«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

Факультет программной инженерии и компьютерной техники

Отчет

По лабораторной работе №4

Вариант 290005.4

Студент:

*Ильин Н. С.*

*Р3110 поток 2.9*

Преподаватель:

*Наумова Н. А.*

Санкт-Петербург, 2023 г.

**Оглавление**

[Задание: 3](#_Toc145408419)

[Выполнение работы: 3](#_Toc145408420)

[Выводы: 29](#_Toc145408421)

# **Задание:**

**Описание предметной области, по которой должна быть построена объектная модель:**

Когда Алиса с Грифоном прибежали. Король и Королева уже сидели на троне, а кругом собралась огромная толпа: пичужки, зверюшки всех пород, не говоря уже о картах всех мастей. Перед судейским троном стоял в цепях под охраной двух солдат - один справа, другой - слева - Червонный Валет.

По правую руку от Короля находился Белый Кролик, с трубой в одной лапке и пергаментным свитком в другой.

А в самом центре судебного зала стоял стол, а на нем красовалось большое блюдо с пирожками, н вид у них был такой аппетитный, что у Алисы прямо слюнки потекли. "Хорошо бы, суд уже кончился и позвали к столу!" - подумала она.

Но так как, судя по всему, до этого было еще далеко, она, чтобы скоротать время, стала рассматривать все окружающее. Хотя Алиса раньше никогда не бывала в суде, она устала про суд в книжках, и ей было очень приятно отметить, что она знает, как тут все - или почти все - называется.

"Вот это судья,- сказала она про себя.- Кто в большом парике, тот и судья".

**Программа должна удовлетворять следующим требованиям:**

1. В программе должны быть реализованы 2 собственных класса исключений (checked и unchecked), а также обработка исключений этих классов.
2. В программу необходимо добавить использование локальных, анонимных и вложенных классов (static и non-static).

**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.
4. Модифицировать программу в соответствии с внесёнными в модель изменениями.

**Отчёт по работе должен содержать:**

1. Текст задания.
2. Диаграмма классов объектной модели.
3. Исходный код программы.
4. Результат работы программы.
5. Выводы по работе.

**Вопросы к защите лабораторной работы:**

1. Обработка исключительных ситуаций, три типа исключений.
2. Вложенные, локальные и анонимные классы.
3. Механизм рефлексии (reflection) в Java. Класс Class.

# **Выполнение работы:**

Исходный код:

#### src/Main.java

  1 │ **import** characters**.\*;**  
  2 │ **import** characters**.Character;**  
  3 │ **import** characters**.**groups**.\*;**  
  4 │ **import** characters**.**movement**.\*;**  
  5 │ **import** exceptions**.\*;**  
  6 │ **import** food**.\*;**  
  7 │ **import** furniture**.\*;**  
  8 │ **import** props**.\*;**  
  9 │ **import** rooms**.\*;**  
  10 │   
  11 │   
  12 │ **public** **class** **Main** **{**  
  13 │  **public** **static** **void** main**(String[]** args**)** **throws** **Exception** **{**  
  14 │  **Queen** queen **=** **new** **Queen(**"Королева"**);**  
  15 │  **King** king **=** **new** **King(**"Король"**);**  
  16 │  **Alice** alice **=** **new** **Alice(**"Алиса"**);**  
  17 │  **Gryphon** gryphon **=** **new** **Gryphon(**"Грифон"**);**  
  18 │  **KnaveOfHearts** knaveOfHearts **=** **new** **KnaveOfHearts(**"Червонный Валет"**);**  
  19 │  **Furniture** kingsThrone **=** **new** **Throne(**"Трон"**);**  
  20 │  **Furniture** queensThrone **=** **new** **Throne(**"Трон"**);**  
  21 │  king**.**sitOn**(**kingsThrone**);**  
  22 │  queen**.**sitOn**(**queensThrone**);**  
  23 │   
  24 │  **int** retryCount **=** 3**;** ***// Количество попыток для прибытия***  
  25 │  ***// Checked exception***  
  26 │  **for** **(Character** character **:** **new** **Character[]** **{**alice**,** gryphon**})** **{**  
  27 │  **for** **(int** i **=** 0**;** i **<** retryCount**;** i**++)** **{**  
  28 │  **try** **{**  
  29 │  character**.**arrive**();**  
  30 │  **break;**  
  31 │  **}** **catch** **(ArrivalException** e**)** **{**  
  32 │  ***// System.err.println(e.getMessage());***  
  33 │  **if** **(**i **<** retryCount **-** 1**)** **{**  
  34 │  **System.**out**.**println**(**character**.**getName**()** **+** " ещё раз пытается найти путь"**);**  
  35 │  **}** **else** **{**  
  36 │  **System.**out**.**println**(**character**.**getName**()** **+** " так и не смог прибежать"**);**  
  37 │  **System.**exit**(**1**);**  
  38 │   
  39 │  **}**  
  40 │  **}**  
  41 │  **}**  
  42 │  **}**  
  43 │   
  44 │   
  45 │   
  46 │  ***// Толпа***  
  47 │  **Crowd** crowd **=** **new** **Crowd();**  
  48 │  ***// Unchecked exception***  
  49 │  **try** **{**  
  50 │  ***// Добавление участников толпы***  
  51 │  crowd**.**addMember**(new** **Crowd.Member(**"Пичужка 1"**));**  
  52 │  crowd**.**addMember**(new** **Crowd.Member(**"Пичужка 2"**));**  
  53 │  crowd**.**addMember**(new** **Crowd.Member(**"Зверушка 1"**));**  
  54 │  crowd**.**addMember**(new** **Crowd.Member(**"Зверушка 2"**));**  
  55 │  crowd**.**addMember**(new** **Crowd.Member(**"Карта Валет"**));**  
  56 │  crowd**.**addMember**(new** **Crowd.Member(**"Карта Туз"**));**  
  57 │  ***// Сбор толпы***  
  58 │  crowd**.**gather**();**  
  59 │  **}**  
  60 │  **catch** **(CantAttendException** e**)** **{**  
  61 │  **System.**err**.**println**(**e**.**getMessage**());**  
  62 │  **System.**err**.**println**(**"В толпу был добавлен запрещённый участник. Выполнение программы завершено."**);**  
  63 │  **System.**exit**(**1**);**  
  64 │  **}**  
  65 │  ***// Охрана***  
  66 │  **KnaveOfHearts.Soldier** guardLeft **=** knaveOfHearts**.new** **Soldier(**"Левый солдат"**);**  
  67 │  **KnaveOfHearts.Soldier** guardRight **=** knaveOfHearts**.new** **Soldier(**"Правый солдат"**);**  
  68 │   
  69 │  knaveOfHearts**.**assignGuardLeft**(**guardLeft**);**  
  70 │  knaveOfHearts**.**assignGuardRight**(**guardRight**);**  
  71 │   
  72 │  ***//=========================================================================***  
  73 │  **WhiteRabbit** whiteRabbit **=** **new** **WhiteRabbit(**"Белый Кролик"**);**  
  74 │  **Courtroom** courtroom **=** **new** **Courtroom(**"Судебный зал"**);**  
  75 │  **Table** table **=** **new** **Table(**"Стол"**);**  
  76 │  **Dish** dish **=** **new** **Dish(**"Большое блюдо с пирожками"**,** 10**);**  
  77 │  dish**.**addFood**(new** **Pie(**"Аппетитный пирожок"**,** "Курица"**));**  
  78 │  dish**.**addFood**(new** **Pie(**"Аппетитный пирожок"**,** "Рыба"**));**  
  79 │  dish**.**addFood**(new** **Pie(**"Аппетитный пирожок"**,** "Картошка"**));**  
  80 │   
  81 │  whiteRabbit**.**standBy**(**king**,** **Direction.**BY\_RIGHT\_HAND**);**  
  82 │  whiteRabbit**.**hold**(new** **Trumpet(**"Труба"**));**  
  83 │  whiteRabbit**.**hold**(new** **Scroll(**"Пергаментный свиток"**));**  
  84 │  courtroom**.**setCenterObject**(**table**);**  
  85 │  table**.**setTopObject**(**dish**);**  
  86 │  dish**.**lookLike**(**"аппетитно"**);**  
  87 │  alice**.**salivateAt**(**dish**);**  
  88 │  alice**.**think**(**"Хорошо бы, суд уже кончился и позвали к столу!"**);**  
  89 │  ***//=========================================================================***  
  90 │   
  91 │  ***// Алиса***  
  92 │  alice**.**observeScene**(new** **String[]{**"судья"**,** "трон"**,** "зверюшки"**,** "пичужки"**,** "карты"**});**  
  93 │  alice**.**think**(**"Вот это судья. Кто в большом парике, тот и судья"**);**  
  94 │  ***// Анонимный класс рассказчика***  
  95 │  **Character** narrator **=** **new** **Character(**"Рассказчик"**)** **{**  
  96 │  @Override  
  97 │  **public** **void** say**(String** speach**)** **{**  
  98 │  **System.**out**.**println**(**speach**);**  
  99 │  **}**  
 100 │  **};**  
 101 │  narrator**.**say**(**"Хотя Алиса раньше никогда не бывала в суде, она устала про суд в книжках, " **+**  
 102 │  "и ей было очень приятно отметить, что она знает, как тут все - или почти все - называется."**);**  
 103 │   
 104 │  **}**  
 105 │   
 106 │ **}**

