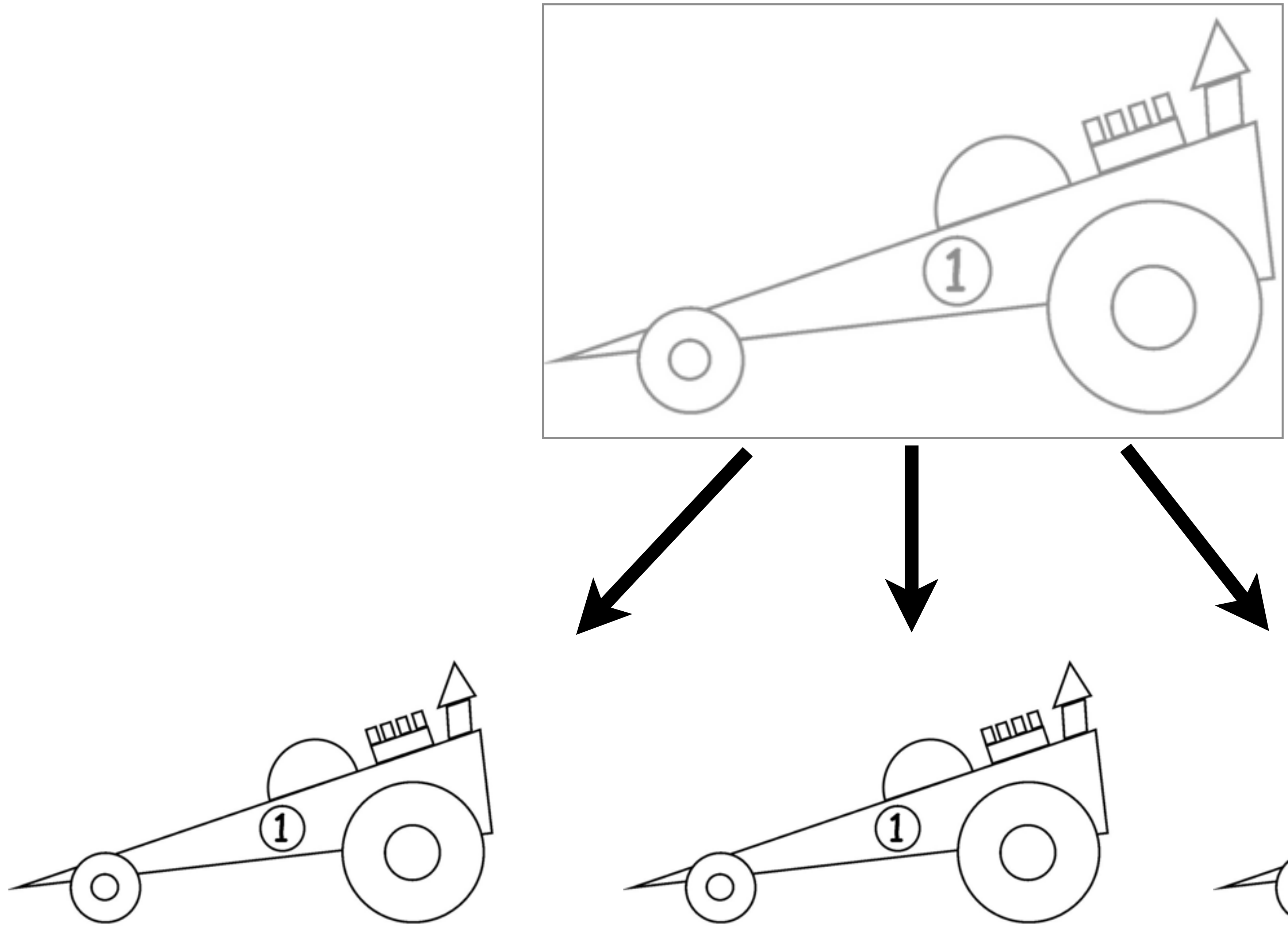
```
jrueCar.color = blue;     kociCar.color = red;
```

Classes (like a blueprint)



```
jrueCar.color = blue;      kociCar.color = red;      grabsCar.color = black;
```

Classes (like a blueprint)

