Содержание

1 лекция Понятия класса, объекта. Инкапсуляция.

Определения

Объект (object) — самоописывающая структура данных, обладающая внутренними состояниями и способная обрабатывать передаваемые ей сообщения.

Инкапсуляция (incapsulation) — один из основных принципов ООП, заключающийся в том, что доступ к внутреннему состоянию объекта осуществляется только через механизм передачи сообщений.

Класс (class) — тип данных, значениями которого являются объекты, имеющие сходное внутреннее состояние и обрабатывающие одинаковый набор сообщений.

Замечание: объекты являются самоописывающими, так как содержат информацию о классе, к которому они принадлежат.

Замечание: класс можно рассматривать как шаблон для порождения объектов.

Замечание: объекты называют также **экземплярами класса** (class instanses).

Пример:

```
// Класс Point
(x, y, z) // координаты - это экземплярные поля
get_r(double x, double y, double z) // экземплярный метод
```

Типы классов:

1. Публичный
   Один на файл.
2. Непубличный
   В одном файле может быть больше одного.

Замечание: файл с исходным кодом описывающим публичный класс, именовывается так же, как и `public` (публичный) класс, с точностью до регистра.

Структура публичного класса

1. Непубличные классы Эти классы видны только внутри этого файла (при объявлении непубличных классов можно не ставить модификатор `public`)
2. Члены класса
   - Экземплярные поля
     Хранение внутренних состояний объекта.
   - Статические поля
     Хранение данных, общих для всех объектов класса. Модификатор `static`.
   - Экземплярные методы
     Выполняют обработку передаваемых объекту сообщений.
   - Экземплярные конструкторы
     Инициализирует только что созданный объект (Метод, который выполняется в момент инициализации объекта).
   - Статический конструктор (набор статических блоков)
     Инициализирует статические поля класса.
   - Статический метод
     Выполняют действия, для которых не нужен доступ к конкретному объекту класса.
   - Вложенные классы
     Объекты, необходимые для реализации данного класса.

Примеры

1. Не нарушаем принцип инкапсуляции

- Main.java:
```
public class Main {
    public static void main(String[] args) {
        Point A = new Point("Tochka A");
        A.setCoords(1.0, 1.0, 34.0);
    }
}
```
- Point.java:
```
public class Point {
    private double x;
    private double y;
    private double z;
    public Point(String Name) { // конструктор
        this.name = Name;
    }
    public void setCoords(double inX, double inY,
     ↪  double inZ) {
        this.x = inX;
        this.y = inY;
        this.z = inZ;
    }
}
```

2. Нарушаем
   - Main.java:
```
public class Main {
    public static void main(String[] args) {
        Point A = new Point("Tochka A");
        A.x = 1.0;
        A.y = 1.0;
        A.z = 34.0;
    }
}
```
   - Point.java:
```
public class Point {
    public double x;
    public double y;
    public double z;
    public Point(String Name) { // конструктор
        this.name = Name;
    }
```

```
}
```

2 лекция Статические методы, статические поля.

Определения статического поля, экземлярного метода

Статическое поле (static field) — это такое поле, принадлежащее некотрому классу, значение которого разделяется всеми объектами этого класса.

Пример:

```
class Point {
    public int x, y; // координаты точки
    public static int count; // общее количество точек
};
```

Экземплярный метод (instance method) — подпрограмма (функция), осуществляющая обработку переданного объекту сообщения.

- *Экземплярный метод* передает объекту сообщения.
- *Экземплярный метод* имеет доступ к внутреннему состоянию объекта (может читать/изменять значения экземплярных полей).

Пример:

```
class Person {
    public String name;
    public int yearOfBirth;
    private String address;
};
```

`name`, `yearOfBirth`, `address` — экземплярные поля. `public`, `private` — модификаторы.

Виды модификаторов

1. `private`
   Доступ разрешен только из тела класса.
2. Без модификатора
   Доступ разрешен для самого класса и классов из того же пакета.
3. `protected`
   Доступ разрешен для самого класса, для классов из того же пакета, а также наследников класса.
4. `public`
   Доступ возможен откуда угодно.

Перегрузка методов

Конструктор не указан. Создается конструктор по умолчанию:

```
public class Cat {
    public String name;
    public int age;
};
```

Явно указан конструктор по умолчанию:

```
public class Cat {
    public String name;
    public int age;
    public Cat() {}
};
```

Перегрузка конструктора:

```
public class Cat {
    public String name;
    public int age;
    public Cat(String name, int age) {
        this.name = name;
        this.age = age;
    }
    public Cat() {}
};
```

`String name, int age` — **структура метода** (method structure).

Примеры создания объекта класса `Cat`:

```
Cat A = new Cat();
Cat B = new Cat("Meow", 5);
```

Статические методы

Статический метод (static method) — метод, не имеющий доступа к внутреннему состоянию этого объекта. Другими словами, статический метод может обратиться только к статическим переменным и методам.

`private static` vs `public static`:

- `public static`
  Статическое поле можно определить через любой объект класса или имя класса.
- `private static`
  Можно определить только внутри класса.

Примеры:

```
public class Point {
    private double x;
    private double y;
    private static int n;
    public static int val;
    public Point() {
        this.n = 10;
        this.val = 100;
    }
    public void setCoords(double varX, double varY) {
        this.x = varX;
        this.y = varY;
    }
    public double getN() {
        return this.n;
    }
    public void setN() {
        this.n = 100;
    }
};
```

```
public class Main {
    public static void main(String[] args) {
        Point PointA = new Point(); // n = 10, val = 100
        PointA.n = 242; // ошибка, т.к. поле private
        Point.n = 100; // ошибка, т.к. поле private
        Point.val = 200; // верно, обращение к static полю
         ↪  через имя класса
        // n = 10, val = 100

        Point PointB = new Point(); // n = 10, val = 100
        PointA.setN(); // n = 100, val = 100
    }
};
```

Статические блоки

Статический блок (static-блоки) — код, расположенный в статическом блоке, будет выполнен во время запуска программы, или при первой загрузке класса, еще до того, как этот класс будет использоваться в программе (т.е. до создания его экземляров, вызова статических методов и обращения к ним и т.д.).

```
public class A {
    static Date timeStart; // время запуска программы
    Date timeStartObj; // время инициализации объекта
    static {
        timeStart = new Date();
    }
    public A() {
        timeStartObj = new Date();
    }
};
```

```
public class Main {
    public static void main(String[] args) {
        A a = new A(); // timeStart != timeStartObj
    }
};
```