#### src/rooms/Courtroom.java

  1 │ **package** rooms**;**  
  2 │   
  3 │ **import** furniture**.\*;**  
  4 │   
  5 │ **public** **class** **Courtroom** **extends** **Room** **{**  
  6 │  **public** **Courtroom(String** name**)** **{**  
  7 │  **super(**name**);**  
  8 │  **}**  
  9 │   
 10 │  **public** **void** setCenterObject**(Table** object**){**  
 11 │  **System.**out**.**println**(**"В центре " **+** getName**()** **+** " стоит " **+** object**.**getName**());**  
 12 │  **}**  
 13 │   
 14 │   
 15 │ **}**

#### src/rooms/Room.java

  1 │ **package** rooms**;**  
  2 │   
  3 │ **import** interfaces**.Nameable;**  
  4 │   
  5 │ **public** **abstract** **class** **Room** **implements** **Nameable** **{**  
  6 │  **private** **String** name**;**  
  7 │  **public** **Room(String** name**){**  
  8 │  **this.**name **=** name**;**  
  9 │  **}**  
 10 │  **public** **String** getName**(){**  
 11 │  **return** **this.**name**;**  
 12 │  **}**  
 13 │   
 14 │  @Override  
 15 │  **public** **boolean** equals**(Object** obj**)** **{**  
 16 │  **if** **(this** **==** obj**)** **return** true**;**  
 17 │  **if** **(**obj **==** null **||** getClass**()** **!=** obj**.**getClass**())** **return** false**;**  
 18 │  **Room** room **=** **(Room)** obj**;**  
 19 │  **return** name**.**equals**(**room**.**name**);**  
 20 │  **}**  
 21 │   
 22 │  @Override  
 23 │  **public** **int** hashCode**()** **{**  
 24 │  **return** name**.**hashCode**();**  
 25 │  **}**  
 26 │  @Override  
 27 │  **public** **String** toString**()** **{**  
 28 │  **return** "Room{" **+**  
 29 │  "name='" **+** name **+** '\'' **+**  
 30 │  '}'**;**  
 31 │  **}**  
 32 │ **}**  
 33 │

#### src/props/Dish.java

  1 │ **package** props**;**  
  2 │   
  3 │   
  4 │ **import** food**.Food;**  
  5 │   
  6 │ **import** java**.**util**.Arrays;**  
  7 │   
  8 │ **public** **class** **Dish** **extends** **Prop** **{**  
  9 │  **private** **Food[]** foodItems**;**  
 10 │  **private** **int** itemCount**;**  
 11 │   
 12 │  **public** **Dish(String** name**,** **int** maxItemCount**)** **{**  
 13 │  **super(**name**);**  
 14 │  **this.**foodItems **=** **new** **Food[**maxItemCount**];**  
 15 │  **this.**itemCount **=** 0**;**  
 16 │  **}**  
 17 │   
 18 │  **public** **void** addFood**(Food** foodItem**)** **{**  
 19 │  **if** **(**itemCount **<** foodItems**.**length**)** **{**  
 20 │  foodItems**[**itemCount**]** **=** foodItem**;**  
 21 │  itemCount**++;**  
 22 │  **}** **else** **{**  
 23 │  **System.**out**.**println**(**"На блюде больше нет места."**);**  
 24 │  **}**  
 25 │  **}**  
 26 │   
 27 │  **public** **Food[]** getFoodItems**()** **{**  
 28 │  **return** foodItems**;**  
 29 │  **}**  
 30 │ **}**

#### src/props/Prop.java

  1 │ **package** props**;**  
  2 │   
  3 │ **import** food**.\*;**  
  4 │ **import** interfaces**.Nameable;**  
  5 │   
  6 │ **public** **abstract** **class** **Prop** **implements** **Nameable** **{**  
  7 │  **protected** **String** name**;**  
  8 │   
  9 │  **public** **Prop(String** name**)** **{**  
 10 │  **this.**name **=** name**;**  
 11 │  **}**  
 12 │   
 13 │  **public** **String** getName**()** **{**  
 14 │  **return** **this.**name**;**  
 15 │  **}**  
 16 │  **public** **void** lookLike**(String** description**)** **{**  
 17 │  **System.**out**.**println**(**getName**()** **+** " выглядит " **+** description**);**  
 18 │  **}**  
 19 │  **public** **boolean** equals**(Object** obj**)** **{**  
 20 │  **if** **(this** **==** obj**)** **return** true**;**  
 21 │  **if** **(**obj **==** null **||** getClass**()** **!=** obj**.**getClass**())** **return** false**;**  
 22 │  **Prop** prop **=** **(Prop)** obj**;**  
 23 │  **return** name**.**equals**(**prop**.**name**);**  
 24 │  **}**  
 25 │   
 26 │  **public** **int** hashCode**()** **{**  
 27 │  **return** name**.**hashCode**();**  
 28 │  **}**  
 29 │   
 30 │  **public** **String** toString**()** **{**  
 31 │  **return** "Prop{" **+**  
 32 │  "name='" **+** name **+** '\'' **+**  
 33 │  '}'**;**  
 34 │  **}**  
 35 │ **}**  
 36 │

