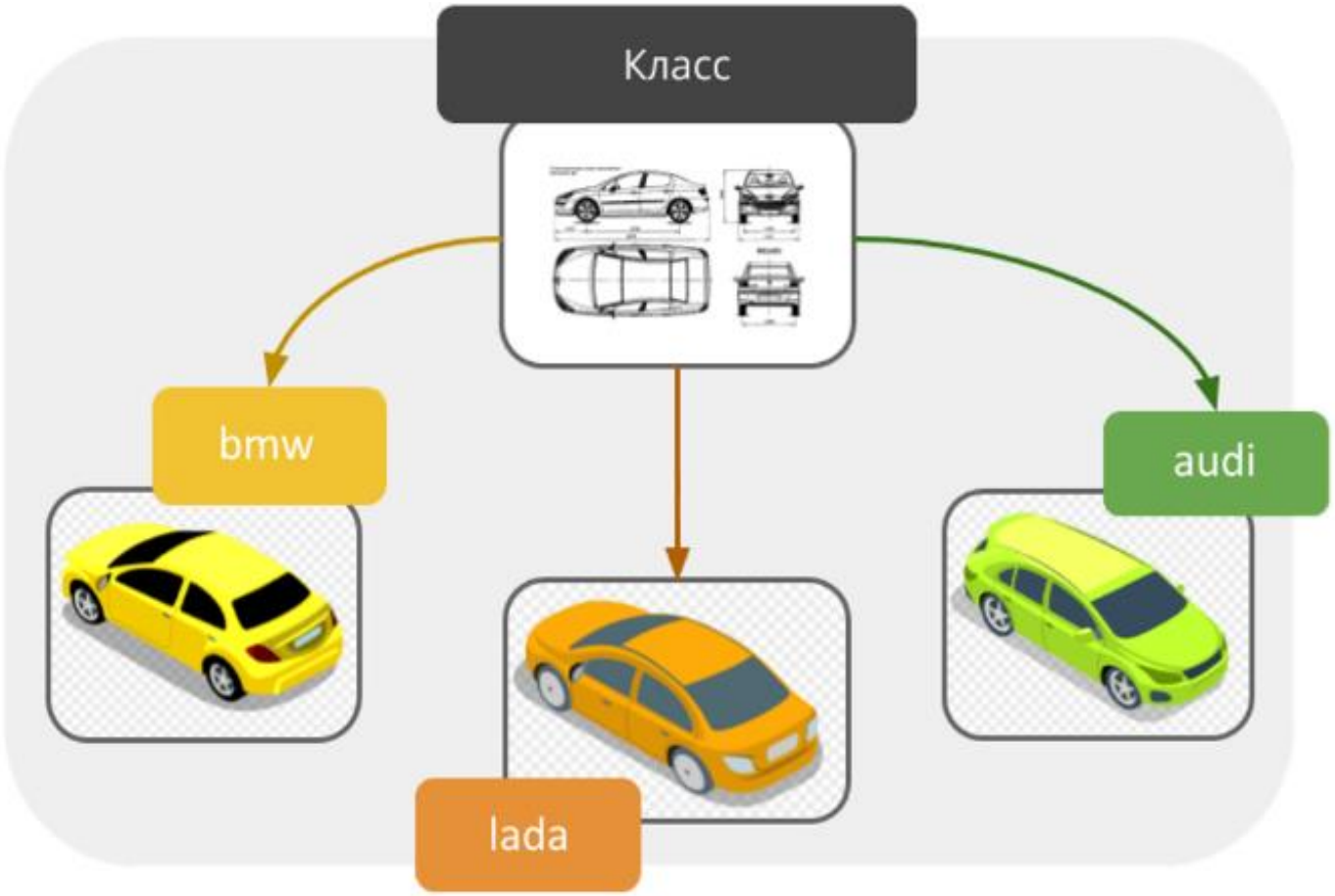
Тема 2
Классы в Java

Макейчик Е.Г.

Классы и объекты



Классы и объекты

Пример описания класса

```
public class Point {
    int x, y;   // Экземплярные переменные
    Point() {   // Конструктор
        x = 0; y = 0;
    }
    Point (int x1,int y1) { // Конструктор
        x = x1; y = y1;
    }
    int getX() { return x; }
    int getY() { return y; }
}
```

Пример создания экземпляра класса и ссылки на него

```
Point p1 = new Point();
Point p2 = p1;
```

Классы и объекты

Пример описания класса

```java
public class Box {
    int width; // Ширина коробки
    int height; // Высота коробки
    int depth; // Глубина коробки

    // Конструктор
    Box(int w, int h, int d) {
        width = w;
        height = h;
        depth = d;
    }
    // Вычисление объёма коробки
    int getVolume() {
        return width * height * depth;
    }
}
```

Статические поля

Объявление: **static <type> <name>**

Обращение: **<classname>.<varname>**

- создаются в единственном экземпляре;
- существуют вне зависимости от объектов класса;
- создаются JVM в момент первого обращения к классу;
- допускают обращение до создания объектов класса.

Статические методы

Статические методы:

- могут вызывать только другие статические методы;
- должны обращаться только к статическим переменным;
- внутри статических методов нельзя использовать ссылки this и super.

