Конструкторы

Привет!

В этом материале рассмотрим ещё один важный компонент классов — конструктор.

До этого говорилось, что классы состоят только из переменных и методов. Представьте, что магазин, корзину для которого мы создавали, находится за границей, и существует некий лимит по стоимости покупок, которые добавляются в неё, при превышении которого нам нужно будет платить повышенную таможенную пошлину.

Этот лимит надо будет учитывать в методе add. Создадим такую переменную:

```
private int limit;
```

и пропишем в коде метода add защиту от добавления товаров и цен сверх этого лимита.

```
public void add(String name, int price) {
    if(contains(name)) {
        return;
    }
    if(totalPrice + price >= limit) {
        return;
    }
```

По сути, эта переменная относится ко всей корзине, и нужно её как-то задавать. Логично было бы задавать её сразу при создании корзины. И вот как раз для инициализации данных при создании объектов существуют конструкторы.

Конструктор с параметрами

01:25–02:50

Создадим конструктор, с помощью которого будем задавать значение этой переменной.

```
public Basket(int totalPriceLimit){
      limit = totalPriceLimit;
}
```

Конструктор выглядит как обычный метод, но только его имя такое же, как и имя класса, и нет возвращаемого значения, но есть параметры.

Ключевое слово this

02:50–03:47

Кстати, если вы решите назвать переменные одинаково:

```
public Basket(int limit){
      limit = limit;
}
```

то код не будет работать, поскольку непонятно, где какая переменная. Если нам необходимо обратиться к параметру класса, вы можете использовать специальное ключевое слово this:

```
public Basket(int limit){
      this.limit = limit;
}
```

И теперь можно создать объект класса Basket с указанием лимита:

```
Basket vasyaBasket = new Basket(15000);
```

То есть скобки, написанные после имени класса, — это место, куда можно вписывать параметры, и эти параметры будут передаваться в конструктор — так же, как и параметры передаются в вызываемые методы.

Конструктор без параметров

03:47–04:10

Обратите внимание, что если мы создали конструктор с одним параметром, то создавать корзину без параметров уже не получится. Чтобы вернуть такую возможность, создадим конструктор без параметров.

```
public Basket(){
```

```
}
```

Зачем ещё может быть нужен конструктор? Например, если мы хотим какие-то данные инициализировать сразу, например создание строки с текстом «Список товаров», это правильно делать в конструкторе.

```
public Basket(int limit) {
    items = "Список товаров:";
    this.limit = limit;
}
```

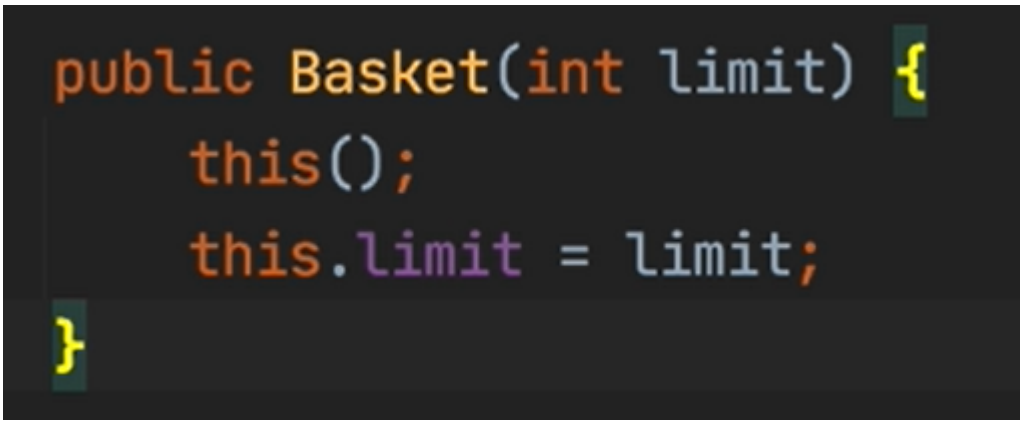
```
public Basket() {
    items = "Список товаров:";
    this.limit = 1000000;
}
```

В таком случае возникнет дублирование кода, которого нужно избегать. Потому что, если возникнет необходимость изменить этот текст, придётся делать это в обоих конструкторах. И чтобы избежать дублирования, выполним следующие действия:

В конструкторе с параметрами уберём строку

```
items = "Список товаров: ";
```

и перед тем, как задавать лимит, вызовем конструктор, который не содержит параметров, с помощью ключевого слова this, чтобы сначала выполнилась инициализация в конструкторе без параметров, а потом уже сработала строка кода с установлением лимита. В этом случае мы инициализируем items один раз в конструкторе без параметров.



Создадим третий конструктор с двумя параметрами.

Пусть это будет конструктор, который инициализирует нашу корзину из заранее сохранённого списка товаров и общей цены.

```
public Basket(String items, int totalPrice){
      this();
      this.items = this.items + items;
      this.totalPrice = totalPrice;
}
```

И добавим ещё одну корзину, используя этот конструктор:

```
Basket mashaBasket = new Basket("Стол", 5000);
```

Итоги

08:04 — до конца видео

Вы изучили конструкторы. По сути, это такие же методы, но их имена совпадают с именем класса.

Они вызываются, когда создаётся объект класса, с помощью ключевого слова new. Их может быть несколько, и в них можно передавать параметры.

Параметров может не быть, может быть один или несколько. Таким образом, классы могут состоять не только из переменных и методов, но и из конструкторов.

Глоссарий

Ключевое слово this, конструктор, конструктор без параметров, конструктор с параметрами