

ООП — основные принципы

Объектно-ориентированное программирование (ООП)

ООП – это программирование с помощью классов и объектов.

Давайте сначала разберемся что такое объект. А потом плавно перейдем к такому понятию как класс.

Всё вокруг нас является
объектом.



У объекта есть
свойства
(еще называют параметры)



У объекта есть
методы
(методы – это действия,
то есть что может
делать объект)

Например, машина – это объект.



У любой машины есть такие
свойства: модель, цвет, размер и т.д.

Методы машины:
затормозить ();
нажатьНаГаз ();
открытьДверь ();
закрытьДверь ();
и т.д.

Например, котёнок – это объект.

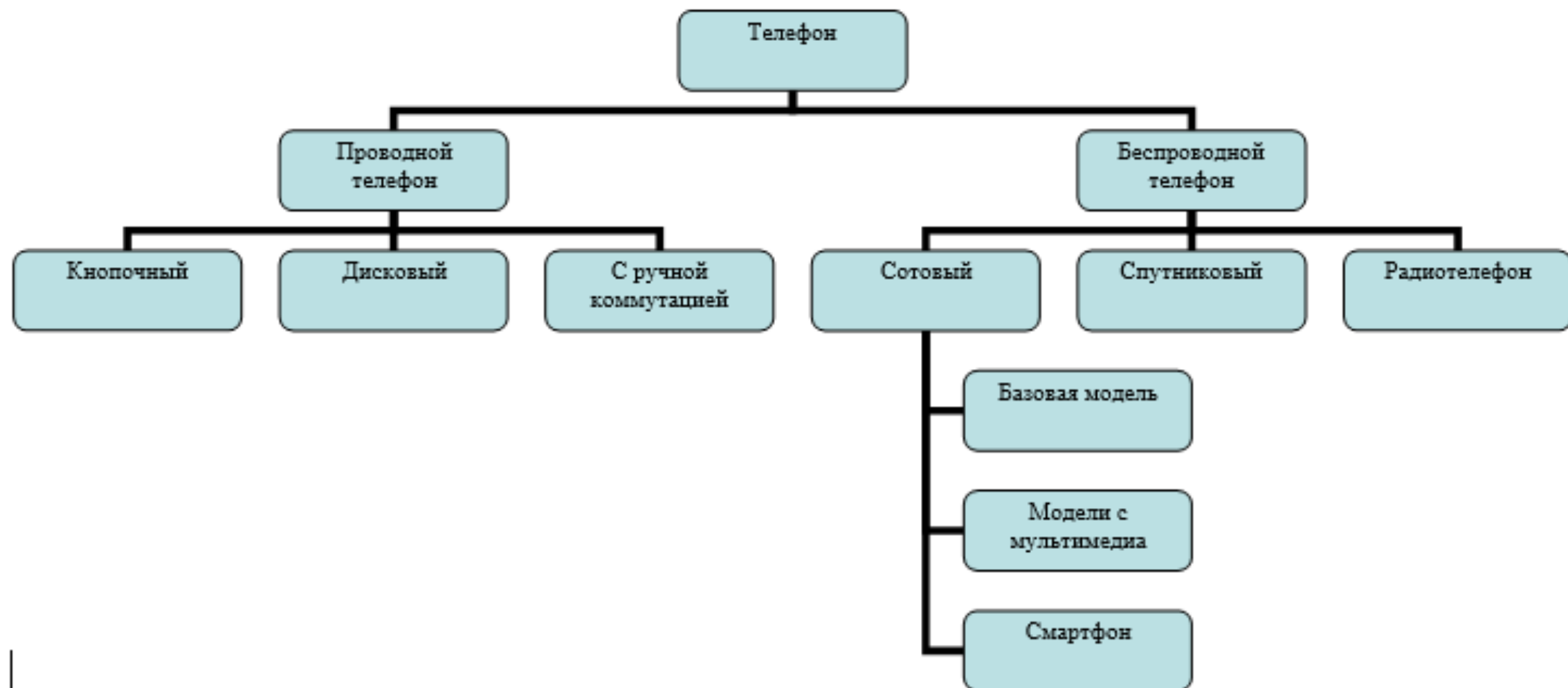


У любого котенка есть свойства:
порода, имя, цвет, длина шерсти,
возраст и т.д.

Методы котенка:
спать();
кушать();
играть ();
шкодить ();
и т.д.