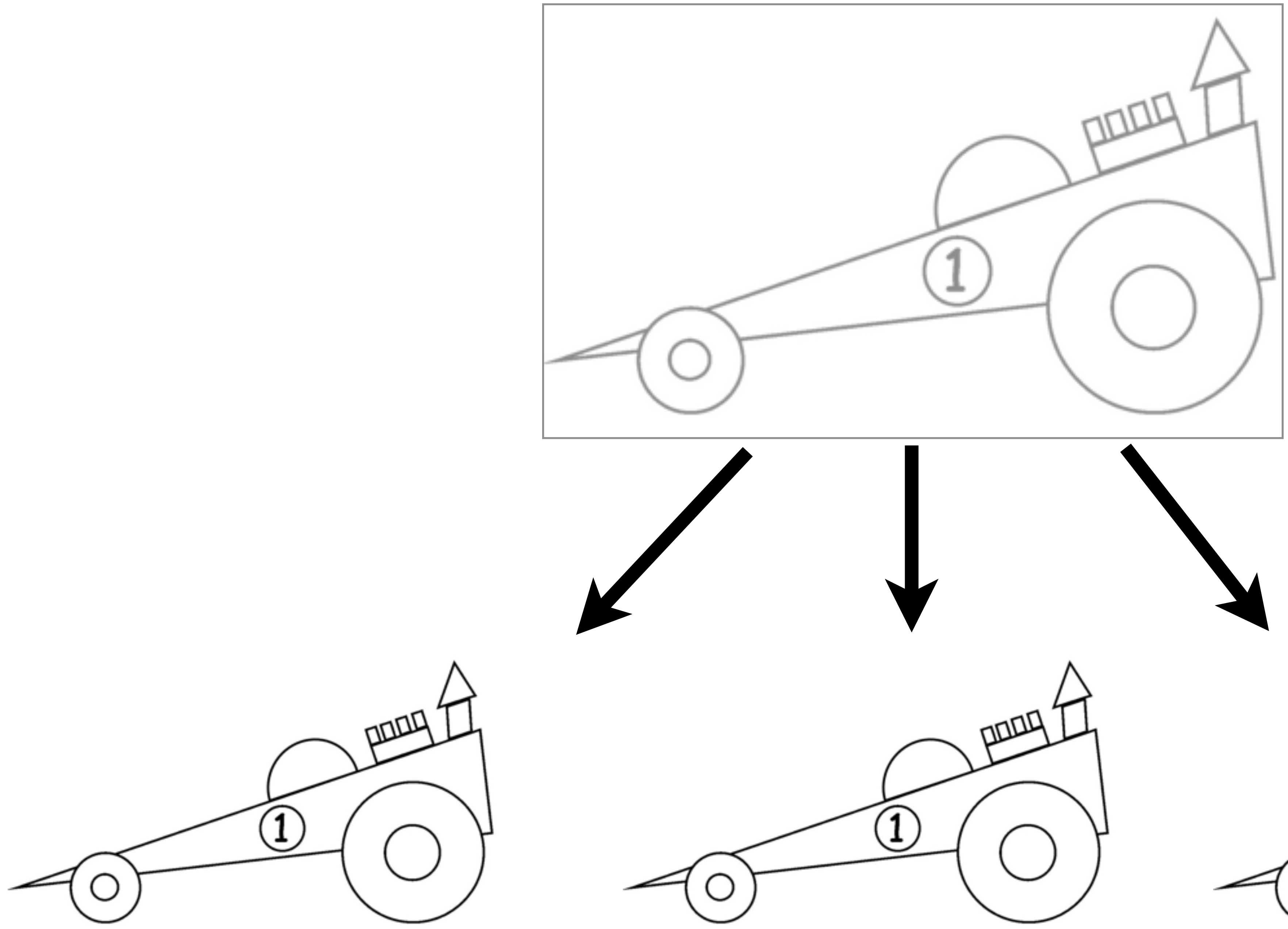


```
jrueCar.color = blue;
jrueCar.start();
```

```
kociCar.color = red;
```

```
grabsCar.color = black;
```

Classes (like a blueprint)

