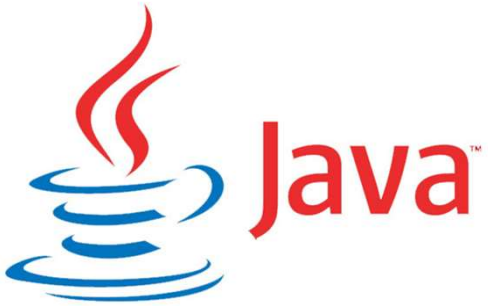
Тема 6.
Классы и объекты



Понятие класса

Java является объектно-ориентированным языком, поэтому такие понятия как "класс" и "объект" играют в нем ключевую роль. Любую программу на Java можно представить как набор взаимодействующих между собой объектов.

Шаблоном или описанием объекта является класс, а объект представляет экземпляр этого класса.

Понятие класса

Можно еще провести следующую аналогию.

У нас у всех есть некоторое представление о человеке - наличие двух рук, двух ног, головы, туловища и т.д.

Есть некоторый шаблон - этот шаблон можно назвать классом. Реально же существующий человек (фактически экземпляр данного класса) является объектом этого класса.

Понятие класса

Класс определяется с помощью ключевого слова class:

```
class Person{

}
```

В данном случае класс называется Person. После названия класса идут фигурные скобки, между которыми помещается тело класса - то есть его поля и методы.

Понятие класса

Любой объект может обладать двумя основными характеристиками:

- состояние - некоторые данные, которые хранит объект
- поведение - действия, которые может совершать объект.

Для хранения состояния объекта в классе применяются поля или **переменные класса**.

Для определения поведения объекта в классе применяются **методы**.

Понятие класса

Например, класс Person, который представляет человека, мог бы иметь следующее определение:

```
class Person{

    String name;          // имя
    int age;              // возраст
    void displayInfo(){
        System.out.printf("Name: %s \tAge: %d\n", name, age);
    }
}
```

В классе Person определены два поля: **name** представляет имя человека, а **age** - его возраст. И также определен метод **displayInfo**, который ничего не возвращает и просто выводит эти данные на консоль.

Понятие класса

Как теперь использовать этот класс:

```
public class Program{

    public static void main(String[] args) {

        Person tom;
    }
}
class Person{

    String name;    // имя
    int age;        // возраст
    void displayInfo(){
        System.out.printf("Name: %s \tAge: %d\n", name, age);
    }
}
```

Понятие класса

Как правило, классы определяются в **разных файлах**. В данном случае для простоты мы определяем два класса в одном файле.

В этом случае только один класс может иметь модификатор **public** (в данном случае это класс **Program**), а сам файл кода должен называться по имени этого класса, то есть в данном случае файл должен называться Program.java.

Про модификаторы позже :-)

Понятие класса

Класс представляет новый тип, поэтому мы можем определять переменные, которые представляют данный тип. Так, здесь в методе main определена переменная tom, которая представляет класс Person.

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

    Person tom;
}
```

Но пока эта переменная не указывает ни на какой объект и по умолчанию она имеет значение null.

Понятие класса

Если переменные и константы хранят некоторые значения, то методы содержат собой набор операторов, которые выполняют определенные действия.

Структура метода:

```
[модификаторы] тип_возвращаемого_значения имя_метода ([параметры]){
    // тело метода
}
```

Модификаторы и параметры необязательны.

Понятие метода класса

Если переменные и константы хранят некоторые значения, то методы содержат собой набор операторов, которые выполняют определенные действия.

Структура метода:

```
[модификаторы] тип_возвращаемого_значения имя_метода ([параметры]){
    // тело метода
}
```

Модификаторы и параметры необязательны.

Понятие метода класса

По умолчанию главный класс любой программы на Java содержит метод main, который служит точкой входа в программу.

```
public static void main(String[] args) {
    System.out.println("привет мир!");
}
```

Ключевые слова **public** и **static** являются модификаторами. Далее идет тип возвращаемого значения.

Ключевое слово **void** указывает на то, что метод ничего не возвращает.

Затем идут название метода - **main** и в скобках параметры метода - **String[] args**. И в фигурные скобки заключено тело метода - все действия, которые он выполняет.

Создание объекта

Для создания объекта Person используется выражение new Person(). Оператор new выделяет память для объекта Person. В итоге после выполнения данного выражения в памяти будет выделен участок, где будут храниться все данные объекта Person. А переменная tom получит ссылку на созданный объект.

