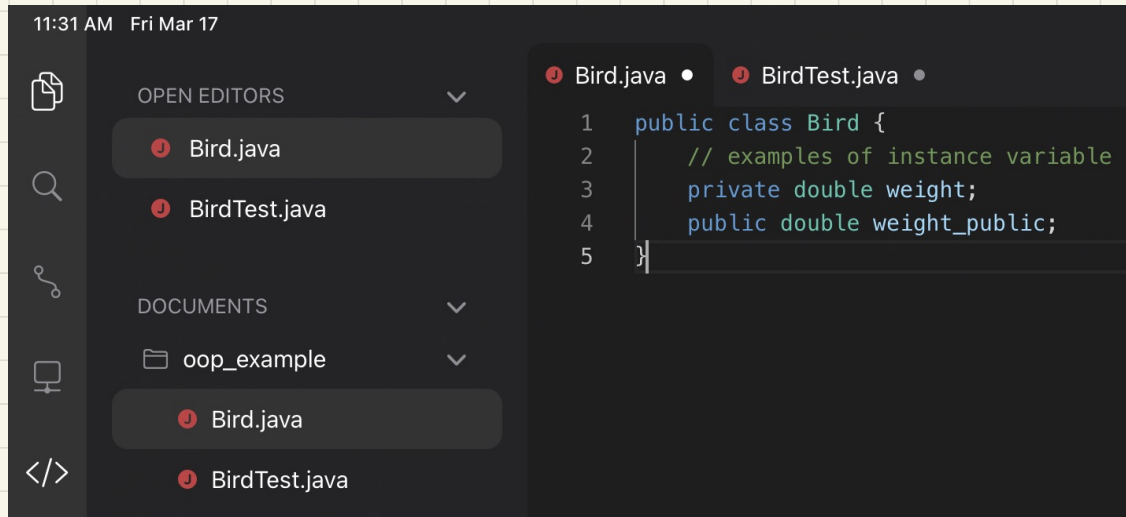



How to create a class?

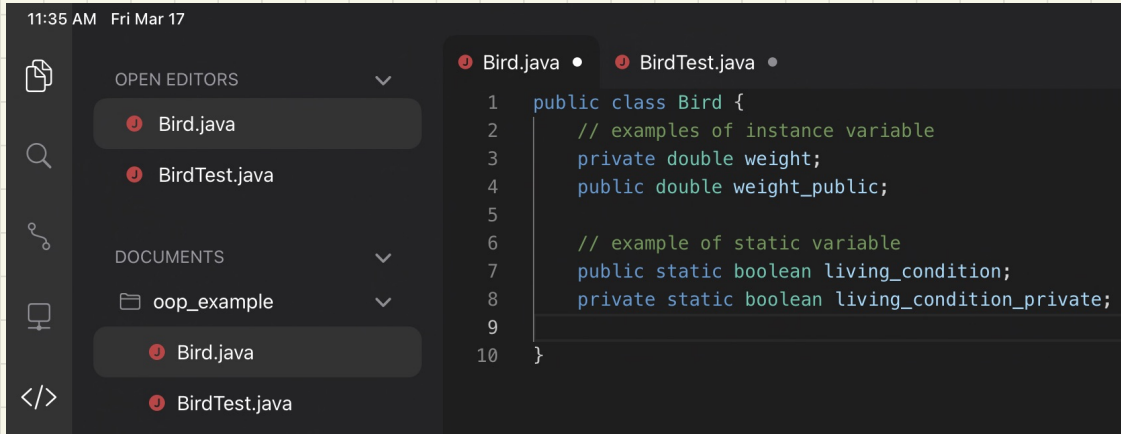
1. Define instance variable



The screenshot shows an IDE interface with a dark theme. On the left, the Explorer sidebar displays the project structure: 'Bird.java' and 'BirdTest.java' under 'OPEN EDITORS', and 'oop_example' under 'DOCUMENTS'. The main editor area shows the 'Bird.java' file with the following code:

```
1 public class Bird {  
2     // examples of instance variable  
3     private double weight;  
4     public double weight_public;  
5 }
```

2. Define static variable



The screenshot shows an IDE interface with a dark theme. The top status bar displays "11:35 AM Fri Mar 17". The left sidebar contains icons for Explorer, Search, Source Control, and Run and Debug. The Explorer view shows a project named "oop_example" containing two files: "Bird.java" and "BirdTest.java". The main editor area displays the code for "Bird.java". The code defines a public class "Bird" with two instance variables: "private double weight;" and "public double weight_public;". It also includes a static variable: "public static boolean living_condition;". The code is as follows:

```
1 public class Bird {
2     // examples of instance variable
3     private double weight;
4     public double weight_public;
5
6     // example of static variable
7     public static boolean living_condition;
8     private static boolean living_condition_private;
9
10 }
```

