

Начало 3 лабы. Часть 1: «Рисование»

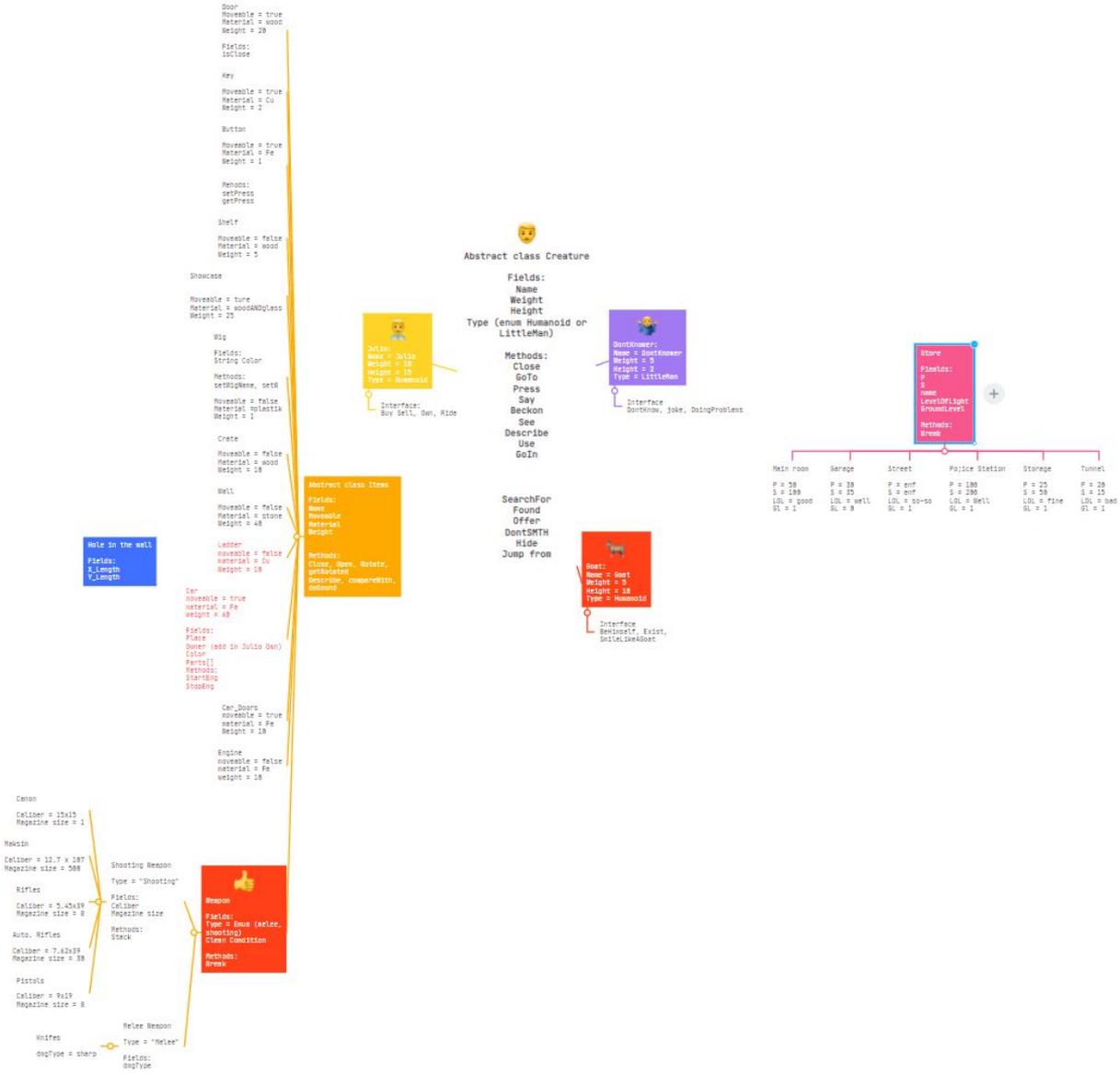
Презентацию подготов... Нахер.
Эту.Часть.

