Передача по ссылке или по значению

Передача по значению

00:00–00:50

Привет!

Сегодня мы поговорим о передаче переменных по ссылке и по значению. Если мы возьмём с вами просто число a…

```
int a = 5;
```

…и создадим ещё одно число b, которое равно числу a…

```
int b = a;
```

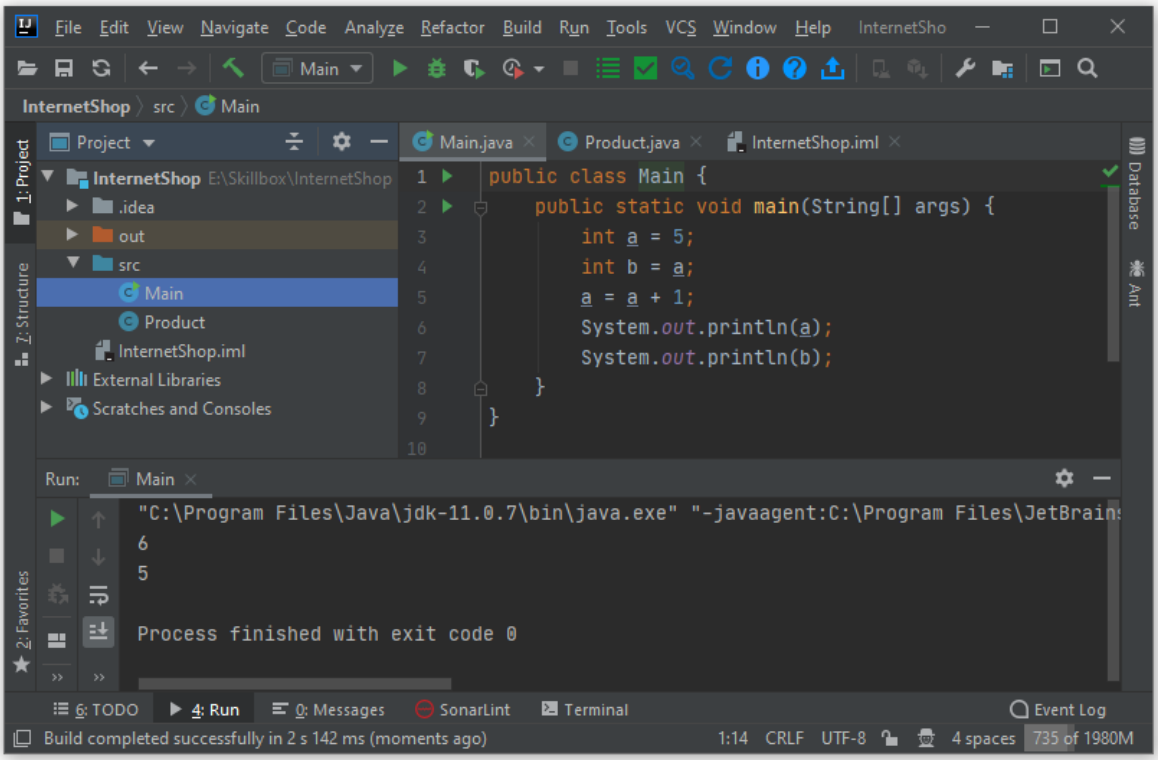
...после чего увеличим число a на единицу и распечатаем эти числа…

```
a = a + 1;
```

…то мы увидим, что число a увеличилось, а число b осталось прежним.

```
System.out.println(a);  //6
System.out.println(b); //5
```

То же в среде разработки:



Мы видим, что в итоге **a = 6**, а **b = 5**. То есть если мы пишем знак равенства, то для чисел у нас по факту создаётся новое место в памяти и новая переменная. И эти переменные никак не связаны.

Передача по ссылке

00:50–04:52

С объектами история другая. Давайте в товар — класс Product — допишем метод toString(), который нам будет возвращать информацию о товаре.

```
public String toString() {
    return name + " - " + price;
}
```

Затем создадим товар a и товар b.

```
Product a = new Product("молоко", 56);
Product b = a;
```