#### src/props/Scroll.java

  1 │ **package** props**;**  
  2 │   
  3 │ **public** **class** **Scroll** **extends** **Prop** **{**  
  4 │  **private** **String** text **=** "..."**;**  
  5 │  **public** **Scroll(String** name**){**  
  6 │  **super(**name**);**  
  7 │  **}**  
  8 │   
  9 │  **public** **String** getText**()** **{**  
 10 │  **return** text**;**  
 11 │  **}**  
 12 │   
 13 │  **public** **void** setText**(String** text**)** **{**  
 14 │  **this.**text **=** text**;**  
 15 │  **}**  
 16 │ **}**  
 17 │

#### src/props/Trumpet.java

  1 │ **package** props**;**  
  2 │   
  3 │ **public** **class** **Trumpet** **extends** **Prop** **{**  
  4 │  **public** **Trumpet(String** name**){**  
  5 │  **super(**name**);**  
  6 │  **}**  
  7 │   
  8 │  **public** **void** makeSound**(){**  
  9 │  **System.**out**.**println**(**getName**()** **+** " издаёт звук"**);**  
 10 │  **}**  
 11 │ **}**  
 12 │

#### src/interfaces/Nameable.java

 1 │ **package** interfaces**;**  
 2 │   
 3 │ **public** **interface** **Nameable** **{**  
 4 │  **String** getName**();**  
 5 │ **}**

#### src/interfaces/Thinkable.java

 1 │ **package** interfaces**;**  
 2 │   
 3 │ **public** **interface** **Thinkable** **{**  
 4 │  **void** think**(String** thought**);**  
 5 │ **}**

#### src/furniture/Furniture.java

  1 │ **package** furniture**;**  
  2 │   
  3 │ **import** interfaces**.Nameable;**  
  4 │   
  5 │ **public** **abstract** **class** **Furniture** **implements** **Nameable** **{**  
  6 │  **protected** **String** name**;**  
  7 │   
  8 │  **public** **Furniture(String** name**)** **{**  
  9 │  **this.**name **=** name**;**  
 10 │  **}**  
 11 │   
 12 │  **public** **String** getName**()** **{**  
 13 │  **return** **this.**name**;**  
 14 │  **}**  
 15 │   
 16 │  **public** **boolean** equals**(Object** obj**)** **{**  
 17 │  **if** **(this** **==** obj**)** **return** true**;**  
 18 │  **if** **(**obj **==** null **||** getClass**()** **!=** obj**.**getClass**())** **return** false**;**  
 19 │  **Furniture** furniture **=** **(Furniture)** obj**;**  
 20 │  **return** name**.**equals**(**furniture**.**name**);**  
 21 │  **}**  
 22 │   
 23 │  **public** **int** hashCode**()** **{**  
 24 │   
 25 │  **return** name**.**hashCode**();**  
 26 │  **}**  
 27 │   
 28 │  **public** **String** toString**()** **{**  
 29 │  **return** "Furniture{" **+**  
 30 │  "name='" **+** name **+** '\'' **+**  
 31 │  '}'**;**  
 32 │  **}**  
 33 │ **}**

#### src/furniture/Table.java

  1 │ **package** furniture**;**  
  2 │   
  3 │ **import** props**.\*;**  
  4 │ **public** **class** **Table** **extends** **Furniture** **{**  
  5 │  **public** **Table(String** name**){**  
  6 │  **super(**name**);**  
  7 │  **}**  
  8 │  **private** **Prop** topObject**;**  
  9 │  **private** **Prop** bottomObject**;**  
 10 │   
 11 │  **public** **void** setTopObject**(Prop** object**){**  
 12 │  **this.**topObject **=** object**;**  
 13 │  **System.**out**.**println**(**"На " **+** getName**()** **+** " стоит " **+** object**.**getName**());**  
 14 │  **}**  
 15 │  **public** **void** setBottomObject**(Prop** object**){**  
 16 │  **this.**bottomObject **=** object**;**  
 17 │  **System.**out**.**println**(**"Под " **+** getName**()** **+** " стоит " **+** object**.**getName**());**  
 18 │  **}**  
 19 │ **}**

#### src/furniture/Throne.java

 1 │ **package** furniture**;**  
 2 │   
 3 │ **public** **class** **Throne** **extends** **Furniture** **{**  
 4 │  **public** **Throne(String** name**)** **{**  
 5 │  **super(**name**);**  
 6 │  **}**  
 7 │ **}**

#### src/food/Food.java

  1 │ **package** food**;**  
  2 │   
  3 │ **import** interfaces**.Nameable;**  
  4 │ **public** **abstract** **class** **Food** **implements** **Nameable** **{**  
  5 │  **protected** **String** name**;**  
  6 │  **public** **Food(String** name**)** **{**  
  7 │  **this.**name **=** name**;**  
  8 │  **}**  
  9 │  **public** **String** getName**()** **{**  
 10 │  **return** **this.**name**;**  
 11 │  **}**  
 12 │  **public** **int** hashCode**()** **{**  
 13 │  **return** name**.**hashCode**();**  
 14 │  **}**  
 15 │  **public** **boolean** equals**(Object** obj**)** **{**  
 16 │  **if** **(this** **==** obj**)** **return** true**;**  
 17 │  **if** **(**obj **==** null **||** getClass**()** **!=** obj**.**getClass**())** **return** false**;**  
 18 │  **Food** food **=** **(Food)** obj**;**  
 19 │  **return** name**.**equals**(**food**.**name**);**  
 20 │  **}**  
 21 │  **public** **String** toString**()** **{**  
 22 │  **return** "Food{" **+**  
 23 │  "name='" **+** name **+** '\'' **+**  
 24 │  '}'**;**  
 25 │  **}**  
 26 │ **}**  
 27 │

#### src/food/Pie.java

  1 │ **package** food**;**  
  2 │   
  3 │ **import** interfaces**.Nameable;**  
  4 │   
  5 │ **import** javax**.**swing**.\*;**  
  6 │   
  7 │ **public** **class** **Pie** **extends** **Food** **{**  
  8 │  **private** **String** filling**;**  
  9 │  **public** **Pie(String** name**,** **String** filling**){**  
 10 │  **super(**name**);**  
 11 │  **this.**filling **=** filling**;**  
 12 │  **}**  
 13 │   
 14 │  **public** **String** getFilling**()** **{**  
 15 │  **return** **this.**filling**;**  
 16 │  **}**  
 17 │ **}**