Абстракция

Абстракция

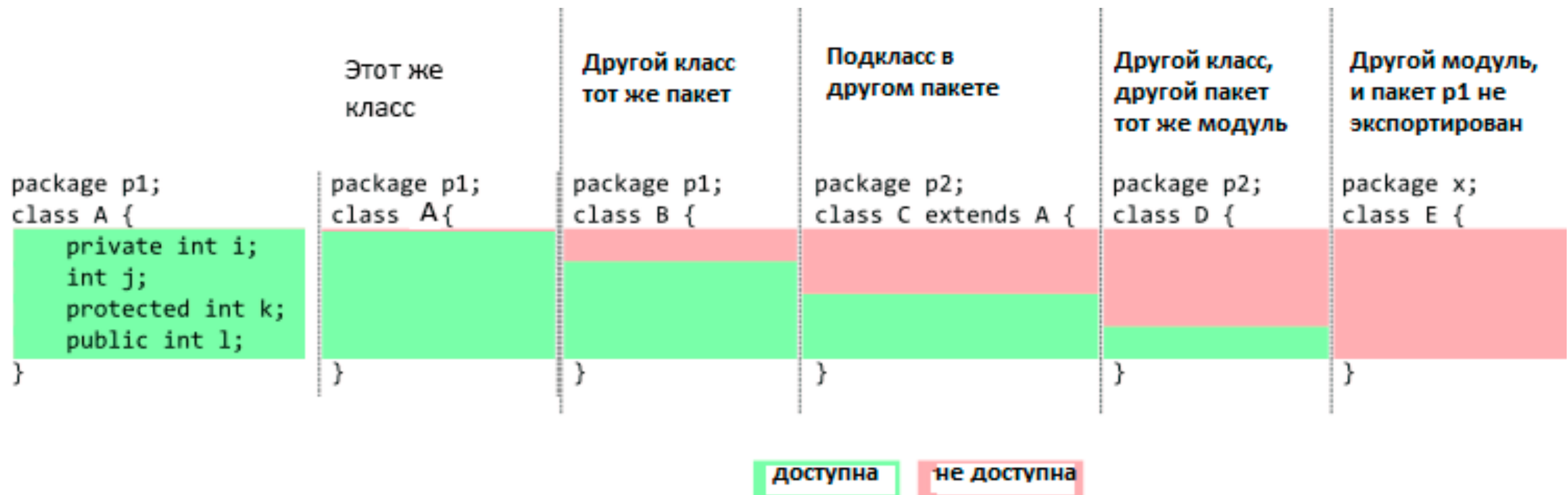


Абстракция

```
1 public abstract class AbstractPhone {  
2     private int year;  
3  
4     public AbstractPhone(int year) {  
5         this.year = year;  
6     }  
7     public abstract void call(int outputNumber);  
8     public abstract void ring (int inputNumber);  
9 }
```