```
Person tom = new Person(); // создание объекта
tom.displayInfo();
```

Конструкторы класса

Кроме обычных методов классы могут определять специальные методы, которые называются конструкторами. Конструкторы вызываются при создании нового объекта данного класса. Конструкторы выполняют инициализацию объекта.

Если в классе не определено ни одного конструктора, то для этого класса автоматически создается конструктор без параметров.

Выше определенный класс Person не имеет никаких конструкторов. Поэтому для него автоматически создается конструктор по умолчанию, который мы можем использовать для создания объекта Person.

Конструкторы класса

Если конструктор не инициализирует значения переменных объекта, то они получают значения по умолчанию. Для переменных числовых типов это число 0, а для типа string и классов - это значение null (то есть фактически отсутствие значения).

После создания объекта мы можем обратиться к переменным объекта Person через переменную tom и установить или получить их значения.

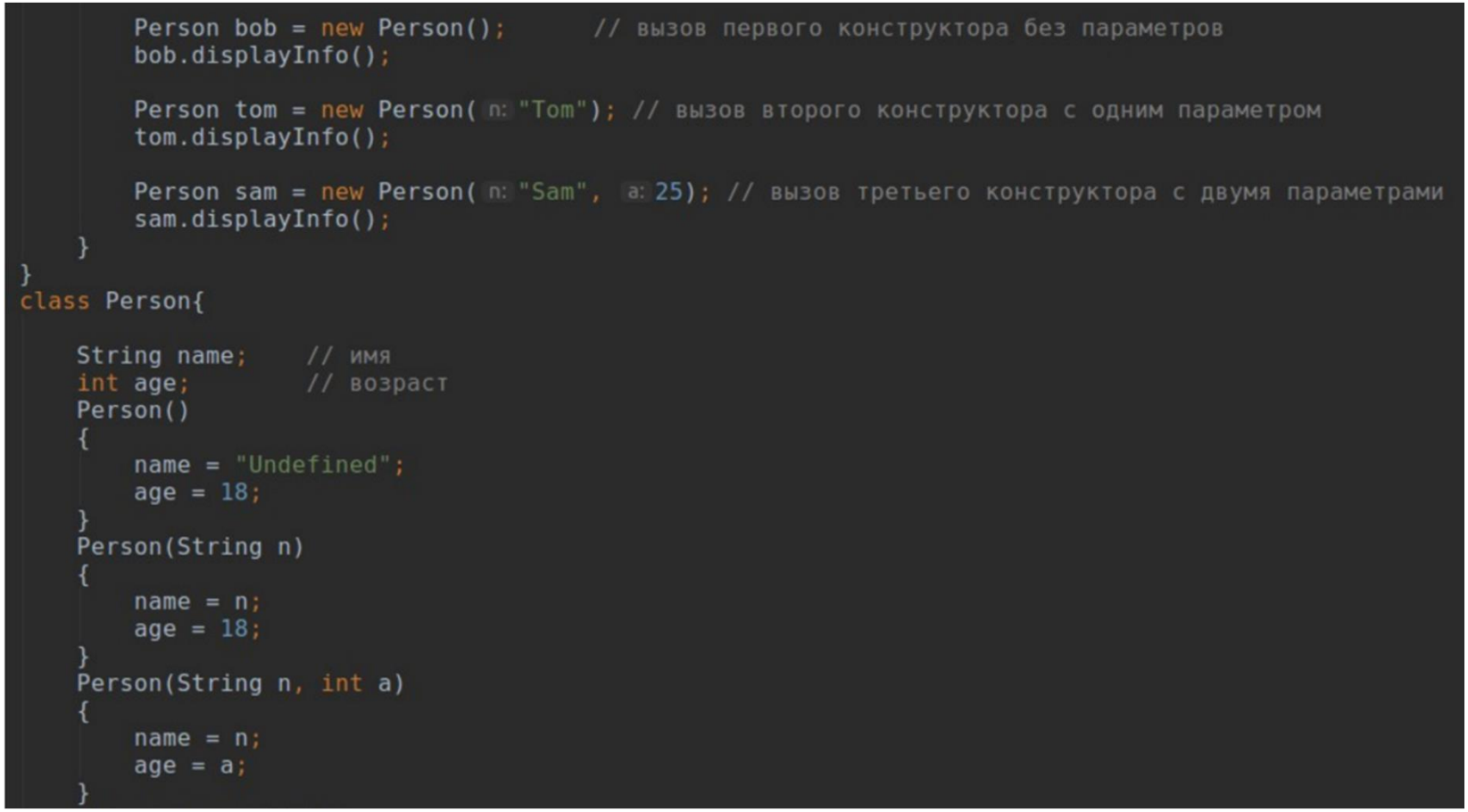
```
Person tom = new Person(); // создание объекта


// изменяем имя и возраст
tom.name = "Tom";
tom.age = 34;
tom.displayInfo();
```

Конструкторы класса

Если необходимо, что при создании объекта производилась какая-то логика, например, чтобы поля класса получали какие-то определенные значения, то можно определить в классе свои конструкторы.

Их может быть несколько.

Конструкторы класса



```
        Person bob = new Person();      // вызов первого конструктора без параметров
        bob.displayInfo();

