

МИНОБРНАУКИ РОССИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«ПОВОЛЖСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СЕРВИСА»
(ФГБОУ ВО «ПВГУС»)

Кафедра «Прикладная информатика в экономике»

ЛАБОРАТОРНЫЙ ПРАКТИКУМ

по дисциплине «Язык программирования Java и средства NetBeans»

Одобрено
Учебно-методическим
Советом университета

Составитель **Малышева Е. Ю.**

Тольятти 2016

УДК 002.52/54(075.8)
ББК -018*32.973я7
Л 12

Рецензент
к.т.н., доц. **Бобровский С. М.**

Л 12 Лабораторный практикум по дисциплине «Язык програм-
мирования Java и средства NetBeans» / сост. Е. Ю. Малышева. –
Тольятти : Изд-во ПВГУС, 2016. – 48 с.

Лабораторный практикум по дисциплине «Язык программирования Java и
средства NetBeans» разработан в соответствии с требованиями ФГОС ВО

УДК 002.52/54(075.8)
ББК -018*32.973я7

© Малышева Е. Ю., составление, 2016
© Поволжский государственный
университет сервиса, 2016

Лабораторная работа № 2. Создание классов и объектов в Java

Цель работы: Научиться создавать классы и объекты в Java

Задание: создать объектно-ориентированное приложение в Java

Указания к работе:

Класс – это описание того, как будет устроен объект, являющийся экземпляром данного класса, а также какие методы объект может вызывать. Заметим, что методы, в отличие от других подпрограмм, могут напрямую обращаться к данным своего объекта. Так как экземплярами классов ("воплощением" в реальность того, что описано в классе) являются объекты, классы называют объектными типами.

Все объекты, являющиеся экземплярами некоторого класса, имеют одинаковые наборы полей данных (атрибуты объекта) – но со значениями этих данных, которые свои для каждого объекта. Поля данных это переменные, заданные на уровне описания класса, а не при описании метода. В процессе жизни объекта эти значения могут изменяться. Значения полей данных объекта задают его состояние. А методы задают поведение объекта. Причем в общем случае на это поведение влияет состояние объекта – методы пользуются значениями его полей данных.

Классы в Java задаются следующим образом. Сначала пишется зарезервированное слово `class`, затем имя класса, после чего в фигурных скобках пишется реализация класса – задаются его поля (глобальные переменные) и методы.

Объектные переменные – такие переменные, которые имеют объектный тип. В Java объектные переменные – это не сами объекты, а только ссылки на них. То есть все объектные типы являются ссылочными.

Объявление объектной переменной осуществляется так же, как и для других типов переменных. Сначала пишется тип, а затем через пробел имя объявляемой переменной. Например, если мы задаем переменную `obj1` типа `Circle`, "окружность", ее задание осуществляется так:

```
Circle obj1;
```

Связывание объектной переменной с объектом осуществляется путем присваивания. В правой части присваивания можно указать либо функцию, возвращающую ссылку на объект (адрес объекта), либо имя другой объектной переменной. Если объектной переменной не присвоено ссылки, в ней хранится значение `null`. Объектные переменные можно сравнивать на равенство, в том числе на равенство `null`. При этом сравниваются не сами объекты, а их адреса, хранящиеся в объектных переменных.

Создается объект с помощью вызова специальной подпрограммы, задаваемой в классе и называемой конструктором. Конструктор возвращает ссылку на созданный объект. Имя конструктора в Java всегда совпадает с именем класса, экземпляр которого создается. Перед именем конструктора во время вызова ставится оператор `new` – "новый", означающий, что создается новый объект. Например, вызов

```
obj1=new Circle();
```

означает, что создается новый объект типа `Circle`, "окружность", и ссылка на него (адрес объекта) записывается в переменную `obj1`. Переменная `obj1` до этого уже должна быть объявлена. Оператор `new` отвечает за динамическое выделение памяти под создаваемый объект.

Часто совмещают задание объектной переменной и назначение ей объекта. В нашем случае оно будет выглядеть как

```
Circle obj1=new Circle();
```

У конструктора, как и у любой подпрограммы, может быть список параметров. Они нужны для того, чтобы задать начальное состояние объекта при его создании. Например, мы хотим, чтобы у создаваемой окружности можно было при вызове конструктора задать координаты x , y ее центра и радиус r . Тогда при написании класса `Circle` можно предусмотреть конструктор, в котором первым параметром задается координата x , вторым – y , третьим – радиус окружности r . Тогда задание переменной `obj1` может выглядеть так:

```
Circle obj1=new Circle(130,120,50);
```

Оно означает, что создается объект-окружность, имеющий центр в точке с координатами $x=130$, $y=120$, и у которой радиус $r=50$.

Если разработчики класса не создали ни одного конструктора, в реализации класса автоматически создается конструктор по умолчанию, имеющий пустой список параметров. И его можно вызывать в программе так, как мы это первоначально делали для класса `Circle`.