Инкапсуляция

Инкапсуляция

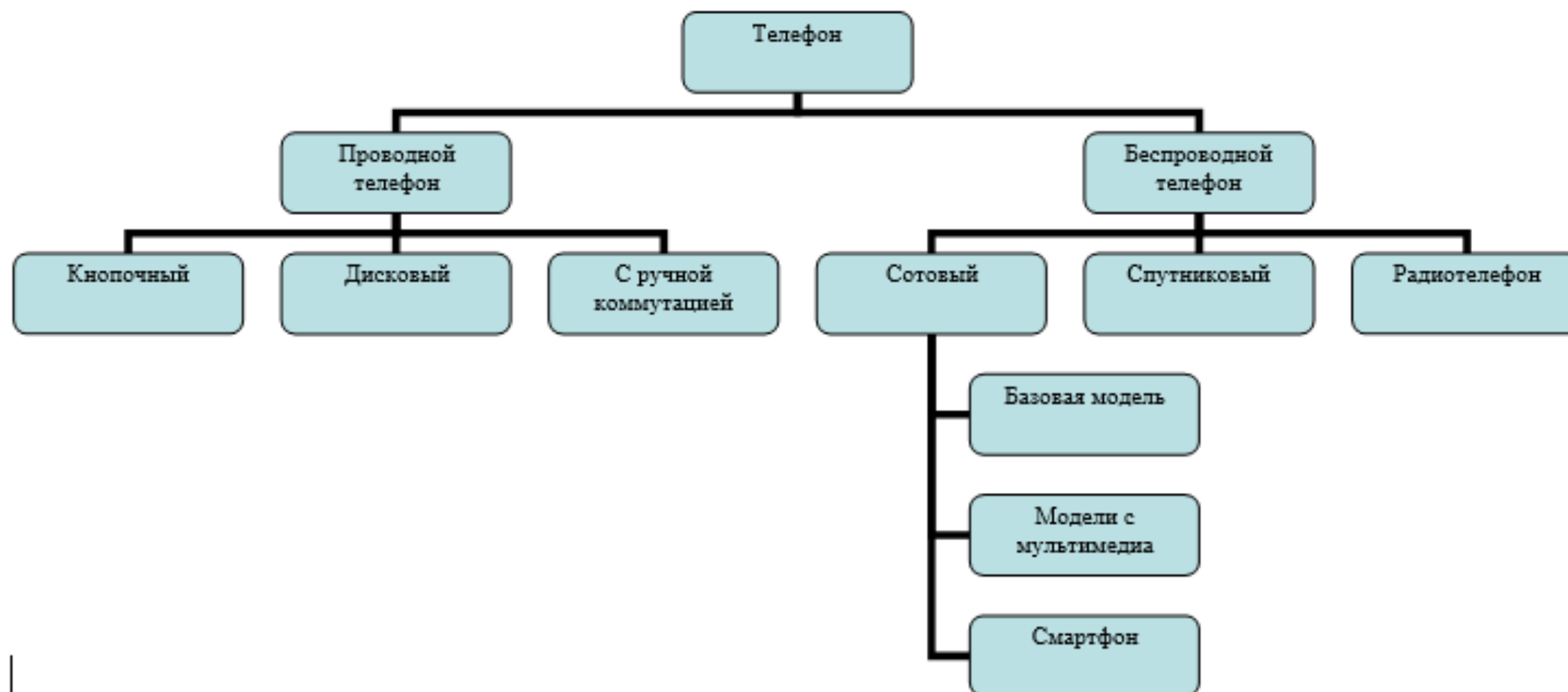


Инкапсуляция

```
1 public class SomePhone {
2
3     private int year;
4     private String company;
5     public SomePhone(int year, String company) {
6         this.year = year;
7         this.company = company;
8     }
9     private void openConnection(){
10        //findComutator
11        //openNewConnection...
12    }
13    public void call() {
14        openConnection();
15        System.out.println("Вызываю номер");
16    }
17
18    public void ring() {
19        System.out.println("Дзынь-дзынь");
20    }
21
22 }
```


Наследование

Наследование



Наследование

```
1 public abstract class WirelessPhone extends AbstractPhone {  
2  
3     private int hour;  
4  
5     public WirelessPhone(int year, int hour) {  
6         super(year);  
7         this.hour = hour;  
8     }  
9 }
```

Наследование

```
1 public class CellPhone extends WirelessPhone {  
2     public CellPhone(int year, int hour) {  
3         super(year, hour);  
4     }  
5  
6     @Override  
7     public void call(int outputNumber) {  
8         System.out.println("Вызываю номер " + outputNumber);  
9     }  
10  
11    @Override  
12    public void ring(int inputNumber) {  
13        System.out.println("Вам звонит абонент " + inputNumber);  
14    }  
15 }
```

Наследование

```
1 public class Smartphone extends CellPhone {  
2  
3     private String operationSystem;  
4  
5     public Smartphone(int year, int hour, String operationSystem) {  
6         super(year, hour);  
7         this.operationSystem = operationSystem;  
8     }  
9     public void install(String program){  
10         System.out.println("Устанавливаю " + program + " для " + operationSystem);  
11     }  
12  
13 }
```

Полиморфизм

Полиморфизм

```
1 public class User {
2     private String name;
3
4     public User(String name) {
5         this.name = name;
6     }
7
8     public void callAnotherUser(int number, AbstractPhone phone){
9         // вот он полиморфизм – использование в коде абстрактного типа AbstractPhone phone!
10        phone.call(number);
11    }
12 }
13 }
```

Полиморфизм

```
1 public class ThomasEdisonPhone extends AbstractPhone {
2
3 public ThomasEdisonPhone(int year) {
4     super(year);
5 }
6
7 @Override
8 public void call(int outputNumber) {
9     System.out.println("Вращайте ручку");
10    System.out.println("Сообщите номер абонента, сэр");
11 }
12
13 @Override
14 public void ring(int inputNumber) {
15     System.out.println("Телефон звонит");
16 }
```


Полиморфизм

```
1 public class Phone extends AbstractPhone {
2
3     public Phone(int year) {
4         super(year);
5     }
6
7     @Override
8     public void call(int outputNumber) {
9         System.out.println("Вызываю номер" + outputNumber);
10    }
11
12    @Override
13    public void ring(int inputNumber) {
14        System.out.println("Телефон звонит");
15    }
16 }
```