Статический блок

Объявление

```
static
{
    // ...
}
```

Статический блок кода выполняется один раз при первоначальной загрузке класса.

```
public class Fruits {
    static String[] list = new String[3];

    static {
        list[0] = "Apple";
        list[1] = "Orange";
        list[2] = "Banana";
    }
    public static void main(String[] args) {
        for(int i = 0; i < list.length; i++)
            System.out.println(list[i]);
    }
}
```

Использование this

Указатель на экземпляр this может использоваться для вызова одного конструктора из другого.

```
public class Rectangle
{
    private int x, y, width, height;
    public Rectangle() {
        this( x: 0,  y: 0,  width: 0,  height: 0);
    }
    public Rectangle(int width, int height) {
        this( x: 0,  y: 0, width, height);
    }
    public Rectangle(int x, int y, int width, int height) {
        this.x = x; this.y = y;
        this.width = width; this.height = height;
    }
}
```

Управление доступом

В Java используется похожая на C++ модель разграничения доступа к элементам классов.

Отличия:

- Спецификатор доступа пишется перед каждым элементом класса;
- Отсутствие спецификатора перед элементом тоже задает уровень доступа.

Управление доступом

Правила доступа

Пользователь	Private	Отсутствие	Protected	Public
Один и тот же класс	Yes	Yes	Yes	Yes
Подкласс класса того же пакета	No	Yes	Yes	Yes
Класс пакета	No	Yes	Yes	Yes
Подкласс класса из другого пакета	No	No	Yes	Yes
Класс другого пакета	No	No	No	Yes

Пакеты

В первой строке файла может быть 1 необязательный оператор package.

В следующих строках может быть 1 или несколько необязательных операторов import- Далее идут описания классов и интерфейсов.

Среди классов, описанных в одном файле, только один может быть объявлен с модификатором public.

Свойства пакетов:

- Каждый пакет предоставляет уникальное пространство имен для своего содержимого.
- Допустимы вложенные пакеты.

Пакеты

Разработка пакета для работы с простыми числами Prime

```java
package Prime;

public class Prime {
    public static boolean testPrime(int value) {
        boolean isprime = true;
        for(int i = 2; i * i <= value; i++)
            if(value % i == 0) {
                isprime = false;
                break;
            }
        return isprime;
    }
    public static int nextPrime(int begin) {
        while(!testPrime( value: ++begin));
        return begin;
    }
    public static int nPrime(int begin,int num) {
        while(num > 0) {
            while(!testPrime( value: ++begin));
            num--;
        }
        return begin;
    }
}
```

```java
package Prime;

public class Factorization
{
    public static void factorIt(int n) {
        for(int i = 2; i * i <= n; i++) {
            if(n % i== 0) {
                System.out.print(" " + i);
                n/=i;
            }
        }
        System.out.println();
    }
}
```

Пакеты

Разработка пакета для работы с простыми числами Prime

```java
import Prime.*;

public class Demo {
    public static void main(String[] args) {
        for(int i = 1; i < 15; i++)
            System.out.println(Prime.nPrime( begin: 1, num: i));
        Factorization.factorIt( n: 123456);
    }
}
```

```
.
├── Demo.class
├── Demo.java
└── Prime
    ├── Factorization.class
    ├── Factorization.java
    ├── Prime.class
    └── Prime.java
```

Методы с переменным числом аргументов

Для указания аргумента переменной длины используют три точки (...).

```
public static void vaTest(int... v)
```

Эта синтаксическая конструкция указывает компилятору, что метод vaTest() может вызываться с нулем или более аргументов.

В результате v неявно объявляется как массив типа int[ ]. Таким образом, внутри метода vaTest() доступ к v осуществляется с использованием синтаксиса обычного массива.

Методы с переменным числом аргументов

```java
public class VarArgs {
    // Здесь vaTest() использует аргументы переменной длины
    public static void vaTest(int... v) {
        System.out.print("Количество аргументов: " + v.length + " Содержимое: ");
        for (int x : v) {
            System.out.print(x + " ");
        }
        System.out.println();
    }
}
```

```java
public class Main {
    public static void main(String args[]) {
        // Создание массива для хранения аргументов
        int n1[] = {10};
        int n2[] = {1, 2, 3};
        int n3[] = {};
        PassArray.vaTest( ...v: n1); // 1 аргумент
        PassArray.vaTest( ...v: n2); // 3 аргумента
        PassArray.vaTest( ...v: n3); // без аргументов
    }
}
```

Классы обертки/оболочки

Примитивный тип	Класс обертка	Аргументы конструктора
byte	Byte	byte or String
int	Integer	int or String
short	Short	short or String
long	Long	long or String
float	Float	float, double, or String
double	Double	double or String
char	Character	char
boolean	Boolean	boolean or String

Классы обертки/оболочки

```java
public class WrapperDemo {
    public static void main(String[] args) {
        Integer integer1 = Integer.valueOf( s: "6");
        Integer integer2 = Integer.valueOf( i: 6);
        // Преобразовывает 101011 к 43
        Integer integer3 = Integer.valueOf( s: "101011",  radix: 2);
        System.out.println(integer1);
        System.out.println(integer2);
        System.out.println(integer3);
    }
}
```

```
6
6
43
```

Нумерованные типы

В простейшем виде механизм нумерованных типов сводится к поддержке списка именованных констант, определяющих новый тип данных. Объект нумерованного типа может принимать лишь значения, содержащиеся в списке. Таким образом, нумерованные типы предоставляют возможность создавать новый тип данных, содержащий лишь фиксированный набор допустимых значений.

Нумерованные типы

```
public enum Transport {
    CAR, TRUCK, AIRPLANE, TRAIN, BOAT
}
```

```
public class Main {
    public static void main(String args[]) {
        Transport tp;
        System.out.println("Here are all Transport constants");
        // Использование метода values()
        // Получение массива констант типа Transport
        Transport allTransports[] = Transport.values();
        for (Transport t : allTransports) {
            System.out.println(t);
        }
            System.out.println();
        // Использование метода valueOf()
        // Получение константы AIRPLANE
        tp = Transport.valueOf( name: "AIRPLANE");
        System.out.println("tp contains " + tp);
    }
}
```