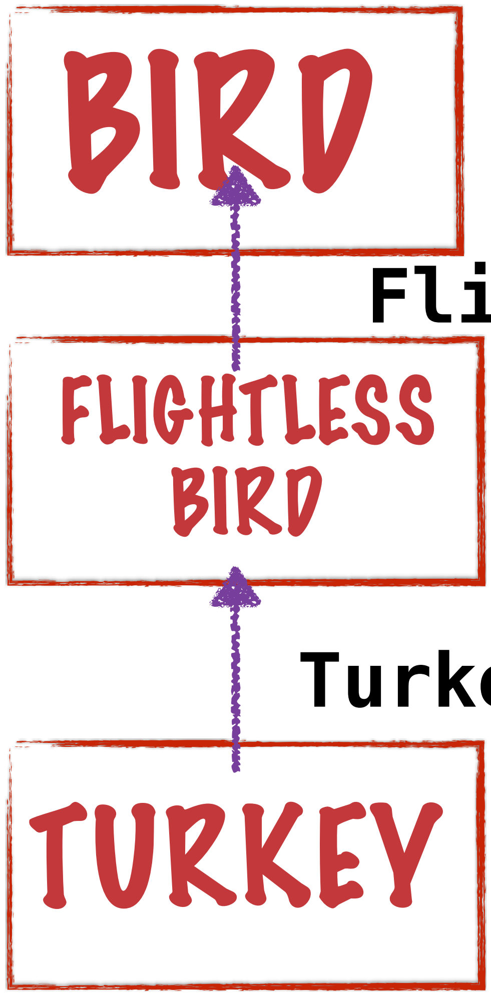
EXAMPLE 48: CREATING AN INHERITANCE HIERARCHY USING CHAINED PROTOTYPES

AN INHERITANCE HIERARCHY USING CHAINED PROTOTYPES

3 OBJECT CONSTRUCTO



BIRD

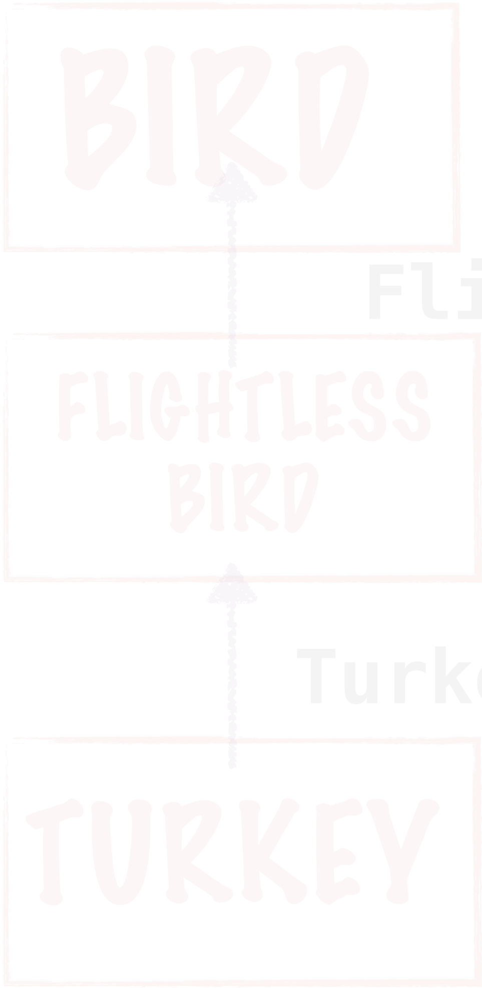
FLIGHTLESS BIRD

TURKEY

```
FlightlessBird.prototype = new Bird();
```

```
Turkey.prototype = new FlightlessBird()
```

3 OBJECT CONSTRUCTO



```
function Bird() {
  this.sayHello = function() {
    console.log("I am a bird, of type = " +
this.name);
  }
  this.fly = function() {
    console.log("I am flying!!");
  }
}
```

