## Статические методы

Привет!

Предыдущий материал был посвящён статическим переменным. Однако статическими могут быть не только переменные, но и методы. О них и пойдёт речь.

Статические методы, как и статические переменные, принадлежат к классам, а не к объектам. В связи с этим они не могут работать с переменными объектов, но могут — со статическими переменными.

Посмотрим на примере.

Ранее в классе Basket мы создали переменную count публичной, что позволяет изменить её снаружи класса произвольно, а это нарушит инкапсуляцию и может вызвать непредсказуемое поведение программы.

Например, кто-то в коде пропишет:

```
Basket.count = 0;
```

и потом забудет убрать. А кто-то следующий решит посчитать среднюю стоимость корзин и где-то в другом коде выполнит вычисление:

```
int averagePrice = totalPrice / Basket.count;
```

и получит ошибку, поскольку на ноль делить нельзя.

Поэтому, чтобы защитить переменную count от несанкционированного изменения, сделаем её приватной и создадим метод, через который мы сможем получать её значение:

```
public static int getBasketCount() {
      return count;
}
```

Теперь поменять значение переменной произвольно не получится, а код перестал компилироваться.

Статические методы, так же как и переменные, можно вызывать по имени класса, поэтому вместо обращения к переменной count пропишем вызов метода getBasketCount:

```
int averagePrice = totalPrice / Basket.getBasketCount();
```
Вы уже увидели, что обращение к статическим переменным происходит либо по имени класса, либо, если мы работаем в этом же классе, напрямую.

Если у вас возникает конфликт имён — то есть вы делаете метод, в который передаётся параметр, и такой же статический параметр доступен в классе, с которым мы работаем, — то можно к нему обращаться так же через имя класса, как будто мы работаем с параметром откуда-то снаружи:

```
public void increaseCount(int count) {
      Basket.count = Basket.count + count;
}
```
При обращении к переменным объекта мы пишем this, а при обращении к переменным класса — имя этого класса. В случае же если конфликта имён нет, то и в статических, и в нестатических методах можно обращаться к любым переменным напрямую.

Обратите внимание, что метод выше нестатический. Со статическими переменными и методами могут работать любые другие методы и конструкторы.

Рассмотрим это ещё на одном примере. Сделаем метод, который увеличивает количество корзин, статическим:

```
public static void increaseCount(int count) {
      Basket.count = Basket.count + count;
}
```
и будем вызывать его в конструкторах:

```
increaseCount(1);
```
При этом статические методы могут работать только со статическими переменными. Почему так? Потому что статические методы и переменные, по сути, существуют в контексте класса, а не объекта, и вызывать их можно по имени класса.

Когда мы это делаем, объекта может ещё не быть. Но зато если мы создали объект какого-то класса, то этот класс точно есть, и из нестатических методов и конструкторов мы вполне можем обращаться к статическим методам и переменным.

Теперь пара слов о том, каким сделать метод — статическим или нестатическим. Если предполагается работать с ним через объекты и он относится к отдельному объекту по своей сути, то он должен быть нестатическим.

Вот, например, уже известный метод add в классе Basket добавляет товары в конкретную корзину, в конкретный объект. Очевидно, что он должен быть только нестатическим, поскольку меняет переменные объекта totalPrice или items. Если же метод принадлежит к классу и не работает с переменными объекта, он может и, по-хорошему, должен быть статическим.

Статический метод ограничен тем, что он может работать только с переданными в него параметрами, статическими переменными и другими статическими методами этого же класса.

Вы уже знаете, что выполнение любой программы начинается с метода main. Этот метод статический, и его вызов при запуске программы происходит напрямую, без создания объекта класса, в котором он находится.

Если вы хотите, чтобы в этом же классе вызывались какие-то методы, они должны быть только статическими. Если они будут нестатические, вы не сможете их вызвать:

```
public void createBasket() {
      Basket basket = new Basket();
}
```

## Итоги

Вы познакомились со статическими методами и узнали, что эти методы не могут работать с нестатическими переменными и методами.

## Глоссарий

статические методы, нестатические методы