А что от нас хотят?

Порядок выполнения работы:

1. Доработать объектную модель приложения.
2. Перерисовать диаграмму классов в соответствии с внесёнными в модель изменениями.
3. Согласовать с преподавателем изменения, внесённые в модель.

Вот это.



А теперь в порядке живой очереди.

Кто такой этот ваш абстрактный класс

```
abstract class Human{  
    double height;  
    double weight;  
  
    public void drink(){  
    public void eat(String what){  
    }  
}
```



```
class Vasya extends Human{  
    height = 200;  
    weight = 200;  
  
    public void drink(){  
        System.out.println("Vasya drink");  
    }  
  
    public void eat(String what){  
        System.out.println("Vasya eat" + what);  
    }  
}
```

```
class Petya extends Human{  
    height = 100;  
    weight = 100;  
  
    public void drink(){  
        System.out.println("Petya drink");  
    }  
  
    public void eat(String what){  
        System.out.println("Petya " + what);  
    }  
}
```

А теперь будем знакомиться с полубратиком абстрактного класса

```
interface policeman_role{  
public void arrest(String who){}  
}
```

```
Abstract class Human{  
double height;  
double weight;  
  
public void drink(){  
public void eat(String what){  
}
```

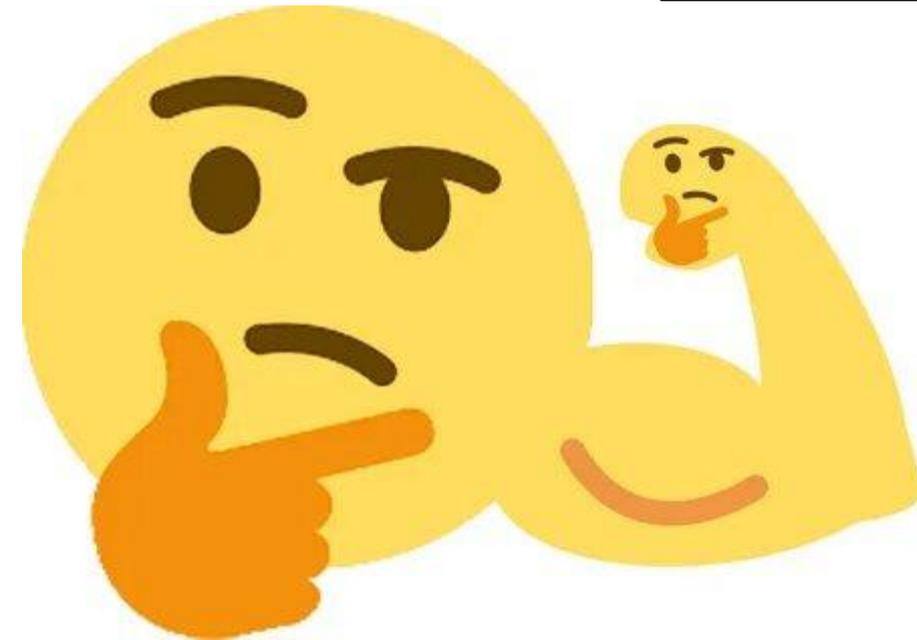
```
interface doctor_role{  
public void heal(){}  
}
```

```
class Vasya extends Human implements policeman_role{  
height = 200;  
weight = 200;  
  
public void drink(){  
System.out.println("Vasya drinks");  
}  
  
public void eat(String what){  
System.out.println("Vasya eats" + what);  
}  
public void arrest(String who){  
System.out.println("Vasya aressts" + who);  
}  
}
```

```
class Petya extends Human implements doctor_role{  
height = 100;  
weight = 100;  
  
public void drink(){  
System.out.println("Petya drinks");  
}  
  
public void eat(String what){  
System.out.println("Petya " + what);  
}  
public void heal(){  
System.out.println("Vasya heals someone");  
}  
}
```

Почему же мы тогда не можем всё засунуть в абстрактный класс?

```
abstract class Human{  
    double height;  
    double weight;  
  
    public void drink(){}  
    public void eat(String what){}  
    public void arrest(String who){}  
    public void heal(){}  
}
```



В чём же плюс таких конструкций?