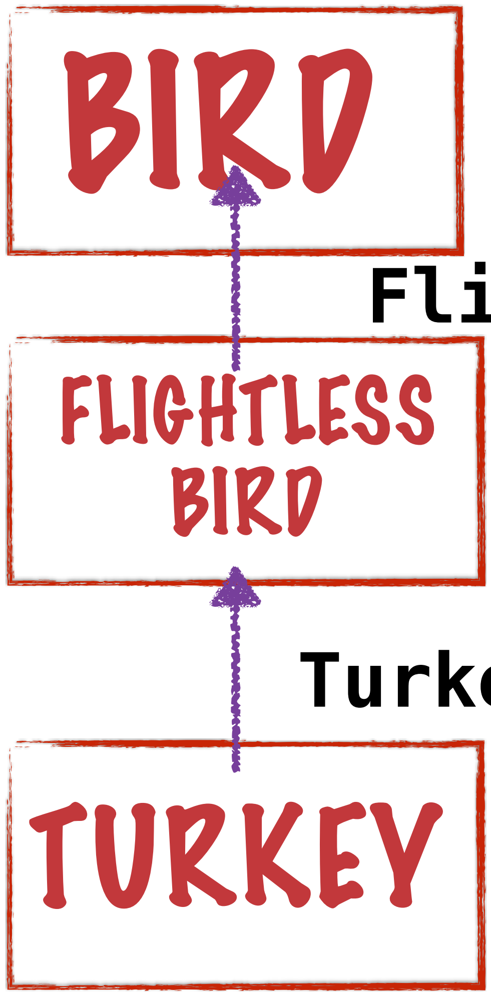
```
FlightlessBird.prototype = new Bird();
```

```
function FlightlessBird() {
 this.fly = function() {
    console.log("I can't fly, even though
   }
}
```

```
Turkey.prototype = new FlightlessBird()
```

```
function Turkey(){
  this.name = "Turkey";
}
```

ANY PROPERTY CALLED ON `Turkey` WILL FIRST LOOK
INSIDE `Turkey`, THEN INSIDE `FlightlessBird`,
AND THEN IN `Bird` (IN THAT ORDER)



