Пример 2: Программа "Book"

Цель примера

Продемонстрировать использование конструкторов при создании объекта, а также вспомогательные методы для работы со свойствами класса.

Поэтапное создание программы

У каждого класса есть понятие "конструктор", которое определяет процесс формирования нового объекта и также его начальное состояние. В рамках данного примера будет объявлен класс – книга `Book` со следующим набором свойств:

- Название: `String title`
- Издательство: `String publisher`
- Уникальный ISBN код: `String isbn`
- Количество страниц: `int pageCount`

Шаг 1.

Созданим класс `Book` в соответствии со свойствами, которые описаны выше:

```
public class Book {

    public String title;
    public String publisher;
    public String isbn;
    public int pageCount;

}
```

Шаг 2.

Создадим класс `BookDemo`, в котором создадим экземпляр книги:

```
public class BookDemo {

    public static void main(String[] args) {

        Book javaBook = new Book();

    }

}
```

Использование оператора `new` вместе с `Book()` вызывает конструктор класса `Book`. Обратите внимание, что в классе `Book` нигде явно конструктор не описан. Это связано с тем, что у любого класса по-умолчанию **всегда** имеется пустой конструктор.

Шаг 3.

Выведем на экран все свойства нашей книги после ее создания:

```
public class BookDemo {

    public static void main(String[] args) {

        Book javaBook = new Book();

        System.out.println("javaBook.title = " + javaBook.title);
        System.out.println("javaBook.publisher = " + javaBook.publisher);
        System.out.println("javaBook.isbn = " + javaBook.isbn);
        System.out.println("javaBook.pageCount = " + javaBook.pageCount);

    }

}
```

Результат работы программы:

```
javaBook.title = null javaBook.publisher = null javaBook.isbn = null javaBook.pageCount = 0
```

Все ссылочные (reference) типы данных по умолчанию имеют значение `null`, в отличие от примитивных типов.

⚠️ **Важно**: Все классы являются ссылочными типами. Значение `null` значит, что у переменной отсутсвует ссылка на объект.

Шаг 4.

Для того, чтобы изменить состояние объекта, можно обратиться к свойству через оператор `.` и присвоить желаемое значение:

```
public class BookDemo {

    public static void main(String[] args) {
```

```
        Book javaBook = new Book();

        javaBook.title = "Head First Java";
        javaBook.publisher = "O'reilly";
        javaBook.isbn = "0596009208";
        javaBook.pageCount = 688;

        System.out.println("javaBook.title = " + javaBook.title);
        System.out.println("javaBook.publisher = " + javaBook.publisher);
        System.out.println("javaBook.isbn = " + javaBook.isbn);
        System.out.println("javaBook.pageCount = " + javaBook.pageCount);



    }

}
```

Результат работы программы:

```
javaBook.title = Head First Java javaBook.publisher = O'reilly javaBook.isbn = 0596009208 javaBook.pageCount = 688
```

Шаг 5.

В случае, если необходимо присвоить свойства в момент создания объекта, нужно объявлять конструктор в классе `Book`. Допустим, для присвоения названия книги и издателя в момент создания книги, конструктор будет выглядеть следующим образом:

```
public class Book {

    public String title;
    public String publisher;
    public String isbn;
    public int pageCount;

    public Book(String title, String publisher) {
        this.title = title;
        this.publisher = publisher;
    }

}
```

Шаг 6.

Теперь в момент создания объекта можно указать его название и издателя в качестве параметров:

```
public class BookDemo {

    public static void main(String[] args) {

        Book javaBook = new Book("Head First Java", "O'reilly");

        System.out.println("javaBook.title = " + javaBook.title);
        System.out.println("javaBook.publisher = " + javaBook.publisher);
```

```
        System.out.println("javaBook.isbn = " + javaBook.isbn);
        System.out.println("javaBook.pageCount = " + javaBook.pageCount);

    }

}
```

Результат работы программы:

```
javaBook.title = Head First Java javaBook.publisher = O'reilly javaBook.isbn = null javaBook.pageCount = 0
```

Как видно, те значения, которые были переданы через конструктор, уже присвоены. Свойства `isbn` и `pageCount` имеют значения по умолчанию в соответствии со своим типом.

⚠️ **Важно**: Если в объявлении класса присутствует хотя бы один явный конструктор, пустой конструктор (неявный) по умолчанию уже не будет создан.

Шаг 7.

У класса может быть множество конструкторов, уникальных по критерию сигнатуры. Например, создать еще один конструктор, который инициализирует `publisher` и `isbn` нельзя, поскольку возникает конфликт, так как имя параметра игнорируется:

- `Book(String title, String publisher)` -> `Book(String, String)`
- `Book(String publisher, String isbn)` -> `Book(String, String)`

Таким образом, добавлять можно только такие конструкторы, у которых сигнатуры будут отличаться, например:

```
public class Book {

    public String title;
    public String publisher;
    public String isbn;
    public int pageCount;

    public Book() {
    }

    public Book(String title, String publisher) {
        this.title = title;
        this.publisher = publisher;
    }

    public Book(String title, String publisher, String isbn, int pageCount) {
        this.title = title;
        this.publisher = publisher;
        this.isbn = isbn;
        this.pageCount = pageCount;
    }

}
```

Шаг 8.

Для вывода информации о книге, в класс `Book` можно добавить метод `printInformation()`, чтобы избежать дублирование кода в местах, где нужно вывести состояние объекта:

```
public class Book {

    public String title;
    public String publisher;
    public String isbn;
    public int pageCount;

    public Book() {
    }

    public Book(String title, String publisher) {
        this.title = title;
        this.publisher = publisher;
    }

    public Book(String title, String publisher, String isbn, int pageCount) {
        this.title = title;
        this.publisher = publisher;
        this.isbn = isbn;
        this.pageCount = pageCount;
    }

    public void printInformation() {
        System.out.println("javaBook.title = " + this.title);
        System.out.println("javaBook.publisher = " + this.publisher);
        System.out.println("javaBook.isbn = " + this.isbn);
        System.out.println("javaBook.pageCount = " + this.pageCount);
    }

}
```

Теперь вывести информацию о книге можно следующим образом:

```
public class BookDemo {

    public static void main(String[] args) {

        Book javaBook = new Book("Head First Java", "O'reilly");
        javaBook.printInformation();

    }

}
```

Результат работы программы:

```
javaBook.title = Head First Java javaBook.publisher = O'reilly javaBook.isbn = null javaBook.pageCount = 0
```

⚠️ **Важно**: Методы, которые не возвращают результат также называют процедурами, поскольку они выполняют простую последовательность действий. В сигнатуре метода пишут ключевое слово `void` на месте возвращаемого типа, чтобы показать отсутствие возвращаемого значения.

⚠️ **Важно**: В `void` методах допускается использование оператора `return`, однако в таком случае метод прекратит свое исполнение без возвращения какого либо результата.

Рекомендации:

- Запустить программу и сравнить результаты;
- Добавить другие конструкторы и изучить свойства объектов после их создания;
- Создать разные экземпляры книг с разными конструкторами и изучить их свойства;