#### src/exceptions/ArrivalException.java

 1 │ **package** exceptions**;**  
 2 │   
 3 │ **public** **class** **ArrivalException** **extends** **Exception** **{**  
 4 │  **public** **ArrivalException(String** message**)** **{**  
 5 │  **super(**message**);**  
 6 │  **}**  
 7 │ **}**  
 8 │

#### src/exceptions/CantAttendException.java

 1 │ **package** exceptions**;**  
 2 │   
 3 │ **public** **class** **CantAttendException** **extends** **RuntimeException{**  
 4 │  **public** **CantAttendException(String** message**)** **{**  
 5 │  **super(**message**);**  
 6 │  **}**  
 7 │ **}**  
 8 │

#### src/characters/Alice.java

 1 │ **package** characters**;**  
 2 │   
 3 │ **public** **class** **Alice** **extends** **Character** **{**  
 4 │  **public** **Alice(String** name**){**  
 5 │  **super(**name**);**  
 6 │  **}**  
 7 │ **}**

#### src/characters/Character.java

  1 │ **package** characters**;**  
  2 │   
  3 │ **import** characters**.**movement**.Direction;**  
  4 │ **import** furniture**.Furniture;**  
  5 │ **import** interfaces**.Nameable;**  
  6 │ **import** interfaces**.Thinkable;**  
  7 │ **import** props**.Prop;**  
  8 │ **import** exceptions**.ArrivalException;**  
  9 │   
  10 │ **public** **abstract** **class** **Character** **implements** **Nameable,** **Thinkable** **{**  
  11 │  **protected** **String** name**;**  
  12 │  **private** **Prop** objectInHand**;**  
  13 │   
  14 │  **public** **Character(String** name**)** **{**  
  15 │  **this.**name **=** name**;**  
  16 │  **}**  
  17 │   
  18 │  **public** **String** getName**()** **{**  
  19 │  **return** **this.**name**;**  
  20 │  **}**  
  21 │   
  22 │  **public** **void** standBy**(Character** character**,** **Direction** direction**)** **{**  
  23 │  **System.**out**.**println**(**getName**()** **+** " находится " **+** direction**.**getDirection**()** **+** " " **+** character**.**getName**());**  
  24 │  **}**  
  25 │   
  26 │  ***// checked exception***  
  27 │  **public** **void** arrive**()** **throws** **ArrivalException** **{**  
  28 │  **if** **(Math.**random**()** **<** 0.3**)** **{**  
  29 │  **throw** **new** **ArrivalException(**getName**()** **+** " заблудился пока бежал"**);**  
  30 │  **}**  
  31 │  **System.**out**.**println**(**getName**()** **+** " прибежал"**);**  
  32 │  **}**  
  33 │   
  34 │  **public** **void** salivateAt**(Prop** prop**)** **{**  
  35 │  **System.**out**.**println**(**"У " **+** getName**()** **+** " текут слюнки от " **+** prop**.**getName**());**  
  36 │  **}**  
  37 │   
  38 │  **public** **void** think**(String** thought**)** **{**  
  39 │  **System.**out**.**println**(**"\"" **+** thought **+** "\" - думает " **+** getName**());**  
  40 │  **}**  
  41 │   
  42 │  **public** **void** say**(String** speach**)** **{**  
  43 │  **System.**out**.**println**(**getName**()** **+** "говорит: " **+** speach**);**  
  44 │  **}**  
  45 │   
  46 │  **public** **void** sitOn**(Furniture** furniture**)** **{**  
  47 │  **System.**out**.**println**(**getName**()** **+** " сидит на " **+** furniture**.**getName**());**  
  48 │  **}**  
  49 │   
  50 │  **public** **void** observeScene**(String[]** sceneElements**)** **{**  
  51 │  ***// Локальный класс для элемента сцены***  
  52 │  **class** **SceneElement** **{**  
  53 │  **private** **String** name**;**  
  54 │   
  55 │  **SceneElement(String** name**)** **{**  
  56 │  **this.**name **=** name**;**  
  57 │  **}**  
  58 │   
  59 │  **public** **void** describe**()** **{**  
  60 │  **System.**out**.**println**(Character.this.**name **+** " рассматривает " **+** **this.**name**);**  
  61 │  **}**  
  62 │  **}**  
  63 │   
  64 │  **for** **(String** element **:** sceneElements**)** **{**  
  65 │  ***// Использование локального класса***  
  66 │  **new** **SceneElement(**element**).**describe**();**  
  67 │  **}**  
  68 │   
  69 │  **}**  
  70 │   
  71 │  **public** **void** walkTo**(Direction** direction**)** **{**  
  72 │  **System.**out**.**println**(**getName**()** **+** " идет " **+** direction**.**getDirection**());**  
  73 │  **}**  
  74 │   
  75 │  @Override  
  76 │  **public** **boolean** equals**(Object** obj**)** **{**  
  77 │  **if** **(this** **==** obj**)** **return** true**;**  
  78 │  **if** **(**obj **==** null **||** getClass**()** **!=** obj**.**getClass**())** **return** false**;**  
  79 │  **Character** character **=** **(Character)** obj**;**  
  80 │  **return** name**.**equals**(**character**.**name**);**  
  81 │  **}**  
  82 │   
  83 │  @Override  
  84 │  **public** **int** hashCode**()** **{**  
  85 │  **return** name**.**hashCode**();**  
  86 │  **}**  
  87 │   
  88 │  @Override  
  89 │  **public** **String** toString**()** **{**  
  90 │  **return** "Character{" **+**  
  91 │  "name='" **+** name **+** '\'' **+**  
  92 │  '}'**;**  
  93 │  **}**  
  94 │   
  95 │  **public** **void** hold**(Prop** object**)** **{**  
  96 │  **this.**objectInHand **=** object**;**  
  97 │  **System.**out**.**println**(**getName**()** **+** " держит " **+** **this.**objectInHand**.**getName**());**  
  98 │  **}**  
  99 │ **}**  
 100 │   
 101 │

#### src/characters/Gryphon.java

 1 │ **package** characters**;**  
 2 │   
 3 │ **public** **class** **Gryphon** **extends** **Character** **{**  
 4 │  **public** **Gryphon(String** name**){**  
 5 │  **super(**name**);**  
 6 │  **}**  
 7 │ **}**  
 8 │   
 9 │

#### src/characters/King.java

  1 │ **package** characters**;**  
  2 │   
  3 │ **public** **class** **King** **extends** **Character** **{**  
  4 │  **public** **King(String** name**){**  
  5 │  **super(**name**);**  
  6 │  **}**  
  7 │  **public** **void** command**(String** command**){**  
  8 │  **System.**out**.**println**(**getName**()** **+** " приказывает: " **+** command**);**  
  9 │  **}**  
 10 │ **}**