        Person tom = new Person( n: "Tom"); // вызов второго конструктора с одним параметром
        tom.displayInfo();

        Person sam = new Person( n: "Sam",  a: 25); // вызов третьего конструктора с двумя параметрами
        sam.displayInfo();
    }
}
class Person{

    String name;    // имя
    int age;        // возраст
    Person()
    {
        name = "Undefined";
        age = 18;
    }
    Person(String n)
    {
        name = n;
        age = 18;
    }
    Person(String n, int a)
    {
        name = n;
        age = a;
    }
```

Ключевое слово this

Ключевое слово this представляет ссылку на текущий экземпляр класса. Через это ключевое слово мы можем обращаться к переменным, методам объекта, а также вызывать его конструкторы.

```
Person(String name)
{
    this(name, age: 18);
}
Person(String name, int age)
{
    this.name = name;
    this.age = age;
}
```

Ключевое слово this

Во втором конструкторе параметры называются так же, как и поля класса. И чтобы разграничить поля и параметры, применяется ключевое слово this:

Так, в данном случае указываем, что значение параметра name присваивается полю name.

Кроме того, у нас два конструктора, которые выполняют идентичные действия: устанавливают поля name и age. Чтобы избежать повторов, с помощью this можно вызвать один из конструкторов класса и передать для его параметров необходимые значения:

Ключевое слово this

Во втором конструкторе параметры называются так же, как и поля класса. И чтобы разграничить поля и параметры, применяется ключевое слово this:

Так, в данном случае указываем, что значение параметра name присваивается полю name.

Кроме того, у нас два конструктора, которые выполняют идентичные действия: устанавливают поля name и age. Чтобы избежать повторов, с помощью this можно вызвать один из конструкторов класса и передать для его параметров необходимые значения:

Пакеты

Как правило, в Java классы объединяются в пакеты. Пакеты позволяют организовать классы логически в наборы. По умолчанию java уже имеет ряд встроенных пакетов, например, java.lang, java.util, java.io и т.д. Кроме того, пакеты могут иметь вложенные пакеты.

Организация классов в виде пакетов позволяет избежать конфликта имен между классами. Принадлежность к пакету позволяет гарантировать однозначность имен.

Чтобы указать, что класс принадлежит определенному пакету, надо использовать директиву package, после которой указывается имя пакета:

package название_пакета;

Импорт пакетов и классов

Если нам надо использовать классы из других пакетов, то нам надо подключить эти пакеты и классы. Исключение составляют классы из пакета java.lang(например, String), которые подключаются в программу автоматически.

Например, класс Scanner находится в пакете java.util, поэтому мы можем получить к нему доступ следующим способом:

```
package study;

import java.util.Scanner; // импорт класса Scanner
```

Импорт пакетов и классов

В примере выше мы подключили только один класс, однако пакет java.util содержит еще множество классов. И чтобы не подключать по отдельности каждый класс, мы можем сразу подключить весь пакет:

```
import java.util.*; // импорт всех классов из пакета java.util
```

Возможна ситуация, когда мы используем два класса с одним и тем же названием из двух разных пакетов, например, класс Date имеется и в пакете java.util, и в пакете java.sql. И если нам надо одновременно использовать два этих класса, то необходимо указывать полный путь к этим классам в пакете:

```
java.util.Date utilDate = new java.util.Date();
java.sql.Date sqlDate = new java.sql.Date();
```

Импорт пакетов и классов

В примере выше мы подключили только один класс, однако пакет java.util содержит еще множество классов. И чтобы не подключать по отдельности каждый класс, мы можем сразу подключить весь пакет:

```
import java.util.*; // импорт всех классов из пакета java.util
```

Возможна ситуация, когда мы используем два класса с одним и тем же названием из двух разных пакетов, например, класс Date имеется и в пакете java.util, и в пакете java.sql. И если нам надо одновременно использовать два этих класса, то необходимо указывать полный путь к этим классам в пакете:

```
java.util.Date utilDate = new java.util.Date();
java.sql.Date sqlDate = new java.sql.Date();
```

Практика

Создайте класс студента с полями:

- имя
- группа
- оценка за диплом

В цикле создайте массив из 14-ти студентов.

Практика

Создайте конструктор класса студент и используйте его в цикле. В конструкторе задайте всем студентам одинаковую группу. И установите каждому студенту различную оценку по 10-ти бальной системе с помощью класса Random.

Практика

Создайте метод класса студент, который будет выводить всю информацию о студенте.

Выведите информацию о всех отличниках (9-10 баллов за диплом) в консоль.

Практика

Создайте класс кота с полями:

- имя
- возраст
- насыщение кота (количество корма)

Создайте метод “кормежка”. Входным параметром укажите количество корма. Возвращаемое значение:

- true - кот наелся
- false - кот не наелся