BIRD

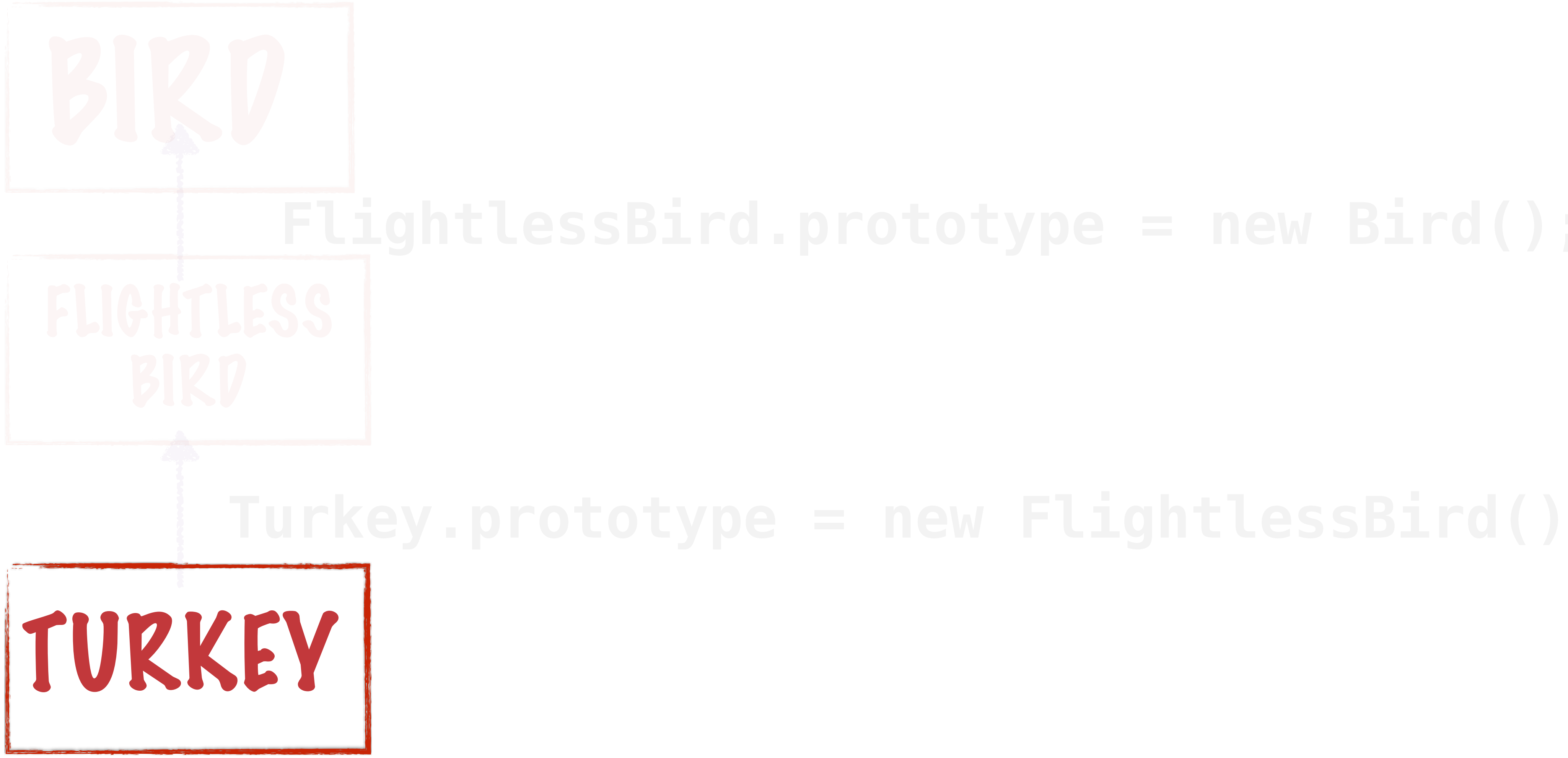
FLIGHTLESS BIRD

TURKEY

```
FlightlessBird.prototype = new Bird();
```

```
Turkey.prototype = new FlightlessBird()
```

ANY PROPERTY CALLED ON Turkey WILL FIRST LOOK INSIDE Turkey, THEN INSIDE FlightlessBird, AND THEN IN Bird (IN THAT ORDER)



BIRD

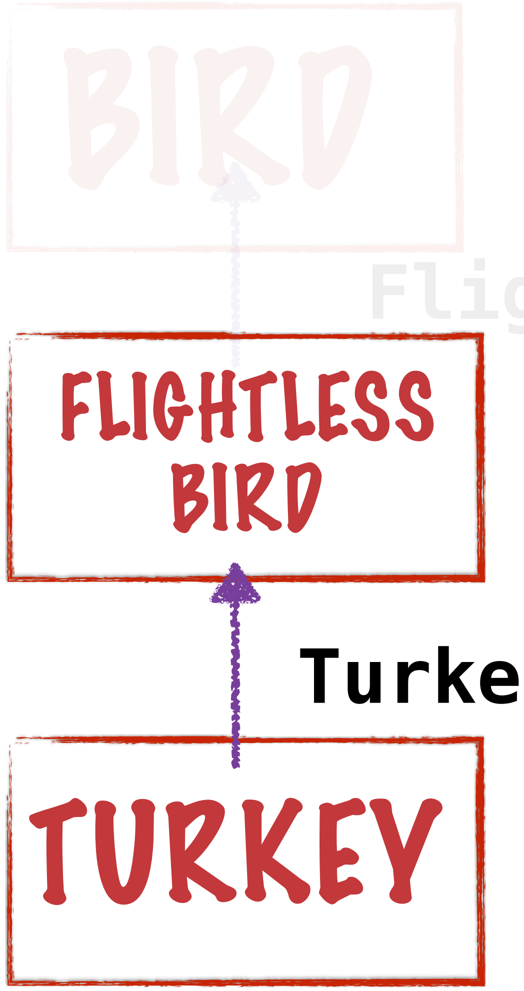
```
FlightlessBird.prototype = new Bird();
```