Полиморфизм

```
1 public class VideoPhone extends AbstractPhone {
2
3     public VideoPhone(int year) {
4         super(year);
5     }
6     @Override
7     public void call(int outputNumber) {
8         System.out.println("Подключаю видеоканал для абонента " + outputNumber );
9     }
10    @Override
11    public void ring(int inputNumber) {
12        System.out.println("У вас входящий видеовызов..." + inputNumber);
13    }
14 }
```

Полиморфизм

```
1 AbstractPhone firstPhone = new ThomasEdisonPhone(1879);
2 AbstractPhone phone = new Phone(1984);
3 AbstractPhone videoPhone=new VideoPhone(2018);
4 User user = new User("Андрей");
5 user.callAnotherUser(224466,firstPhone);
6 // Вращайте ручку
7 //Сообщите номер абонента, сэр
8 user.callAnotherUser(224466,phone);
9 //Вызываю номер 224466
10 user.callAnotherUser(224466,videoPhone);
11 //Подключаю видеоканал для абонента 224466
```

Перегрузка методов (Overloading)

Перегрузка методов

```
1 public class HelloWorld {  
2  
3     public static void main(String []args){  
4         HelloWorld hw = new HelloWorld();  
5         hw.say(1);  
6         hw.say("1");  
7     }  
8  
9     public static void say(Integer number) {  
10        System.out.println("Integer " + number);  
11    }  
12    public static void say(String number) {  
13        System.out.println("String " + number);  
14    }  
15 }
```

Перегрузка методов

You cannot declare more than one method with the same name and the same number and type of arguments, because the compiler cannot tell them apart.

Java Documentation : Defining Methods



Перегрузка методов

You cannot declare more than one method with the same name and the same number and type of arguments, because the compiler cannot tell them apart.