МНОЖЕСТВЕНН
ОЕ
НАСЛЕДОВАНИ

```
interface policeman_role{  
    public void arrest(String who){}  
}
```

```
Abstract class Human{  
    double height;  
    double weight;  
  
    public void drink(){  
    public void eat(String what){  
    }
```

```
interface doctor_role{  
    public void heal(){  
    }
```

```
class Vasya extends Human implements policeman_role{  
    height = 200;  
    weight = 200;  
  
    public void drink(){  
        System.out.println("Vasya drinks");  
    }  
  
    public void eat(String what){  
        System.out.println("Vasya eats" + what);  
    }  
    public void arrest(String who){  
        System.out.println("Vasya aressts" + who);  
    }  
}
```

```
class Petya extends Human implements doctor_role{  
    height = 100;  
    weight = 100;  
  
    public void drink(){  
        System.out.println("Petya drinks");  
    }  
  
    public void eat(String what){  
        System.out.println("Petya " + what);  
    }  
    public void heal(){  
        System.out.println("Vasya heals someone");  
    }  
}
```

Final and Static

```
Class World{  
public static void boom(){  
System.out.println("Your world was destroyed by stupid programmer");  
}  
}
```

```
final int a = 1;  
a = ... а хрен тебе, не  
сработает
```

Enum..ему...enum или по-русский – enum

```
enum Race{Human, Orc, Elf}
```

```
final String[] Race = new String[]{"Human", "Orc", "Elf"};
```

НО

Есть new-
анс

```
Race race_of_Maks = Race.Orc;
```

```
String race_of_Maks = Race[1];
```

И чё дальше, дальше-то

Всё тривиально. Что тебе нужно, то и используй:-)

Лучший ответ

Массив - это переменная, которая может содержать несколько элементов с индексом, начинающимся с 0, тогда как enum - это определяемый пользователем тип данных, содержащий список элементов, для которых назначается целочисленная константа, начиная с 0. в случае enum числа, начинающиеся с 0, не являются индексами, тогда как в случае массива они являются индексами. Также в случае enum вы можете назначить свои собственные постоянные значения членам, которые могут начинаться или не начинаться с 0 и могут быть или не быть в последовательности.



Ramesh Babu 4 Авг 2015 в 06:20

Значение массива присваивается с помощью оператора =, например. `int a [4] = {1,2,3,4}` и enum - это тип данных, определенный пользователем, и он назначается как enum `days {вс, пн, вт, ср, чт, пт, сб}`



sohan yadav 17 Июнь 2016 в 09:44

Tab P11 с долгоиграющей батареей

АFTER WORK CLUB

время Для тебя

Подробнее

Lenovo

Оба они совершенно разные. Массив - это значение, а перечисление - это тип. array - это набор разных значений, тогда как enum value - это просто одно значение. Массив используется для перебора различных значений с использованием индекса, тогда как перечислению присваивается некоторое атомарное значение и выполняется итерация, чтобы мы могли легко перебирать тип.



Akansha 4 Авг 2015 в 06:19

Основное отличие состоит в том, что массив - это значение, а перечисление - это тип. И одно из основных отличий: мы можем сказать, что массив - это набор других значений (то есть он содержит другие значения, вы можете перебирать их или получать доступ к отдельным по индексу), тогда как значение перечисления - это просто одно атомарное значение. Никаких других значений не содержит.

А теперь давайте-ка порисуем.
Начнём с малого.