#### src/characters/KnaveOfHearts.java

  1 │ **package** characters**;**  
  2 │   
  3 │ **import** characters**.**movement**.Direction;**  
  4 │   
  5 │ **public** **class** **KnaveOfHearts** **extends** **Character** **{**  
  6 │  **private** **Soldier** guardLeft**;**  
  7 │  **private** **Soldier** guardRight**;**  
  8 │   
  9 │  **public** **KnaveOfHearts(String** name**)** **{**  
 10 │  **super(**name**);**  
 11 │  **}**  
 12 │   
 13 │  ***// Вложенный нестатический класс Солдата***  
 14 │  **public** **class** **Soldier** **extends** **Character** **{**  
 15 │  **public** **Soldier(String** name**)** **{**  
 16 │  **super(**name**);**  
 17 │  **}**  
 18 │   
 19 │  **public** **void** guard**(KnaveOfHearts** knave**)** **{**  
 20 │  **System.**out**.**println**(**getName**()** **+** " охраняет " **+** knave**.**getName**());**  
 21 │  **}**  
 22 │  **}**  
 23 │   
 24 │  ***// Методы для создания солдат-охранников***  
 25 │  **public** **void** assignGuardLeft**(Soldier** guard**)** **{**  
 26 │  **this.**guardLeft **=** guard**;**  
 27 │  guard**.**guard**(this);**  
 28 │  guard**.**standBy**(this,** **Direction.**BY\_LEFT\_HAND**);**  
 29 │  **}**  
 30 │   
 31 │  **public** **void** assignGuardRight**(Soldier** guard**)** **{**  
 32 │  **this.**guardRight **=** guard**;**  
 33 │  guard**.**guard**(this);**  
 34 │  guard**.**standBy**(this,** **Direction.**BY\_RIGHT\_HAND**);**  
 35 │  **}**  
 36 │ **}**  
 37 │

#### src/characters/Queen.java

 1 │ **package** characters**;**  
 2 │   
 3 │ **public** **class** **Queen** **extends** **Character** **{**  
 4 │  **public** **Queen(String** name**)** **{**  
 5 │  **super(**name**);**  
 6 │  **}**  
 7 │ **}**  
 8 │   
 9 │

#### src/characters/WhiteRabbit.java

  1 │ **package** characters**;**  
  2 │   
  3 │ **import** props**.Prop;**  
  4 │   
  5 │ **public** **class** **WhiteRabbit** **extends** **Character** **{**  
  6 │  **private** **Prop** objectInHand**;**  
  7 │  **public** **WhiteRabbit(String** name**){**  
  8 │  **super(**name**);**  
  9 │  **}**  
 10 │  @Override  
 11 │  **public** **void** hold**(Prop** object**){**  
 12 │  **this.**objectInHand **=** object**;**  
 13 │  **System.**out**.**println**(**getName**()** **+** " держит в лапке " **+** **this.**objectInHand**.**getName**());**  
 14 │  **}**  
 15 │ **}**

#### src/characters/movement/Direction.java

  1 │ **package** characters**.**movement**;**  
  2 │   
  3 │ **public** **enum** **Direction** **{**  
  4 │  CENTER**(**"в центр"**),**  
  5 │  LEFT**(**"налево"**),**  
  6 │  RIGHT**(**"направо"**),**  
  7 │  UP**(**"вверх"**),**  
  8 │  DOWN**(**"вниз"**),**  
  9 │  FRONT**(**"вперед"**),**  
 10 │  BACK**(**"назад"**),**  
 11 │  RIGHT\_SIDE**(**"с правой стороны от"**),**  
 12 │  LEFT\_SIDE**(**"с левой стороны от"**),**  
 13 │  BY\_RIGHT\_HAND**(**"по правую руку от"**),**  
 14 │  BY\_LEFT\_HAND**(**"по левую руку от"**),**  
 15 │  BEHIND**(**"сзади"**),**  
 16 │  AHEAD**(**"спереди"**);**  
 17 │   
 18 │  **private** **String** direction**;**  
 19 │   
 20 │  **Direction(String** direction**)** **{**  
 21 │  **this.**direction **=** direction**;**  
 22 │  **}**  
 23 │   
 24 │  **public** **String** getDirection**()** **{**  
 25 │  **return** direction**;**  
 26 │  **}**  
 27 │   
 28 │ **}**

#### src/characters/groups/Crowd.java

  1 │ **package** characters**.**groups**;**  
  2 │   
  3 │ **import** java**.**util**.ArrayList;**  
  4 │ **import** java**.**util**.List;**  
  5 │ **import** exceptions**.CantAttendException;**  
  6 │   
  7 │ **public** **class** **Crowd** **{**  
  8 │  **private** **List<Member>** members **=** **new** **ArrayList<>();**  
  9 │   
 10 │  ***// Статический вложенный класс Member***  
 11 │  **public** **static** **class** **Member** **{**  
 12 │  **private** **String** name**;**  
 13 │  **public** **Member(String** name**)** **{**  
 14 │  **this.**name **=** name**;**  
 15 │  **if** **(**name **==** "Карта Король"**)** **{**  
 16 │  **throw** **new** **CantAttendException(**"Короли не присоединяются к толпе"**);**  
 17 │  **}**  
 18 │  **}**  
 19 │   
 20 │  **public** **void** attend**()** **{**  
 21 │  **System.**out**.**println**(this.**name **+** " собирается в толпу"**);**  
 22 │  **}**  
 23 │  **}**  
 24 │   
 25 │  ***// Метод для добавления нового участника в толпу***  
 26 │  **public** **void** addMember**(Member** member**)** **{**  
 27 │  members**.**add**(**member**);**  
 28 │  **}**  
 29 │   
 30 │  ***// Метод для иллюстрации присутствия толпы***  
 31 │  **public** **void** gather**()** **{**  
 32 │  **for** **(Member** member **:** members**)** **{**  
 33 │  member**.**attend**();**  
 34 │  **}**  
 35 │  **}**  
 36 │ **}**  
 37 │