Java Documentation : Defining Methods

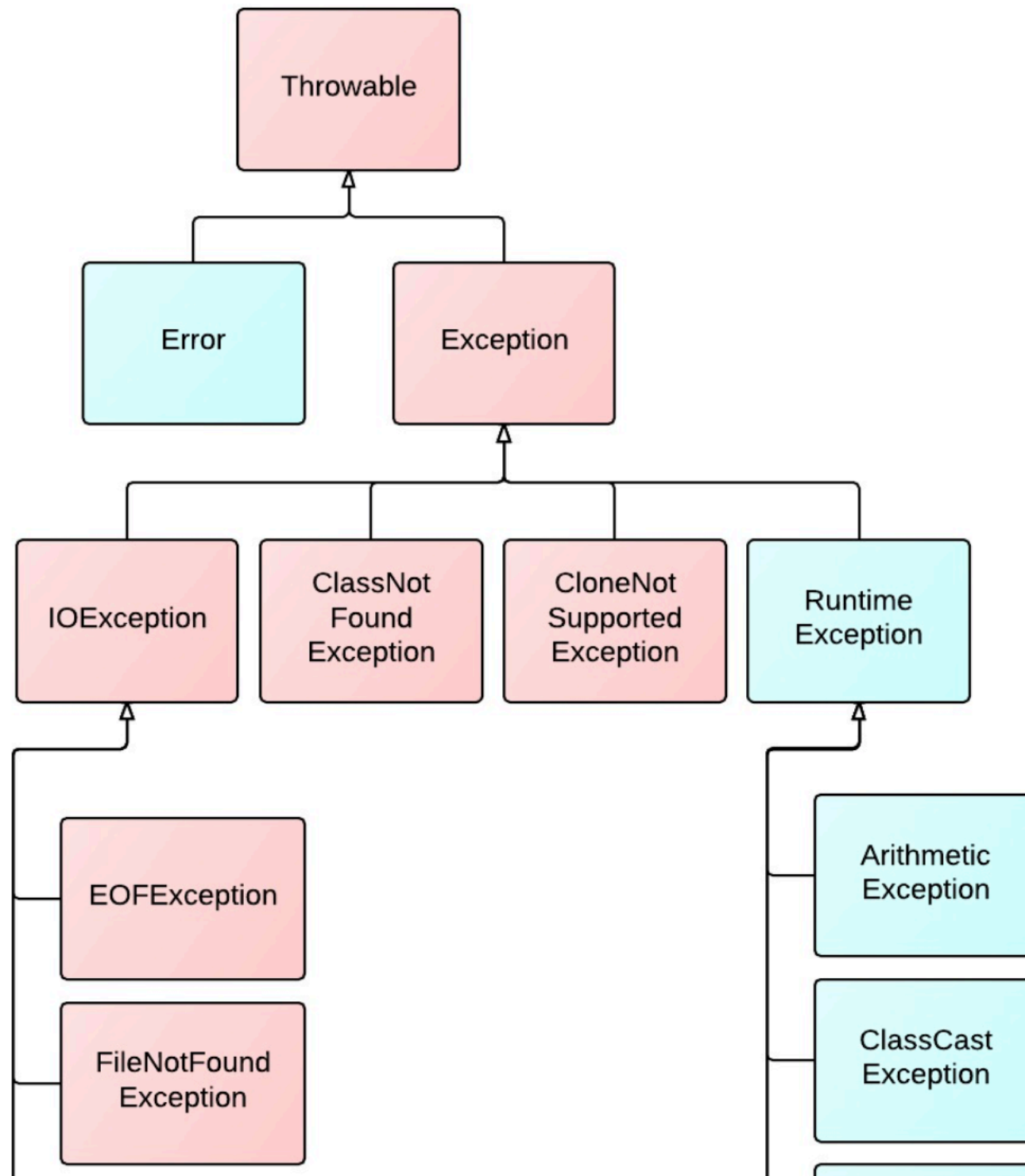


"15.12.2.5. Choosing the Most Specific Method".

Перегрузка методов

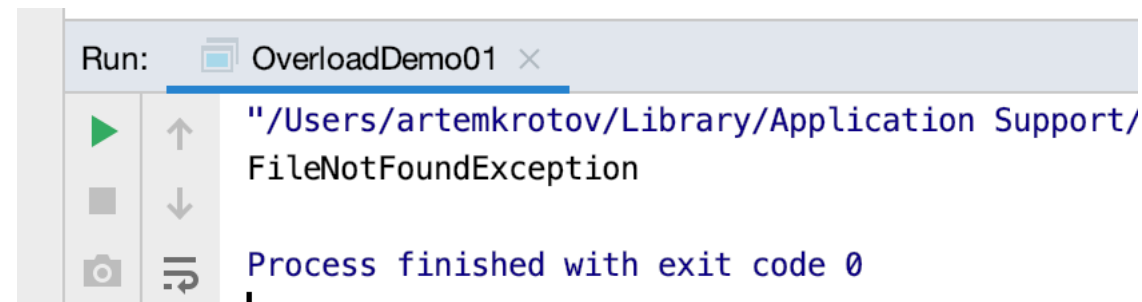
```
1 public class Overload{
2     public void method(Object o) {
3         System.out.println("Object");
4     }
5     public void method(java.io.FileNotFoundException f) {
6         System.out.println("FileNotFoundException");
7     }
8     public void method(java.io.IOException i) {
9         System.out.println("IOException");
10    }
11    public static void main(String args[]) {
12        Overload test = new Overload();
13        test.method(null);
14    }
15 }
```