FLIGHTLESS BIRD

```
Turkey.prototype = new FlightlessBird()
```

TURKEY

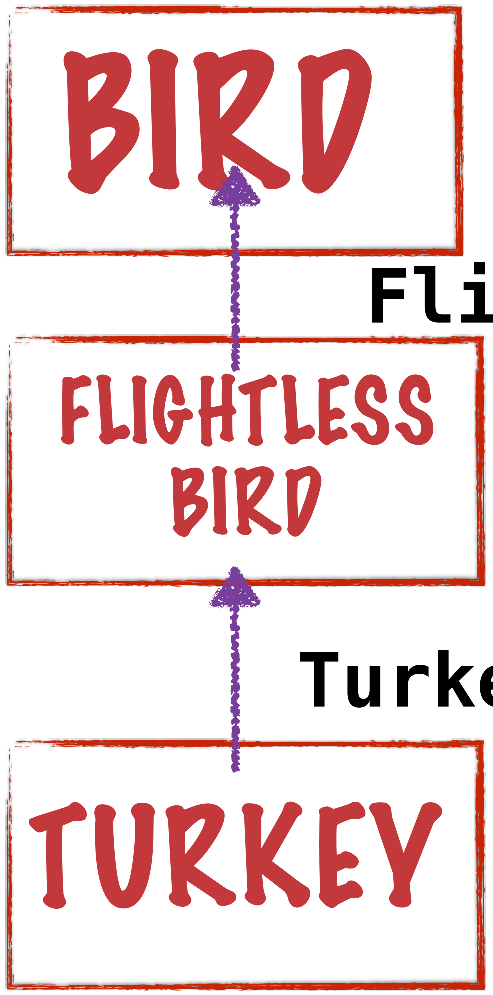
ANY PROPERTY CALLED ON Turkey WILL FIRST LOOK INSIDE Turkey, THEN INSIDE FlightlessBird, AND THEN IN Bird (IN THAT ORDER)



```
FlightlessBird.prototype = new Bird();
```

```
Turkey.prototype = new FlightlessBird()
```

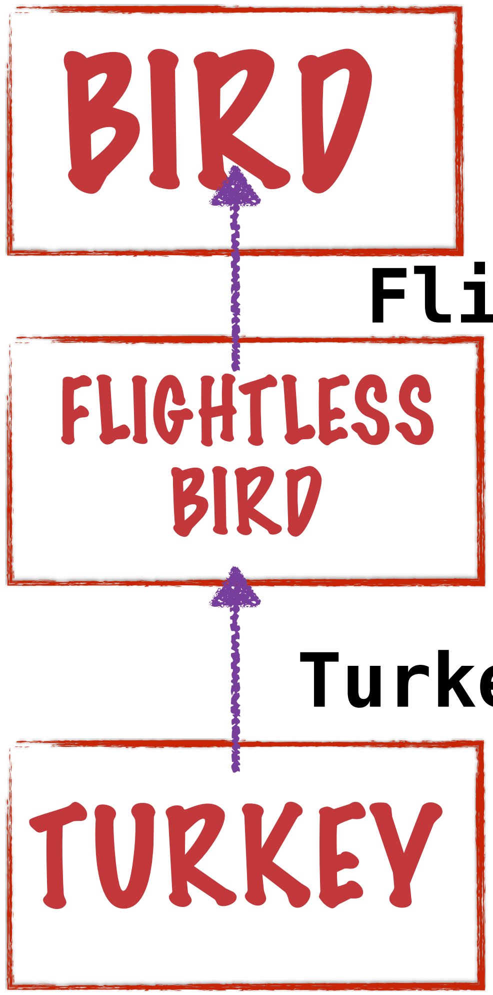
ANY PROPERTY CALLED ON `Turkey` WILL FIRST LOOK INSIDE `Turkey`, THEN INSIDE `FlightlessBird`, AND THEN IN `Bird` (IN THAT ORDER)



```
FlightlessBird.prototype = new Bird();
```

```
Turkey.prototype = new FlightlessBird()
```

