www.performance-lab.ru

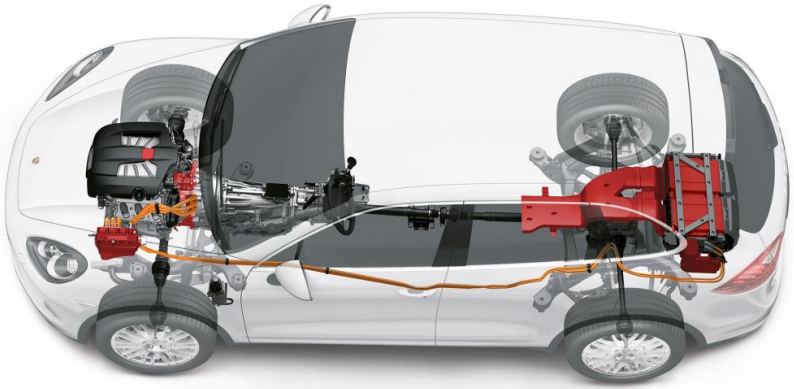
План лекции 2

1. Пакеты, классы, статические классы, перечисления.
2. Наследование, особенности наследования, ООП в целом.
3. Абстрактные классы, интерфейсы, методы по умолчанию.
4. Модификаторы, аннотации (введение).
6. Ввод/Вывод.
7. Исключения, обработка исключений, интерфейс Closable.

Определения

Класс – это способ описания сущности, определяющий состояние и поведение, зависящее от этого состояния, а также правила для взаимодействия с данной сущностью (контракт).

Объект (экземпляр) – это отдельный представитель класса, имеющий конкретное состояние и поведение, полностью определяемое классом.



Классы

```
public class Lrn {
    private bool learning = false;

    /* блок инициализации, не статический */
    { System.out.println("non-static"); }

    /* конструктор */
    public Lrn(bool learning){
        this.learning = true;
    }

    public finalize(){ } //Деструктор
}
```

Пакеты

```
package com.pflb.learning;

import java.util.Arrays;
Import javafx.scene.*;

import static java.lang.Math.sqrt;

public class Lrn { }
```

Структура путей:

com/pflb/learning/Lrn.java

com.pflb.learning/Lrn.java

Модификаторы доступа

`public` – доступен из любого места

`protected` – доступен в пределах пакета, текущего класса и классов-наследников.

`private` – в пределах текущего класса.

<без модификатора> - доступен в пределах пакета.

Статические методы

```
class Parent {
    public static void staticMethod(){
        System.out.println("parent");
}}

class Child extends Parent{
    public static void staticMethod(){
        System.out.println("child");
}}

class Main {
    public static void main(String args[]){
        Parent c = new Child(); c.staticMethod(); //parent
}}
```

Статические классы

```
public class Outer {
    public static class Inner {//доступ через Outer.Inner
        public Inner {...}
    }
}
public class Test {
    /* статический блок инициализации*/
   static {System.out.println("static"); }
}
```

Модификатор final

- Может быть помечен класс, метод, переменная, поле класса.
- Если класс объявлен `final`, от него невозможно унаследоваться.
- Если метод объявлен `final`, его невозможно переопределить в наследниках.
- Если переменная объявлена `final`, значение ей можно присвоить единожды.
- Если в final переменной ссылочный тип (например массив), то менять его содержимое разрешается.
- Поля могут быть объявлены как `static final`, такие поля считаются константами (напр. `Integer.MAX_VALUE`).

Наследование

```
public class Cat extends Animal {

    Cat(String name) {
        super("cat1"); // Вызов конструктора родителя
    }

    @Override
    public void oldMethod() { /* ... */ }

    public void newMethod() {
        super.oldMethod();
    }
}
```

Особенности наследования

- Наследуемый класс автоматически получает поля и методы родителя (в зависимости от модификаторов доступа)
- Наследоваться можно только от одного класса (в отличие от реализации интерфейсов)
- Поля и методы можно переопределять
- Конструктор суперкласса можно вызвать только в начале конструктора наследника
- Если не указано иного, любой класс наследуется от класса Object
- У класса `Object` среди прочих есть методы `equals(Object obj)` и `hashcode()`
- Существует требование на согласованность вышеуказанных методов

Абстрактные классы, интерфейсы

```
public abstract class Shape {
    String type;

    public abstract double getArea();

    public String getType() { return this.type; }
}

public interface Geom {
    double getArea();

    default double getPerimeter { return Double.NaN; }
}
```

Перечисления

```
public enum DayOfWeek {
    MONDAY,
    TUESDAY
    /* ... */
}

DayOfWeek day = DayOfWeek.MONDAY;
DayOfWeek allDays[] = DayOfWeek.values();
DayOfWeek vday = DayOfWeek.valueOf("MONDAY");
int pos = vday.ordinal();
```

PerformanceLab

Обеспечение производительности ИТ

Спасибо за внимание!