Перегрузка методов



Перегрузка методов

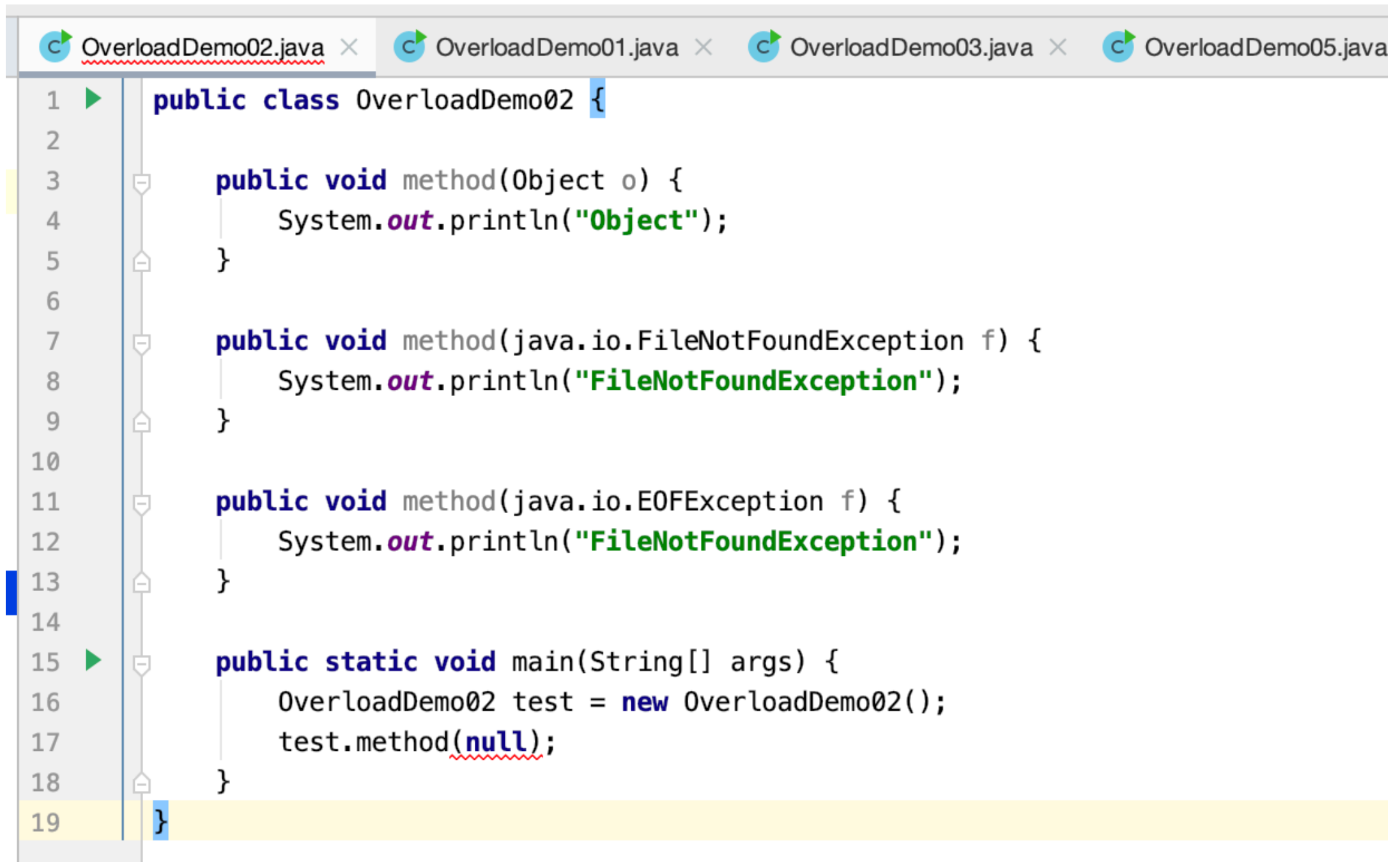
```
1 public class Overload{
2     public void method(Object o) {
3         System.out.println("Object");
4     }
5     public void method(java.io.FileNotFoundException f) {
6         System.out.println("FileNotFoundException");
7     }
8     public void method(java.io.IOException i) {
9         System.out.println("IOException");
10    }
11    public static void main(String args[]) {
12        Overload test = new Overload();
13        test.method(null);
14    }
15 }
```



Перегрузка методов

```
1  ▶ public class OverloadDemo02 {  
2  
3  | public void method(Object o) {  
4      System.out.println("Object");  
5  }  
6  
7  public void method(java.io.FileNotFoundException f) {  
8      System.out.println("FileNotFoundException");  
9  }  
10  
11 public void method(java.io.EOFException f) {  
12     System.out.println("FileNotFoundException");  
13 }  
14  
15 ▶ public static void main(String[] args) {  
16     OverloadDemo02 test = new OverloadDemo02();  
17     // test.method(null);  
18 }  
19 }
```