#### src/Main.java

  1 │ **import** characters**.\*;**  
  2 │ **import** characters**.Character;**  
  3 │ **import** characters**.**groups**.\*;**  
  4 │ **import** characters**.**movement**.\*;**  
  5 │ **import** exceptions**.\*;**  
  6 │ **import** food**.\*;**  
  7 │ **import** furniture**.\*;**  
  8 │ **import** props**.\*;**  
  9 │ **import** rooms**.\*;**  
  10 │   
  11 │   
  12 │ **public** **class** **Main** **{**  
  13 │  **public** **static** **void** main**(String[]** args**)** **throws** **Exception** **{**  
  14 │  **Queen** queen **=** **new** **Queen(**"Королева"**);**  
  15 │  **King** king **=** **new** **King(**"Король"**);**  
  16 │  **Alice** alice **=** **new** **Alice(**"Алиса"**);**  
  17 │  **Gryphon** gryphon **=** **new** **Gryphon(**"Грифон"**);**  
  18 │  **KnaveOfHearts** knaveOfHearts **=** **new** **KnaveOfHearts(**"Червонный Валет"**);**  
  19 │  **Furniture** kingsThrone **=** **new** **Throne(**"Трон"**);**  
  20 │  **Furniture** queensThrone **=** **new** **Throne(**"Трон"**);**  
  21 │  king**.**sitOn**(**kingsThrone**);**  
  22 │  queen**.**sitOn**(**queensThrone**);**  
  23 │   
  24 │  **int** retryCount **=** 3**;** ***// Количество попыток для прибытия***  
  25 │  ***// Checked exception***  
  26 │  **for** **(Character** character **:** **new** **Character[]** **{**alice**,** gryphon**})** **{**  
  27 │  **for** **(int** i **=** 0**;** i **<** retryCount**;** i**++)** **{**  
  28 │  **try** **{**  
  29 │  character**.**arrive**();**  
  30 │  **break;**  
  31 │  **}** **catch** **(ArrivalException** e**)** **{**  
  32 │  ***// System.err.println(e.getMessage());***  
  33 │  **if** **(**i **<** retryCount **-** 1**)** **{**  
  34 │  **System.**out**.**println**(**character**.**getName**()** **+** " ещё раз пытается найти путь"**);**  
  35 │  **}** **else** **{**  
  36 │  **System.**out**.**println**(**character**.**getName**()** **+** " так и не смог прибежать"**);**  
  37 │  **System.**exit**(**1**);**  
  38 │   
  39 │  **}**  
  40 │  **}**  
  41 │  **}**  
  42 │  **}**  
  43 │   
  44 │   
  45 │   
  46 │  ***// Толпа***  
  47 │  **Crowd** crowd **=** **new** **Crowd();**  
  48 │  ***// Unchecked exception***  
  49 │  **try** **{**  
  50 │  ***// Добавление участников толпы***  
  51 │  crowd**.**addMember**(new** **Crowd.Member(**"Пичужка 1"**));**  
  52 │  crowd**.**addMember**(new** **Crowd.Member(**"Пичужка 2"**));**  
  53 │  crowd**.**addMember**(new** **Crowd.Member(**"Зверушка 1"**));**  
  54 │  crowd**.**addMember**(new** **Crowd.Member(**"Зверушка 2"**));**  
  55 │  crowd**.**addMember**(new** **Crowd.Member(**"Карта Валет"**));**  
  56 │  crowd**.**addMember**(new** **Crowd.Member(**"Карта Туз"**));**  
  57 │  ***// Сбор толпы***  
  58 │  crowd**.**gather**();**  
  59 │  **}**  
  60 │  **catch** **(CantAttendException** e**)** **{**  
  61 │  **System.**err**.**println**(**e**.**getMessage**());**  
  62 │  **System.**err**.**println**(**"В толпу был добавлен запрещённый участник. Выполнение программы завершено."**);**  
  63 │  **System.**exit**(**1**);**  
  64 │  **}**  
  65 │  ***// Охрана***  
  66 │  **KnaveOfHearts.Soldier** guardLeft **=** knaveOfHearts**.new** **Soldier(**"Левый солдат"**);**  
  67 │  **KnaveOfHearts.Soldier** guardRight **=** knaveOfHearts**.new** **Soldier(**"Правый солдат"**);**  
  68 │   
  69 │  knaveOfHearts**.**assignGuardLeft**(**guardLeft**);**  
  70 │  knaveOfHearts**.**assignGuardRight**(**guardRight**);**  
  71 │   
  72 │  ***//=========================================================================***  
  73 │  **WhiteRabbit** whiteRabbit **=** **new** **WhiteRabbit(**"Белый Кролик"**);**  
  74 │  **Courtroom** courtroom **=** **new** **Courtroom(**"Судебный зал"**);**  
  75 │  **Table** table **=** **new** **Table(**"Стол"**);**  
  76 │  **Dish** dish **=** **new** **Dish(**"Большое блюдо с пирожками"**,** 10**);**  
  77 │  dish**.**addFood**(new** **Pie(**"Аппетитный пирожок"**,** "Курица"**));**  
  78 │  dish**.**addFood**(new** **Pie(**"Аппетитный пирожок"**,** "Рыба"**));**  
  79 │  dish**.**addFood**(new** **Pie(**"Аппетитный пирожок"**,** "Картошка"**));**  
  80 │   
  81 │  whiteRabbit**.**standBy**(**king**,** **Direction.**BY\_RIGHT\_HAND**);**  
  82 │  whiteRabbit**.**hold**(new** **Trumpet(**"Труба"**));**  
  83 │  whiteRabbit**.**hold**(new** **Scroll(**"Пергаментный свиток"**));**  
  84 │  courtroom**.**setCenterObject**(**table**);**  
  85 │  table**.**setTopObject**(**dish**);**  
  86 │  dish**.**lookLike**(**"аппетитно"**);**  
  87 │  alice**.**salivateAt**(**dish**);**  
  88 │  alice**.**think**(**"Хорошо бы, суд уже кончился и позвали к столу!"**);**  
  89 │  ***//=========================================================================***  
  90 │   
  91 │  ***// Алиса***  
  92 │  alice**.**observeScene**(new** **String[]{**"судья"**,** "трон"**,** "зверюшки"**,** "пичужки"**,** "карты"**});**  
  93 │  alice**.**think**(**"Вот это судья. Кто в большом парике, тот и судья"**);**  
  94 │  ***// Анонимный класс рассказчика***  
  95 │  **Character** narrator **=** **new** **Character(**"Рассказчик"**)** **{**  
  96 │  @Override  
  97 │  **public** **void** say**(String** speach**)** **{**  
  98 │  **System.**out**.**println**(**speach**);**  
  99 │  **}**  
 100 │  **};**  
 101 │  narrator**.**say**(**"Хотя Алиса раньше никогда не бывала в суде, она устала про суд в книжках, " **+**  
 102 │  "и ей было очень приятно отметить, что она знает, как тут все - или почти все - называется."**);**  
 103 │   
 104 │  **}**  
 105 │   
 106 │ **}**

#### src/rooms/Courtroom.java

  1 │ **package** rooms**;**  
  2 │   
  3 │ **import** furniture**.\*;**  
  4 │   
  5 │ **public** **class** **Courtroom** **extends** **Room** **{**  
  6 │  **public** **Courtroom(String** name**)** **{**  
  7 │  **super(**name**);**  
  8 │  **}**  
  9 │   
 10 │  **public** **void** setCenterObject**(Table** object**){**  
 11 │  **System.**out**.**println**(**"В центре " **+** getName**()** **+** " стоит " **+** object**.**getName**());**  
 12 │  **}**  
 13 │   
 14 │   
 15 │ **}**