Мой текст:

Он закрыл входную дверь изнутри на ключ, после чего подошёл к витрине с париками и нажал скрытую в боковой стенке кнопку. Витрина тотчас же повернулась со скрипом, и за ней обнаружилось четырехугольное отверстие в стене. Господин Жулио шагнул в это отверстие и сказал, поманив рукой:

— Пожалуйста за мной.

Незнайка и Козлик шагнули в отверстие и очутились в складском помещении с полками, на которых лежали деревянные ящики с ружьями, автоматами, пистолетами, кинжалами и другими подобного рода изделиями. Вдоль стены на полу стояли несколько пулемётов на колёсиках и даже одна небольшая



Abstract class Creature

Fields:

Name
Weight
Height

Type (enum Humanoid or LittleMan)

Methods:

Close
GoTo
Press
Say
Beckon
See
Describe
Use
GoIn

SearchFor
Found
Offer
DontSMTH
Hide
Jump from



```

Julio:
Name = Julio
Weight = 10
Height = 15
Type = Humanoid

```

Interface:
Buy Sell, Own, Ride



```

DontKowner:
Name = DontKowner
Weight = 5
Height = 2
Type = LittleMan

```

Interface
DontKnow, joke, DoingProblems



```

Goat:
Name = Goat
Weight = 5
Height = 10
Type = Humanoid

```

Interface
BeHimself, Exist,
SmileLikeAGoat

Crate

```
Moveable = false  
Material = wood  
Weight = 10
```

Wall

```
Moveable = false  
Material = stone  
Weight = 40
```

Ladder

```
moveable = false  
material = Cu  
Weight = 10
```

Abstract class Items

Fields:

```
Name  
Moveable  
Material  
Weight
```

Methods:

```
Close, Open, Rotate,  
getRotated  
Describe, compareToWith,  
doSound
```

Canon

Caliber = 15x15
Magazine size = 1

Maksim

Caliber = 12.7 x 107
Magazine size = 500

Rifles

Caliber = 5.45x39
Magazine size = 8

Auto. Rifles

Caliber = 7.62x39
Magazine size = 30

Pistols

Caliber = 9x19
Magazine size = 8

Knives

dmgType = sharp

Shooting Weapon

Type = "Shooting"

Fields:
Caliber
Magazine size

Methods:
Stack

Melee Weapon

Type = "Melee"

Fields:
dmgType



Weapon

Fields:
Type = Enum (melee,
shooting)
Clean Condition

Methods:
Break

```
Hole in the wall  
  
Fields:  
X_Length  
Y_Length
```

```
Store  
  
Fields:  
P  
S  
name  
LevelOfLight  
GroundLevel  
  
Methods:  
Break
```

Main room	Garage	Street	Police Station	Storage	Tunnel
P = 50	P = 30	P = enf	P = 100	P = 25	P = 20
S = 100	S = 35	S = enf	S = 200	S = 50	S = 15
LOL = good	LOL = well	LOL = so-so	LOL = Well	LOL = fine	LOL = bad
GL = 1	GL = 0	GL = 1	GL = 1	GL = 1	GL = 1

Имеем...

Вопросы к защите лабораторной работы:

1. Принципы объектно-ориентированного программирования SOLID и STUPID.
2. Класс `Object`. Реализация его методов по умолчанию.
3. Особенности реализации наследования в Java. Простое и множественное наследование.
4. Понятие абстрактного класса. Модификатор `abstract`.
5. Понятие интерфейса. Реализация интерфейсов в Java, методы по умолчанию. Отличия от абстрактных классов.
6. Перечисляемый тип данных (`enum`) в Java. Особенности реализации и использования.
7. Методы и поля с модификаторами `static` и `final`.
8. Перегрузка и переопределение методов. Коварианты возвращаемых типов данных.
9. Элементы функционального программирования в синтаксисе Java. Функциональные интерфейсы, лямбда-выражения. Ссылки на методы.



СПАСИБО ЗА ВНИМАНИЕ!!!