Перегрузка методов



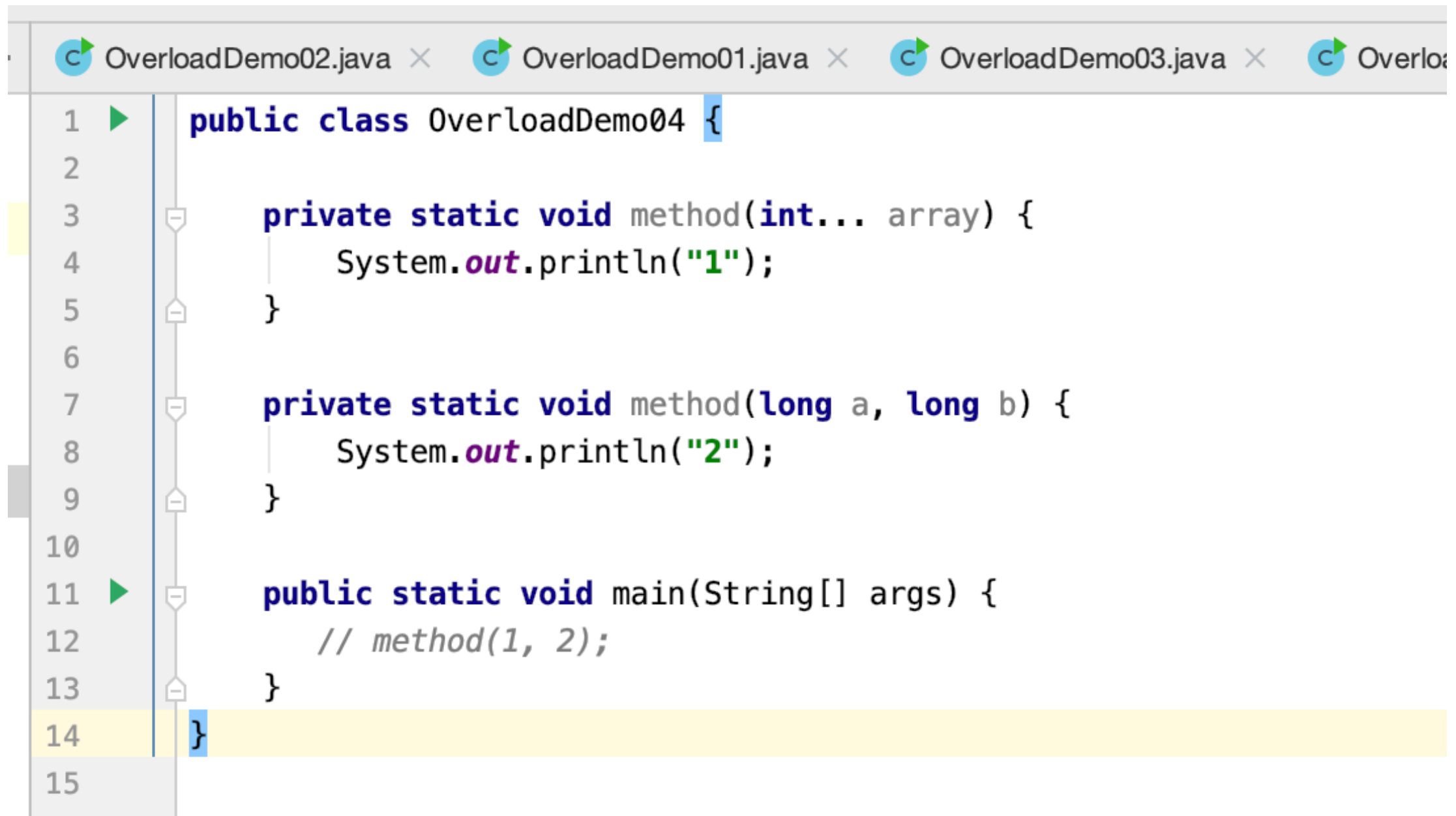
The screenshot shows an IDE with four tabs: OverloadDemo02.java (active), OverloadDemo01.java, OverloadDemo03.java, and OverloadDemo05.java. The code in the active tab defines a class OverloadDemo02 with three overloaded methods and a main method. The methods are: method(Object o), method(java.io.FileNotFoundException f), and method(java.io.EOFException f). The main method creates an instance of OverloadDemo02 and calls the method with null.

```
1 public class OverloadDemo02 {  
2  
3     public void method(Object o) {  
4         System.out.println("Object");  
5     }  
6  
7     public void method(java.io.FileNotFoundException f) {  
8         System.out.println("FileNotFoundException");  
9     }  
10  
11     public void method(java.io.EOFException f) {  
12         System.out.println("FileNotFoundException");  
13     }  
14  
15     public static void main(String[] args) {  
16         OverloadDemo02 test = new OverloadDemo02();  
17         test.method(null);  
18     }  
19 }
```

Перегрузка методов

```
1 public class Overload{
2     public static void method(int... array) {
3         System.out.println("1");
4     }
5
6     public static void main(String args[]) {
7         method(1, 2);
8     }
9 }
```