#### src/rooms/Room.java

  1 │ **package** rooms**;**  
  2 │   
  3 │ **import** interfaces**.Nameable;**  
  4 │   
  5 │ **public** **abstract** **class** **Room** **implements** **Nameable** **{**  
  6 │  **private** **String** name**;**  
  7 │  **public** **Room(String** name**){**  
  8 │  **this.**name **=** name**;**  
  9 │  **}**  
 10 │  **public** **String** getName**(){**  
 11 │  **return** **this.**name**;**  
 12 │  **}**  
 13 │   
 14 │  @Override  
 15 │  **public** **boolean** equals**(Object** obj**)** **{**  
 16 │  **if** **(this** **==** obj**)** **return** true**;**  
 17 │  **if** **(**obj **==** null **||** getClass**()** **!=** obj**.**getClass**())** **return** false**;**  
 18 │  **Room** room **=** **(Room)** obj**;**  
 19 │  **return** name**.**equals**(**room**.**name**);**  
 20 │  **}**  
 21 │   
 22 │  @Override  
 23 │  **public** **int** hashCode**()** **{**  
 24 │  **return** name**.**hashCode**();**  
 25 │  **}**  
 26 │  @Override  
 27 │  **public** **String** toString**()** **{**  
 28 │  **return** "Room{" **+**  
 29 │  "name='" **+** name **+** '\'' **+**  
 30 │  '}'**;**  
 31 │  **}**  
 32 │ **}**  
 33 │

#### src/props/Dish.java

  1 │ **package** props**;**  
  2 │   
  3 │   
  4 │ **import** food**.Food;**  
  5 │   
  6 │ **import** java**.**util**.Arrays;**  
  7 │   
  8 │ **public** **class** **Dish** **extends** **Prop** **{**  
  9 │  **private** **Food[]** foodItems**;**  
 10 │  **private** **int** itemCount**;**  
 11 │   
 12 │  **public** **Dish(String** name**,** **int** maxItemCount**)** **{**  
 13 │  **super(**name**);**  
 14 │  **this.**foodItems **=** **new** **Food[**maxItemCount**];**  
 15 │  **this.**itemCount **=** 0**;**  
 16 │  **}**  
 17 │   
 18 │  **public** **void** addFood**(Food** foodItem**)** **{**  
 19 │  **if** **(**itemCount **<** foodItems**.**length**)** **{**  
 20 │  foodItems**[**itemCount**]** **=** foodItem**;**  
 21 │  itemCount**++;**  
 22 │  **}** **else** **{**  
 23 │  **System.**out**.**println**(**"На блюде больше нет места."**);**  
 24 │  **}**  
 25 │  **}**  
 26 │   
 27 │  **public** **Food[]** getFoodItems**()** **{**  
 28 │  **return** foodItems**;**  
 29 │  **}**  
 30 │ **}**

#### src/props/Prop.java

  1 │ **package** props**;**  
  2 │   
  3 │ **import** food**.\*;**  
  4 │ **import** interfaces**.Nameable;**  
  5 │   
  6 │ **public** **abstract** **class** **Prop** **implements** **Nameable** **{**  
  7 │  **protected** **String** name**;**  
  8 │   
  9 │  **public** **Prop(String** name**)** **{**  
 10 │  **this.**name **=** name**;**  
 11 │  **}**  
 12 │   
 13 │  **public** **String** getName**()** **{**  
 14 │  **return** **this.**name**;**  
 15 │  **}**  
 16 │  **public** **void** lookLike**(String** description**)** **{**  
 17 │  **System.**out**.**println**(**getName**()** **+** " выглядит " **+** description**);**  
 18 │  **}**  
 19 │  **public** **boolean** equals**(Object** obj**)** **{**  
 20 │  **if** **(this** **==** obj**)** **return** true**;**  
 21 │  **if** **(**obj **==** null **||** getClass**()** **!=** obj**.**getClass**())** **return** false**;**  
 22 │  **Prop** prop **=** **(Prop)** obj**;**  
 23 │  **return** name**.**equals**(**prop**.**name**);**  
 24 │  **}**  
 25 │   
 26 │  **public** **int** hashCode**()** **{**  
 27 │  **return** name**.**hashCode**();**  
 28 │  **}**  
 29 │   
 30 │  **public** **String** toString**()** **{**  
 31 │  **return** "Prop{" **+**  
 32 │  "name='" **+** name **+** '\'' **+**  
 33 │  '}'**;**  
 34 │  **}**  
 35 │ **}**  
 36 │

#### src/props/Scroll.java

  1 │ **package** props**;**  
  2 │   
  3 │ **public** **class** **Scroll** **extends** **Prop** **{**  
  4 │  **private** **String** text **=** "..."**;**  
  5 │  **public** **Scroll(String** name**){**  
  6 │  **super(**name**);**  
  7 │  **}**  
  8 │   
  9 │  **public** **String** getText**()** **{**  
 10 │  **return** text**;**  
 11 │  **}**  
 12 │   
 13 │  **public** **void** setText**(String** text**)** **{**  
 14 │  **this.**text **=** text**;**  
 15 │  **}**  
 16 │ **}**  
 17 │

#### src/props/Trumpet.java

  1 │ **package** props**;**  
  2 │   
  3 │ **public** **class** **Trumpet** **extends** **Prop** **{**  
  4 │  **public** **Trumpet(String** name**){**  
  5 │  **super(**name**);**  
  6 │  **}**  
  7 │   
  8 │  **public** **void** makeSound**(){**  
  9 │  **System.**out**.**println**(**getName**()** **+** " издаёт звук"**);**  
 10 │  **}**  
 11 │ **}**  
 12 │

#### src/interfaces/Nameable.java

 1 │ **package** interfaces**;**  
 2 │   
 3 │ **public** **interface** **Nameable** **{**  
 4 │  **String** getName**();**  
 5 │ **}**

#### src/interfaces/Thinkable.java

 1 │ **package** interfaces**;**  
 2 │   
 3 │ **public** **interface** **Thinkable** **{**  
 4 │  **void** think**(String** thought**);**  
 5 │ **}**

#### src/furniture/Furniture.java

  1 │ **package** furniture**;**  
  2 │   
  3 │ **import** interfaces**.Nameable;**  
  4 │   
  5 │ **public** **abstract** **class** **Furniture** **implements** **Nameable** **{**  
  6 │  **protected** **String** name**;**  
  7 │   
  8 │  **public** **Furniture(String** name**)** **{**  
  9 │  **this.**name **=** name**;**  
 10 │  **}**  
 11 │   
 12 │  **public** **String** getName**()** **{**  
 13 │  **return** **this.**name**;**  
 14 │  **}**  
 15 │   
 16 │  **public** **boolean** equals**(Object** obj**)** **{**  
 17 │  **if** **(this** **==** obj**)** **return** true**;**  
 18 │  **if** **(**obj **==** null **||** getClass**()** **!=** obj**.**getClass**())** **return** false**;**  
 19 │  **Furniture** furniture **=** **(Furniture)** obj**;**  
 20 │  **return** name**.**equals**(**furniture**.**name**);**  
 21 │  **}**  
 22 │   
 23 │  **public** **int** hashCode**()** **{**  
 24 │   
 25 │  **return** name**.**hashCode**();**  
 26 │  **}**  
 27 │   
 28 │  **public** **String** toString**()** **{**  
 29 │  **return** "Furniture{" **+**  
 30 │  "name='" **+** name **+** '\'' **+**  
 31 │  '}'**;**  
 32 │  **}**  
 33 │ **}**