3. Create constructor

```
Bird.java • BirdTest.java •
1  public class Bird {
2      // examples of instance variable
3      private double weight;
4      public double weight_public;
5
6      // example of static variable
7      public static boolean living_condition;
8      private static boolean living_condition_private;
9
10     // create constructor that used to initialize instance variable
11     // constructor allow you to create an object
12     public Bird(double initial_weight) {
13
14     }
15 }
```

Constructor is a method that has the same name as the class and no return type. It is used to initialize instance variables.

When you create an object, you will need to call the constructor in the client program.

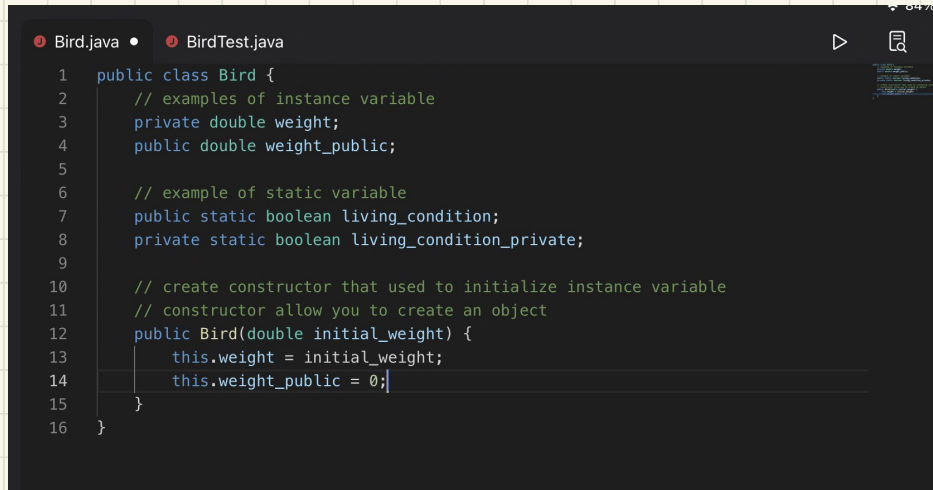
With a constructor, you can create an object

A screenshot of a code editor window with a dark theme. The window has two tabs: 'Bird.java' and 'BirdTest.java', with 'BirdTest.java' being the active tab. The code in 'BirdTest.java' is as follows:

```
1 public class BirdTest {  
2  
3     public static void main(String args[]) {  
4  
5         Bird little_bird = new Bird(10);  
6     }  
7 }
```

Line numbers 1 through 7 are visible on the left side of the editor. The code is color-coded: 'public' is blue, 'class' is green, 'void' is blue, 'main' is green, 'String' is blue, 'args' is blue, 'new' is purple, and 'Bird' is green. The 'Bird' class is not defined in this file. In the top right corner of the editor, there are icons for running (a play button), searching (a magnifying glass), and a menu (three dots). A status bar at the very top right shows '93%' and a battery icon.

4. Initialize instance variable inside the constructor



```
1 public class Bird {
2     // examples of instance variable
3     private double weight;
4     public double weight_public;
5
6     // example of static variable
7     public static boolean living_condition;
8     private static boolean living_condition_private;
9
10    // create constructor that used to initialize instance variable
11    // constructor allow you to create an object
12    public Bird(double initial_weight) {
13        this.weight = initial_weight;
14        this.weight_public = 0;
15    }
16 }
```

5. Write some instance method

```
public double getWeight() {  
    return this.weight;  
}  
  
public void eat(double amount) {  
    this.weight += amount;  
}  
  
private double private_get_weight() {  
    return this.weight;  
}
```