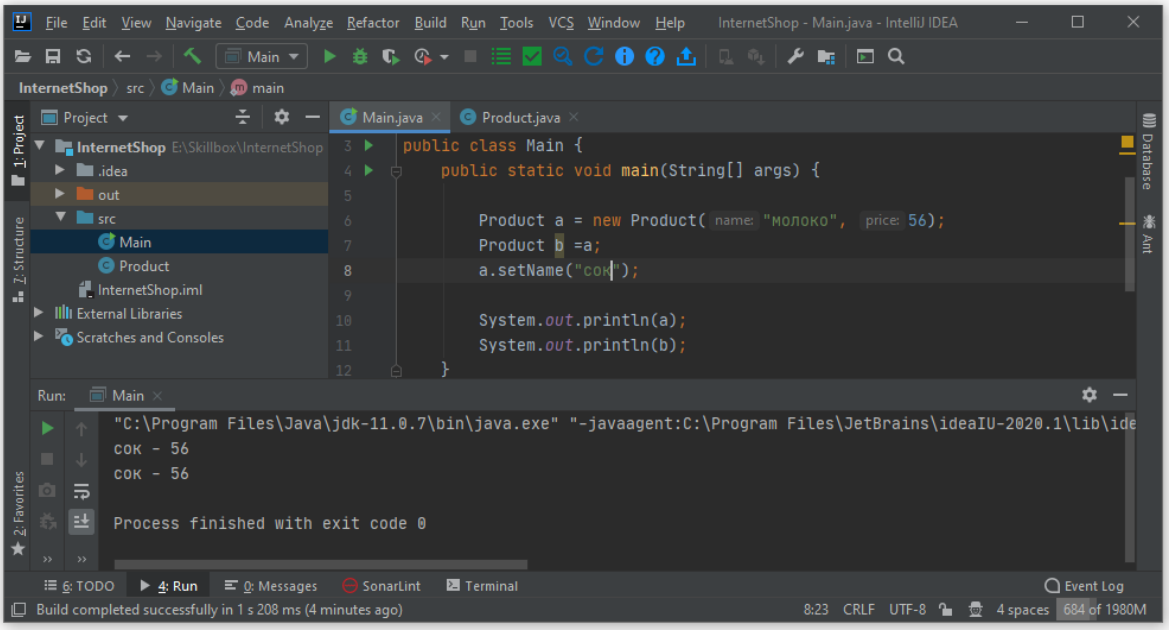
После этого поменяем у товара a имя, теперь это будет «сок».

```
a.setName("сок");
```

Теперь распечатаем товар a и товар b (мы помним, что в случае с числами число a увеличилось, а число b осталось неизменным).

```
System.out.println(a);
System.out.println(b);
```

Запускаем и видим, что объект a и объект b — это один и тот же объект (мы изменили объект a, при этом объект b у нас тоже изменился и стал «соком»).



Почему это происходит? Потому что a и b — это хоть и разные переменные, но они обе ссылаются на один и тот же объект в памяти. И в случае объектов знак равенства не создаёт копию объекта, а создаёт фактически новую ссылку на тот же объект.

Где можно столкнуться с проблемой в реальной жизни? Например, вы создали товар:

```
Product product = new Product("молоко", 56);
```

Дальше его кто-то покупает. Создадим класс Order (заказ), в который можно добавлять товары, и метод addProduct(Product product), который будет добавлять товар в заказ.

```
public class Order {
    public void addProduct(Product product) {
        //adds product
    }
}
```