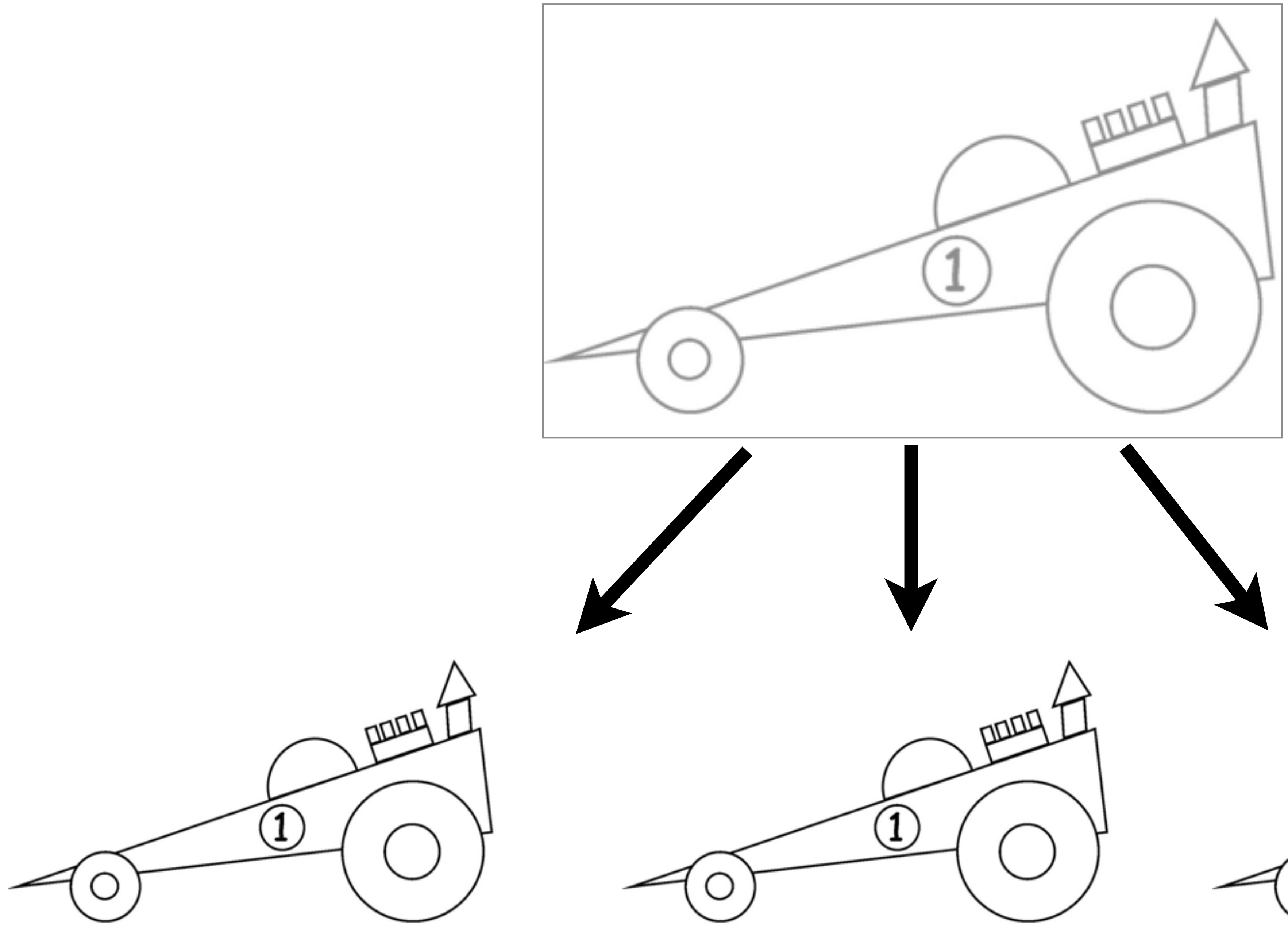


```
jrueCar.color = blue;
jrueCar.start();
```

```
kociCar.color = red;
kociCar.stop();
```

```
grabsCar.color = black;
```

Classes (like a blueprint)



```
jrueCar.color = blue;
jrueCar.start();
```

```
kociCar.color = red;
kociCar.stop();
```

```
grabsCar.color = black;
grabsCar.breakDown();
```

How does it look in code?

```
Car();
```

How does it look in code?

```
Car();
```

*Step 1: **Instantiate** your object from the class.*

```
var jrueCar = new Car();
```