Перегрузка методов

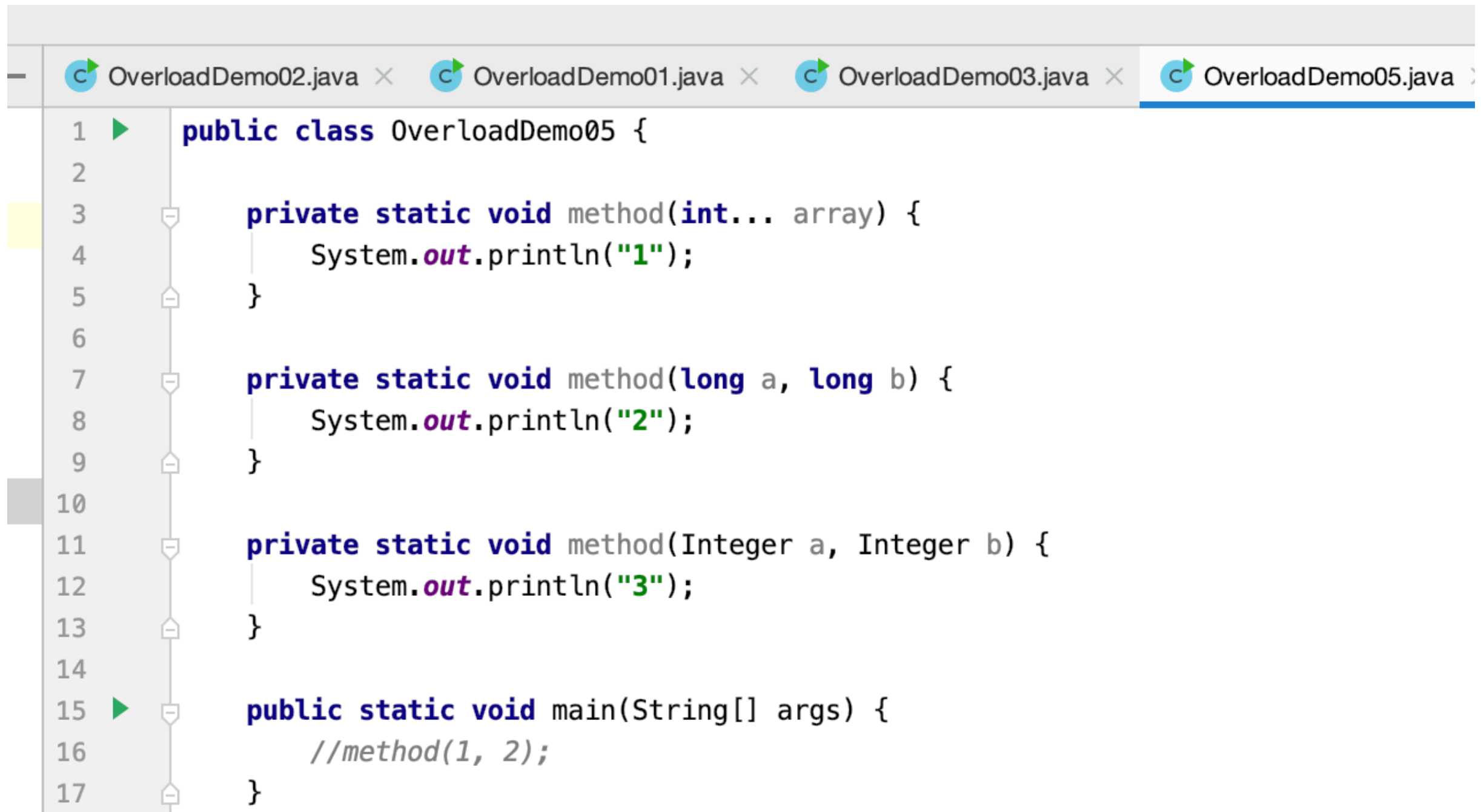


The screenshot shows an IDE window with four tabs: OverloadDemo02.java, OverloadDemo01.java, OverloadDemo03.java, and OverloadDemo04.java. The active tab is OverloadDemo04.java, which contains the following Java code:

```
1  public class OverloadDemo04 {  
2  
3      private static void method(int... array) {  
4          System.out.println("1");  
5      }  
6  
7      private static void method(long a, long b) {  
8          System.out.println("2");  
9      }  
10  
11     public static void main(String[] args) {  
12         // method(1, 2);  
13     }  
14 }  
15
```

The code defines a class `OverloadDemo04` with two overloaded `method` methods: one accepting a variable number of `int` arguments and another accepting two `long` arguments. A `main` method is also present, with a commented-out call to `method(1, 2)`. The IDE interface includes a line number gutter on the left and a vertical scrollbar.

Перегрузка методов



```
1  public class OverloadDemo05 {  
2  
3      private static void method(int... array) {  
4          System.out.println("1");  
5      }  
6  
7      private static void method(long a, long b) {  
8          System.out.println("2");  
9      }  
10  
11     private static void method(Integer a, Integer b) {  
12         System.out.println("3");  
13     }  
14  
15     public static void main(String[] args) {  
16         //method(1, 2);  
17     }
```

Перегрузка методов

```
OverloadDemo02.java × OverloadDemo01.java × OverloadDemo03.java × OverloadDemo05.java ×  
1 public class OverloadDemo06 {  
2  
3     public static void method(int... array) {  
4         System.out.println("1");  
5     }  
6  
7     public static void method(long a, long b) {  
8         System.out.println("2");  
9     }  
10  
11     public static void method(Integer a, Integer b) {  
12         System.out.println("3");  
13     }  
14  
15     public static void main(String[] args) {  
16         method(Integer.valueOf(1), Integer.valueOf(2));  
17     }  
18 }
```