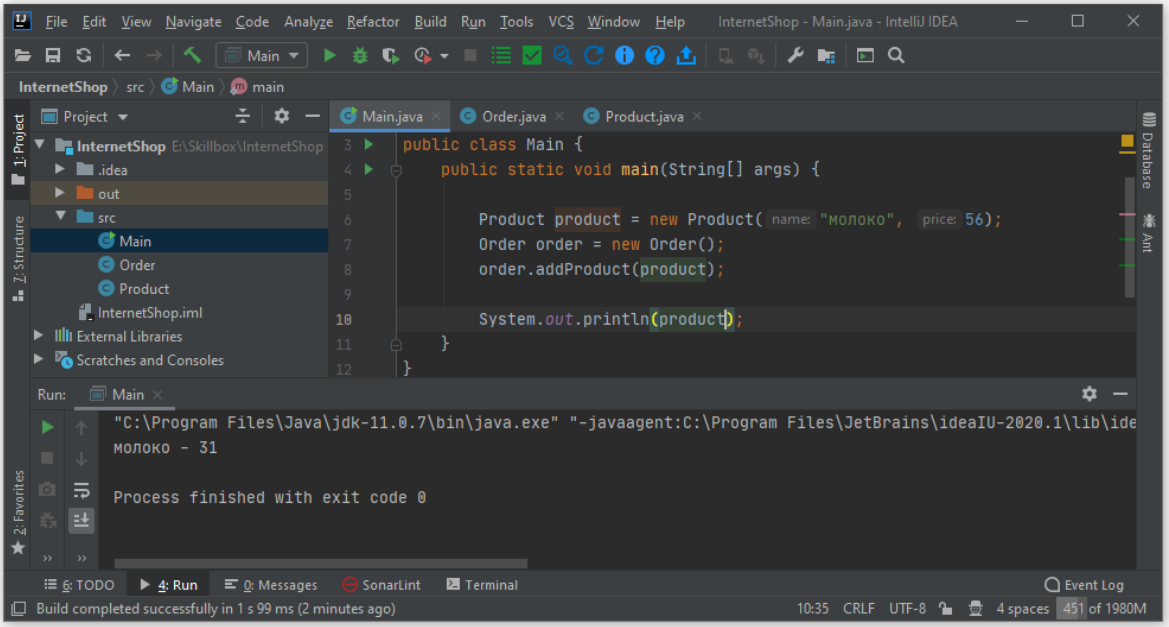
Теперь напишем такой код в классе Main…

```
Order order = new Order();
order.addProduct(product);
System.out.println(product);
```

...и представим, что в этом заказе применяется скидка к некоторым товарам — например, к тем, у которых цена выше 50 рублей. И из цены такого товара вычитается 25 рублей. Напишем это в классе Order в методе addProduct.

```
public class Order {
    public void addProduct(Product product) {
        if (product.getPrice() > 50) {
            product.setPrice(product.getPrice() -
25);
        }
        //adds product
    }
}
```

Запускаем.



Мы видим, что цена у этого товара снизилась во всей программе. Даже если мы добавим этот товар в другие заказы, в которых нет этой скидки, цена везде будет 31.

Таким образом, мы должны помнить о том, что объекты передаются по ссылке. И если мы где-то объект поменяем, то он поменяется везде, поскольку это один и тот же объект.

Если мы хотим обезопасить объект от изменений, мы можем сделать его ***иммутабельным***, и тогда его просто нельзя будет поменять.

Добавляем в объект ключевое слово final перед каждой переменной, удаляем сеттеры, так как изменять эти поля нам больше нельзя.

```
public class Product {
    private final String name;
    private final int price;

    public Product(String name, int price) {
        this.name = name;
        this.price = price;
```

```
    }

    public String getName() {
        return name;
    }

    public int getPrice() {
        return price;
    }

    public String toString() {
        return name + " - " + price;
    }
}
```

Такой объект тоже будет передаваться по ссылке, но изменить его будет нельзя.

Сбор «мусора»

04:52–06:38

Хотелось бы затронуть ещё одну важную тему. Объект, который мы создаём с помощью ключевого слова new, существует в памяти с момента своего создания.

Но что происходит с объектом, если у нас не осталось на него ссылки? Например, код метода, в котором создавался объект и внутри которого он существовал, выполнился?

Например, в классе Product создаём метод вывода информации о продукте в консоль.

```
public void print() {
    String info = name + " - " + price;
    System.out.println(info);
}
```

В нём создаём строку, в которой будет и название товара, и его цена. Что произойдёт с этой строкой info после выполнения метода print? Он так и останется в памяти, но на него не останется ни одной ссылки. По сути, память будет использоваться неэффективно.

Для того чтобы очищать память от таких объектов, на которые больше нет ссылок, в виртуальной машине Java (JVM) есть специальный механизм. Называется он **garbage collector** («**сборщик мусора**»).

Он позволяет писать код, не беспокоясь лишний раз об очищении памяти от объектов, которые нам больше не нужны. Этот подход был придуман давно, но широкую известность получил именно благодаря языку программирования Java, после чего начал массово появляться в других языках программирования.

Итоги

06:38 — до конца

Итак, мы узнали, что объекты передаются по ссылкам и это создаёт опасность их неконтролируемого изменения. Если же мы работаем с объектами иммутабельных классов, то такой проблемы не будет.

Но что делать, если мы всё же захотим менять такие объекты? Ответ: создавать их копии с соответствующими изменениями. О копировании объектов мы и поговорим с вами далее.

Глоссарий

Garbage collector (**сборщик мусора)** — специальный механизм JVM, позволяющий очищать память от объектов, на которые больше нет ссылок.