How does it look in code?

```
Car();
```

*Step 1: **Instantiate** your object from the class.*

```
var jrueCar = new Car();
```

*Step 2: Set **properties** or run **methods***

```
jrueCar.color = "blue";
jrueCar.type = "sedan";
jrueCar.go(65);
```

Instances of the class

```
var jruesCar = new Car();
var kocisCar = new Car();
var grabsCar = new Car();


grabsCar.color = "black";
kocisCar.go(100);
jruesCar.playMusic();
```

Instances of the class

```
var jruesCar = new Car();
var kocisCar = new Car();
var grabsCar = new Car();
```

```
grabsCar.color = "black";
kocisCar.go(100);
jruesCar.playMusic();
```

Audio Player Code

Audio Player Code

```
var introSound = new AudioPlayer();
```

Audio Player Code

```
var introSound = new AudioPlayer();

introSound.song = "let_it_be.mp3";
```

Audio Player Code

```
var introSound = new AudioPlayer();

introSound.song = "let_it_be.mp3";

introSound.play();
```

Audio Player Code

```
var introSound = new AudioPlayer();

introSound.song = "let_it_be.mp3";

introSound.play();

var bgSound = new AudioPlayer();
```

Audio Player Code

```
var introSound = new AudioPlayer();

introSound.song = "let_it_be.mp3";

introSound.play();

var bgSound = new AudioPlayer();

bgSound.song = "hey_jude.mp3";
```

Audio Player Code

```
var introSound = new AudioPlayer();

introSound.song = "let_it_be.mp3";

introSound.play();

var bgSound = new AudioPlayer();

bgSound.song = "hey_jude.mp3";

introSound.stop();
```

Audio Player Code

```
var introSound = new AudioPlayer();

introSound.song = "let_it_be.mp3";

introSound.play();

var bgSound = new AudioPlayer();

bgSound.song = "hey_jude.mp3";

introSound.stop();

bgSound.play();
```

Prototyping (extending object)

Prototyping (extending object)

```
var intro = new AudioPlayer();
```

Prototyping (extending object)

```
var intro = new AudioPlayer();

intro.song = "help.mp3";
```

Prototyping (extending object)

```
var intro = new AudioPlayer();

intro.song = "help.mp3";

intro.play();
```

Prototyping (extending object)

```
var intro = new AudioPlayer();

intro.song = "help.mp3";

intro.play();

AudioPlayer.prototype.pause = function()
{
    //pause code
}
```

Prototyping (extending object)

```
var intro = new AudioPlayer();

intro.song = "help.mp3";

intro.play();

AudioPlayer.prototype.pause = function()
{
   //pause code
}

intro.pause();
```

Polymorphism (inheritance)

Polymorphism (inheritance)

```
var AudioPlayer = new Audio();

AudioPlayer.prototype.play = function()
{


}
```

Polymorphism (inheritance)

Polymorphism (inheritance)

```
var myname = "Jeremy";


console.log( myname.length );
```

Polymorphism (inheritance)

```
var myname = "Jeremy";


console.log( myname.length );
```

Polymorphism (inheritance)

$$\begin{tikzcd}
\text{Superclasses (built into language)} \arrow[d] \\
\text{subclasses} \arrow[d] \\
\text{extended classes} \arrow[d] \\
\text{your code}
\end{tikzcd}$$

Every variable in JavaScript is an Object

```
var fruit = new Array("Pear", "Apple");
```

```
fruit.push("Banana");
```

fruit is now three items, Pear, Apple, Banana

```
fruit.pop();
```

fruit is now two items, Pear, Apple.

Every variable in JavaScript is an Object

```
var myname = new String("Jeremy");
```

```
myname.toUpperCase();
```

myname variable is now JEREMY

```
myname.trim();
```

myname has trimmed the whitespace from beginning and end of the string

If someone asks you:

JavaScript is a *prototypical* language

While JavaScript is technically object oriented, it's a prototypical language which means objects (instances) can be extended through a built-in prototype method every object has.

Polymorphism

JavaScript has the ability to create objects which inherit characteristics from superclasses.

Duck Typing

"When I see a bird that walks like a duck and swims like a duck and quacks like a duck, I call that bird a duck."

- James Whitcomb Riley

Objects can inherit all of the properties and methods from another class without explicitly instantiating it from that class.