ANY PROPERTY CALLED ON `Turkey` WILL FIRST LOOK INSIDE `Turkey`, THEN INSIDE `FlightlessBird`, AND THEN IN `Bird` (IN THAT ORDER)



BIRD

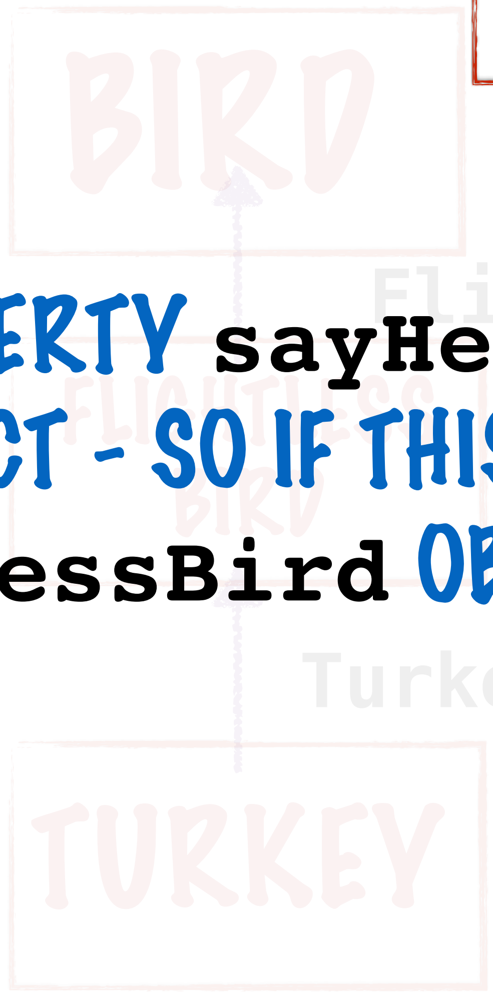
FLIGHTLESS BIRD

TURKEY

```
FlightlessBird.prototype = new Bird();
```

```
Turkey.prototype = new FlightlessBird()
```

3 OBJECT CONSTRUCTO



```
function Bird() {
  this.sayHello = function() {
    console.log("I am a bird, of type = " +
this.name);
  }
  this.fly = function() {
    console.log("I am flying!!");
  }
}
```

THE PROPERTY `sayHello` APPEARS ONLY IN THE `Bird` OBJECT - SO IF THIS IS CALLED ON A `Turkey` OR `FlightlessBird` OBECT, THIS VERSION WILL BE CALLED.

```
FlightlessBird.prototype = new Bird();
function FlightlessBird(){
  this.fly = function() {
    console.log(
  }
}
Turkey.prototype = new FlightlessBird()

  function Turkey(){
    this.name =
  }
```

3 OBJECT CONSTRUCTO

BIRD
FLIGHTLESS BIRD
TURKEY

```
function Bird() {
  this.sayHello = function() {
    console.log("I am a bird, of type = " +
this.name);
  }
  this.fly = function() {
    console.log("I am flying!!");
  }
}
FlightlessBird.prototype = new Bird();
function FlightlessBird() {
this.fly = function() {
   console.log("I can't fly, even though I'
 }
}
Turkey.prototype = new FlightlessBird()
  function Turkey(){
    this.name =
  }
```

THE PROPERTY `fly` APPEARS BOTH IN THE `Bird` OBJECT AND IN THE `FlightlessBird` OBECT - SO IF THIS IS CALLED ON A `Turkey` OR, THE `FlightlessBird` VERSION WILL BE CALLED.