Лекция 4

Принципы объектно-ориентированного программирования на Java

Java — это объектно-ориентированный язык программирования

В основе языка Java лежат понятия класса и объекта

В чем же отличие процедурного языка от объектно-ориентированного?

Разберем это отличие на примере...

Пример процедурного подхода к программированию

```
import java.util.*;
import static java.lang.Math.*;

public class Main {

  public static void main(String[ ] args) {
     double z;
     double T;
     double E;
     z = 2.0;
     T = 293.15;
     E = 610.78*pow(10, 7.63*(T-273.15)/(T-31.25));

    // Печать значений
    System.out.printf(Locale.US, "%.0f\t%.1f\t%.1f\n", z, T, E);
  }
}
```

Классы

Класс — это именованная совокупность полей, содержащих значения данных и методы для работы с этими значениями.

Например, создадим класс, имеющий *поля* для хранения значения высоты над земной поверхностью и температуры воздуха, а также *метод* для расчета давления насыщения водяного пара.

Назовем этот класс Air

Пример класса

```
import static java.lang.Math.*;

public class Air {
  // Поля класса
  private double z;
  private double T;

  // Конструктор класса (задает начальные значения полям)
  public Air(double z, double T) {
    this.z = z;
    this.T = T;
  }

  // Методы класса
  public double getHeight() {
    return z;
  }

  public double getTemperature() {
    return T;
  }

  public double getSaturatedVapourPressure() {
    return 610.78*pow(10, 7.63*(T-273.15)/(T-31.25));
  }
}
```

Типы данных

Создание нового *класса* ведет к появлению нового *типа* данных.

Помните простые *типы* данных?

Объекты

Например, простые типы:

```
double T;
int i;
boolean flag;
```

Создавая класс Air мы по сути создаем новый тип данных:

Air

И, следовательно, можно создавать переменные такого типа — они называются объектами:

```
Air air1;
Air air2;
```

Пример создания объекта

```
import java.util.*;

public class Main {

  public static void main(String[ ] args) {
    Air air = new Air(2.0, 298.15);

    // Печать значений
    System.out.printf(Locale.US, "%.0f\t%.1f\t%.1f\n",
            air.getHeight(),  air.getTemperature(), air.getSaturatedVapourPressure());
  }
}
```

Классы и объекты

Каждый **объект** представляет собой экземпляр какого-либо **класса**.

Объекты содержат данные и методы обработки этих данных.

Так, по аналогии, конкретный автомобиль марки ВАЗ 2101 является **объектом**, а техническая документация на ВАЗ 2101 – это описание **класса** ВАЗ 2101

Модификаторы классов

public — открытый класс, который может быть переопределен и расширен

final — класс не может содержать подклассов

Например,

```
public class Air {
...
}
```

Модификаторы переменных (1)
Область видимости

public — поля, доступные для всех методов

protected — защищенные поля, доступ только из методов того же класса и подклассов

private — закрытые поля, доступны только для методов того же класса

Например,

public **int** i = 12;

Модификаторы переменных (2)
Способ использования

static — статическая переменная, значение этого поля является общим для всех объектов этого класса

final — константа, значение этого поля задается один раз при объявлении поля и не может изменяться

Например,

public static final double Rd = 287.05;

Модификаторы методов (1)
Область видимости

public — методы, доступные для всех методов

protected — защищенные методы, доступ только из методов того же класса и подклассов

private — закрытые методы, доступны только для методов того же класса

Например,

```
public double getTemperature() {
...
}
```

Модификаторы методов (2)
Способ использования

static — метод относится ко всему классу, а не к конкретному экземпляру класса

final — метод не может быть переопределен в подклассах

Например,

```
public static double sin(double x) {
...
}
```

Вопросы?

http://www.chukin.ru/edu/java/

chukin@rshu.ru