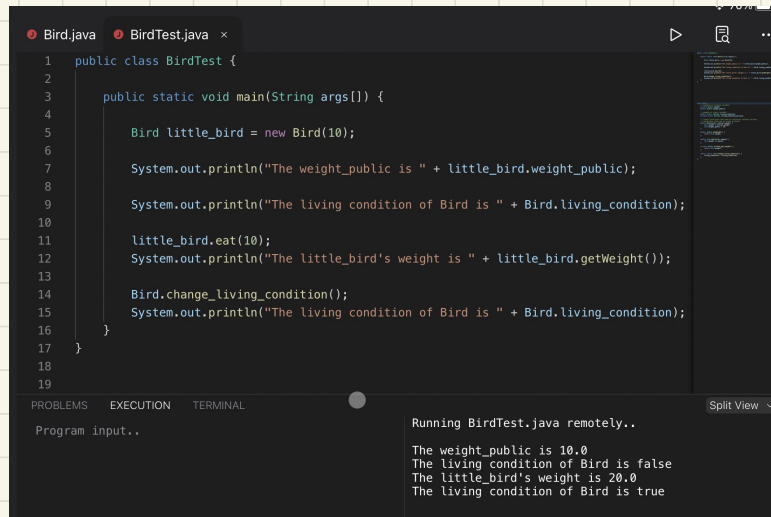
What is mutator method and accessor method

6. Write some static method

```
public static void change_living_condition() {  
    living_condition = !living_condition;  
}
```


The complete Bird class and BirdTest class

```
public class Bird {  
    // examples of instance variable  
    private double weight;  
    public double weight_public;  
  
    // example of static variable  
    public static boolean living_condition;  
    private static boolean living_condition_private;  
  
    // create constructor that used to initialize instance variable  
    // constructor allow you to create an object  
    public Bird(double initial_weight) {  
        this.weight = initial_weight;  
        this.weight_public = 10;  
    }  
  
    public double getWeight() {  
        return this.weight;  
    }  
  
    public void eat(double amount) {  
        this.weight += amount;  
    }  
  
    private double private_get_weight() {  
        return this.weight;  
    }  
  
    public static void change_living_condition() {  
        living_condition = !living_condition;  
    }  
}
```



The screenshot shows an IDE with two tabs: Bird.java and BirdTest.java. The BirdTest.java file contains the following code:

```
1 public class BirdTest {  
2  
3     public static void main(String args[]) {  
4  
5         Bird little_bird = new Bird(10);  
6  
7         System.out.println("The weight_public is " + little_bird.weight_public);  
8  
9         System.out.println("The living condition of Bird is " + Bird.living_condition);  
10  
11         little_bird.eat(10);  
12         System.out.println("The little_bird's weight is " + little_bird.getWeight());  
13  
14         Bird.change_living_condition();  
15         System.out.println("The living condition of Bird is " + Bird.living_condition);  
16     }  
17 }  
18  
19
```

The IDE's terminal window shows the output of the program:

```
Running BirdTest.java remotely..  
  
The weight_public is 10.0  
The living condition of Bird is false  
The little_bird's weight is 20.0  
The living condition of Bird is true
```

You should know the following:

1. Call public instance variable inside class: `this.var` or `var`
2. Call public instance variable in client program: `object.var`
3. Call private instance variable inside class: `this.var` or `var`
4. Call private instance variable in client program: `not allowed`
5. Call public static variable inside class: `var`
6. Call public static variable in client program: `Class.var`
7. Call private static variable inside class: `var`
8. Call private static variable in client program: `not allowed`
9. Call public instance method inside class: `this.method()` or `method()`
10. Call public instance method in client program: `object.method()`
11. Call private instance method inside class: `this.method()` or `method()`
12. Call private instance method in client program: `not allowed`
13. Call public static method inside class: `method()`
14. Call public static method in client program: `Class.method()`
15. Call private static method inside class: `method()`
16. Call private static method in client program: `not allowed`

You cannot only use static method/variable inside static method

1. Example: use `this.weight_public` or `weight_public` inside the bird class
2. Example: `print(little_bird.weight_public)` in the BirdTest class
3. Example: use `this.weight` or `weight` inside the bird class
5. Example: use `living_condition` inside the bird class
6. Example: use `Bird.living_condition` in the BirdTest class
7. Example: use `living_condition_private` inside the bird class
9. Example: use `this.getWeight()` or `getWeight()` inside the bird class
10. Example: use `little_bird.eat(10)` in BirdTest class
11. Example: use `this.private_get_weight()` or `private_get_weight()` inside the bird class
13. Example: use `change_living_condition()` inside the the bird class
14. Example: use `Bird.change_living_condition()` in the BirdTest class
15. We seldom use private static method

Quick demo of writing movingparticle

Method overload and constructor overload:

You can have two method/constructor with the same name but different parameter type or number of parameter

Passing reference type to method:

You make a copy of the reference when it passes to the method. You can change the instance variable of the object, but you cannot let the passed in reference refers to another variable