Отметим еще одно правило, касающееся используемых имен. Как мы помним, имена объектных типов принято писать с заглавной буквы, а имена объектных переменных – с маленькой. Если объектная переменная имеет тип `Circle`, она служит ссылкой на объекты-окружности. Поэтому имя `obj1` не очень удачно – мы используем его только для того, чтобы подчеркнуть, что именно с помощью этой переменной осуществляется связь с объектом, и чтобы читатель не путал тип переменной, ее имя и имя конструктора. В Java принято называть объектные переменные так же, как их типы, но начинать имя со строчной буквы. Поэтому предыдущий оператор мог бы выглядеть так:

```
Circle circle=new Circle(130,120,50);
```

Если требуется работа с несколькими объектными переменными одного типа, их принято называть в соответствии с указанным выше правилом, но добавлять порядковый номер. Следующие строки программного кода создают два независимых объекта с одинаковыми начальными параметрами:

```
Circle circle1=new Circle(130,120,50);  
Circle circle2=new Circle(130,120,50);
```

С помощью объектных переменных осуществляется доступ к полям данных или методам объекта: сначала указывается имя переменной, затем точка, после чего пишется имя поля данных или метода. Например, если имя объектной переменной `circle1`, а имя целочисленного поля данных `x`, то присваивание ему нового значения будет выглядеть как

```
circle1.x=5;
```

А если имя подпрограммы `show`, у нее нет параметров и она не возвращает никакого значения, то ее вызов будет выглядеть как

```
circle1.show();
```

Ход работы:

1. Создайте проект Java. Назовите пакет `com.example`, а главный класс `EmployeeTest`
2. Создайте пакет `com.example.domain`, а в нем класс `Employee` с указанными полями:

Field name	Access	Recommended field type
<code>empId</code>	<code>public</code>	<code>int</code>
<code>name</code>	<code>public</code>	<code>String</code>
<code>ssn</code>	<code>public</code>	<code>String</code>
<code>salary</code>	<code>public</code>	<code>double</code>

```
public int empId;  
public String name;  
public String ssn;  
public double salary;
```

3. Добавьте конструктор класса:

```
public Employee() {}
```

4. Создайте методы чтения и записи («геттеры» и «сеттеры») для каждого поля. Используйте для этого контекстное меню редактора NetBeans

5. Добавьте в файл класса EmployeeTest импорт класса Employee

```
import com.example.domain.Employee;
```

6. Добавьте в процедуру main класса EmployeeTest команды создания объекта класса Employee и заполнение его полей

```
Employee emp = new Employee();  
emp.setEmpId(101);  
emp.setName("Jane Smith");  
emp.setSalary(120_345.27);  
emp.setSsn("012-34-5678");
```

7. Добавьте в процедуру main класса EmployeeTest команды отображения данных объекта класса Employee

```
System.out.println("Employee ID: "+emp.getEmpId());  
System.out.println("Employee Name: "+emp.getName());  
System.out.println("Employee Soc Sec # "+emp.getSsn());  
System.out.println("Employee salary: "+emp.getSalary());
```

8. Запустите приложение. Результат должен выглядеть примерно так:

```
Employee id:      101  
Employee name:    Jane Smith  
Employee Soc Sec #: 012-34-5678  
Employee salary:  120345.27  
BUILD SUCCESSFUL (total time: 1 second)
```

Вопросы для самоконтроля

1. Что такое модификатор доступа в классе?
2. Какие бывают модификаторы доступа в классе?
3. Что такое поле (переменная) класса?
4. Что такое метод класса?

Варианты заданий

<i>Вариант</i>	<i>Класс</i>
1	Сотрудник, 3 поля
2	Студент, 2 поля
3	Товар, 3 поля

4	Собака, 2 поля
5	Геометрическая фигура, 2 поля
6	Программное обеспечение, 3 поля
7	Аппаратное обеспечение, 3 поля
8	Город, 2 поля
9	Страна, 2 поля
10	Книга, 3 поля

Литература: 1, 2, 4, 5.

Учебное издание

ЛАБОРАТОРНЫЙ ПРАКТИКУМ

по дисциплине «**Язык программирования Java и средства NetBeans**»

Составитель

Малышева Елена Юрьевна

Издается в авторской редакции.

Подписано в печать с электронного оригинал-макета 30.06.2016.

Бумага офсетная. Печать трафаретная. Усл. печ. л. 3,0.

Тираж 500 экз. Заказ 78/01.

Издательско-полиграфический центр

Поволжского государственного университета сервиса.

445677, г. Тольятти, ул. Гагарина, 4.

rio@tolgas.ru, тел. (8482) 222-650.

Электронную версию этого издания

вы можете найти на сайте ЭБС ПВГУС <http://elib.tolgas.ru/>.