Урок 7. Области видимости

Привет!

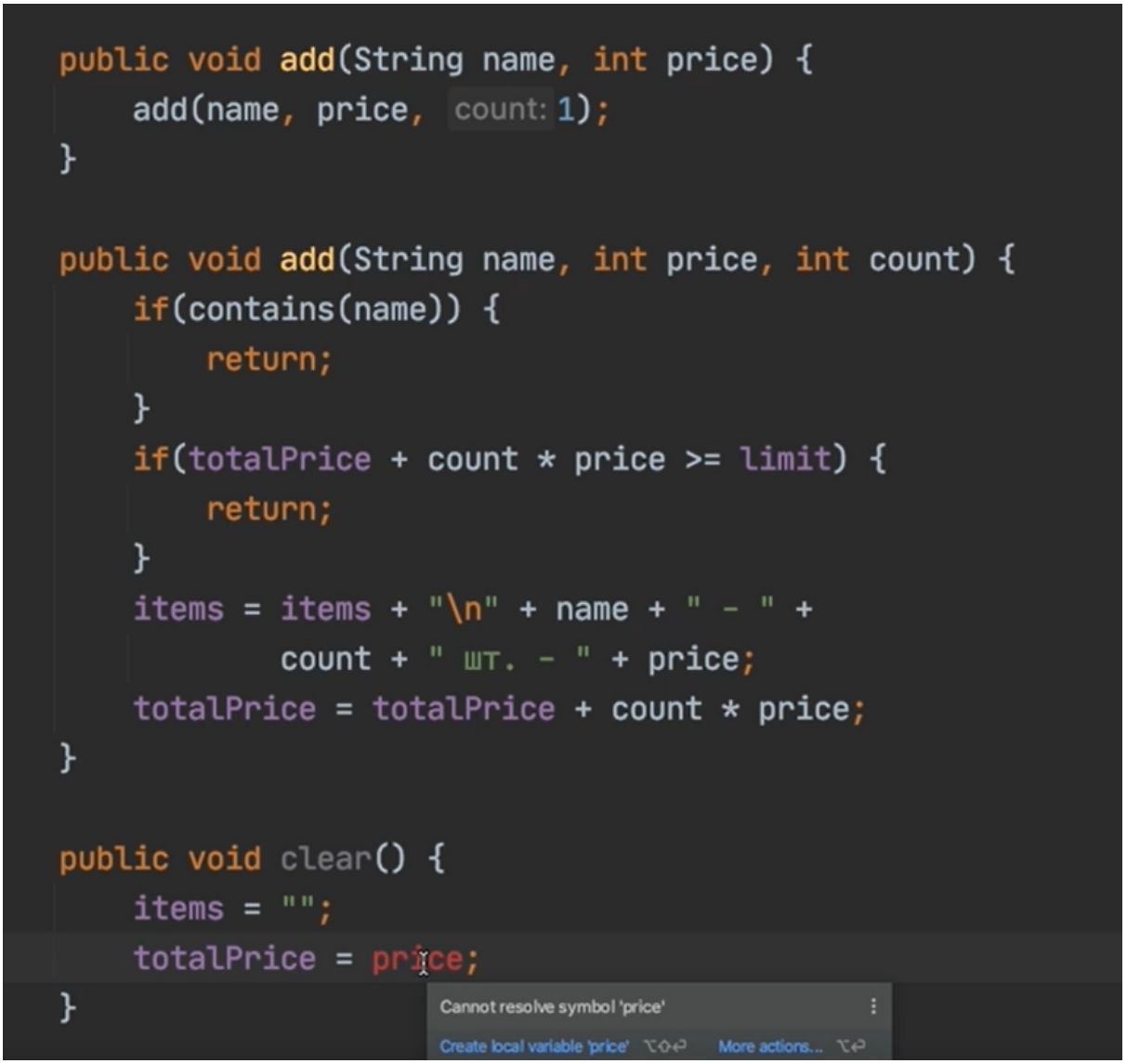
Ранее мы говорили о различных методах и конструкторах, в коде которых использовались не только переменные, переданные в них в качестве параметров, но и переменные, которые доступны на уровне класса.

Доступность переменных в том или ином фрагменте кода определяется областями видимости, о которых мы и поговорим.

Обратите также внимание, что методы могут работать не только со своими параметрами — то есть переменными, которые были им переданы в круглых скобках, — но и с переменными, которые находятся вне этого метода — в том же классе.

Области видимости переменных определяются просто — фигурными скобками тех блоков, на уровне которых они заданы. К примеру, переменная, заданная на уровне класса, доступна во всём классе — в любом его методе или конструкторе. В этом коде переменная items используется практически везде.

А вот если попробовать, например, использовать переданную в метод add переменную price в другом методе, то она там не будет доступна.



Либо если, к примеру, создать в методе какую-то переменную

```
int totalCount = 0;
```

то использовать её в других методах будет нельзя. То есть переменные, созданные в методах или конструкторах, доступны только внутри их области видимости.

```
public void add(String name, int price) {
    add(name, price, count: 1);
    totalCount
}

public void add(String name, int price, int count) {
    int totalCount = 0;
```

Теперь разберёмся с другими конструкциями, в которых используются фигурные скобки. Например, если в теле условия if мы создадим какую-то переменную, то она вне этого условия также не будет доступна:

```
if (contains(name)) {
   boolean error = true;
}
if (totalPrice + count * price >= limit) {
   boolean error = true;
}
if (error) {
   System.out.println("Error");
   return;
}
```

Такой код не работает. Чтобы он заработал, нужно вынести переменную на тот уровень, на котором она будет использоваться, например так:

```
boolean error = false;
if (contains(name)) {
   error = true;
}
if (totalPrice + count * price >= limit) {
   error = true;
}
if (error) {
   System.out.println("Error");
   return;
}
```

Если бы здесь было условие else, то для него этот принцип тоже бы действовал. Также этот принцип действует и для циклов.

Посмотрим на примере:

```
public static void printEvenNumbers() {
    for(int i = 0; i < 1000; i = i + 2) {
        System.out.println(i);
    }
```

Рассмотрим метод printEvenNumbers, который выводит на консоль все чётные числа до 1000. Внутри цикла for мы создаём переменную i типа int. Если мы попробуем вывести эту переменную вне цикла, то получим ошибку

```
public static void printEvenNumbers() {
    for(int i = 0; i < 1000; i = i + 2) {
        System.out.println(i);
    }
    System.out.println(i);
}
```

поскольку область видимости этой переменной — только внутри цикла. Также если мы создадим любую другую переменную внутри цикла, она будет недоступна вне цикла for.

Итоги

В методах и конструкторах доступны объявленные в них переменные, а также переменные, объявленные на уровне класса.

Если в методе есть какой-то код, образующий новые области видимости, например условие if или цикл for, то в этих областях видимости доступны все переменные из «родительских» областей видимости.

При этом соседние области видимости никогда не пересекаются: переменные, объявленные в одном методе, не будут доступны в другом.

Глоссарий

Области видимости