#### src/furniture/Table.java

  1 │ **package** furniture**;**  
  2 │   
  3 │ **import** props**.\*;**  
  4 │ **public** **class** **Table** **extends** **Furniture** **{**  
  5 │  **public** **Table(String** name**){**  
  6 │  **super(**name**);**  
  7 │  **}**  
  8 │  **private** **Prop** topObject**;**  
  9 │  **private** **Prop** bottomObject**;**  
 10 │   
 11 │  **public** **void** setTopObject**(Prop** object**){**  
 12 │  **this.**topObject **=** object**;**  
 13 │  **System.**out**.**println**(**"На " **+** getName**()** **+** " стоит " **+** object**.**getName**());**  
 14 │  **}**  
 15 │  **public** **void** setBottomObject**(Prop** object**){**  
 16 │  **this.**bottomObject **=** object**;**  
 17 │  **System.**out**.**println**(**"Под " **+** getName**()** **+** " стоит " **+** object**.**getName**());**  
 18 │  **}**  
 19 │ **}**

#### src/furniture/Throne.java

 1 │ **package** furniture**;**  
 2 │   
 3 │ **public** **class** **Throne** **extends** **Furniture** **{**  
 4 │  **public** **Throne(String** name**)** **{**  
 5 │  **super(**name**);**  
 6 │  **}**  
 7 │ **}**

#### src/food/Food.java

  1 │ **package** food**;**  
  2 │   
  3 │ **import** interfaces**.Nameable;**  
  4 │ **public** **abstract** **class** **Food** **implements** **Nameable** **{**  
  5 │  **protected** **String** name**;**  
  6 │  **public** **Food(String** name**)** **{**  
  7 │  **this.**name **=** name**;**  
  8 │  **}**  
  9 │  **public** **String** getName**()** **{**  
 10 │  **return** **this.**name**;**  
 11 │  **}**  
 12 │  **public** **int** hashCode**()** **{**  
 13 │  **return** name**.**hashCode**();**  
 14 │  **}**  
 15 │  **public** **boolean** equals**(Object** obj**)** **{**  
 16 │  **if** **(this** **==** obj**)** **return** true**;**  
 17 │  **if** **(**obj **==** null **||** getClass**()** **!=** obj**.**getClass**())** **return** false**;**  
 18 │  **Food** food **=** **(Food)** obj**;**  
 19 │  **return** name**.**equals**(**food**.**name**);**  
 20 │  **}**  
 21 │  **public** **String** toString**()** **{**  
 22 │  **return** "Food{" **+**  
 23 │  "name='" **+** name **+** '\'' **+**  
 24 │  '}'**;**  
 25 │  **}**  
 26 │ **}**  
 27 │

#### src/food/Pie.java

  1 │ **package** food**;**  
  2 │   
  3 │ **import** interfaces**.Nameable;**  
  4 │   
  5 │ **import** javax**.**swing**.\*;**  
  6 │   
  7 │ **public** **class** **Pie** **extends** **Food** **{**  
  8 │  **private** **String** filling**;**  
  9 │  **public** **Pie(String** name**,** **String** filling**){**  
 10 │  **super(**name**);**  
 11 │  **this.**filling **=** filling**;**  
 12 │  **}**  
 13 │   
 14 │  **public** **String** getFilling**()** **{**  
 15 │  **return** **this.**filling**;**  
 16 │  **}**  
 17 │ **}**

#### src/exceptions/ArrivalException.java

 1 │ **package** exceptions**;**  
 2 │   
 3 │ **public** **class** **ArrivalException** **extends** **Exception** **{**  
 4 │  **public** **ArrivalException(String** message**)** **{**  
 5 │  **super(**message**);**  
 6 │  **}**  
 7 │ **}**  
 8 │

#### src/exceptions/CantAttendException.java

 1 │ **package** exceptions**;**  
 2 │   
 3 │ **public** **class** **CantAttendException** **extends** **RuntimeException{**  
 4 │  **public** **CantAttendException(String** message**)** **{**  
 5 │  **super(**message**);**  
 6 │  **}**  
 7 │ **}**  
 8 │

#### src/characters/Alice.java

 1 │ **package** characters**;**  
 2 │   
 3 │ **public** **class** **Alice** **extends** **Character** **{**  
 4 │  **public** **Alice(String** name**){**  
 5 │  **super(**name**);**  
 6 │  **}**  
 7 │ **}**

#### src/characters/Character.java

  1 │ **package** characters**;**  
  2 │   
  3 │ **import** characters**.**movement**.Direction;**  
  4 │ **import** furniture**.Furniture;**  
  5 │ **import** interfaces**.Nameable;**  
  6 │ **import** interfaces**.Thinkable;**  
  7 │ **import** props**.Prop;**  
  8 │ **import** exceptions**.ArrivalException;**  
  9 │   
  10 │ **public** **abstract** **class** **Character** **implements** **Nameable,** **Thinkable** **{**  
  11 │  **protected** **String** name**;**  
  12 │  **private** **Prop** objectInHand**;**  
  13 │   
  14 │  **public** **Character(String** name**)** **{**  
  15 │  **this.**name **=** name**;**  
  16 │  **}**  
  17 │   
  18 │  **public** **String** getName**()** **{**  
  19 │  **return** **this.**name**;**  
  20 │  **}**  
  21 │   
  22 │  **public** **void** standBy**(Character** character**,** **Direction** direction**)** **{**  
  23 │  **System.**out**.**println**(**getName**()** **+** " находится " **+** direction**.**getDirection**()** **+** " " **+** character**.**getName**());**  
  24 │  **}**  
  25 │   
  26 │  ***// checked exception***  
  27 │  **public** **void** arrive**()** **throws** **ArrivalException** **{**  
  28 │  **if** **(Math.**random**()** **<** 0.3**)** **{**  
  29 │  **throw** **new** **ArrivalException(**getName**()** **+** " заблудился пока бежал"**);**  
  30 │  **}**  
  31 │  **System.**out**.**println**(**getName**()** **+** " прибежал"**);**  
  32 │  **}**  
  33 │   
  34 │  **public** **void** salivateAt**(Prop** prop**)** **{**  
  35 │  **System.**out**.**println**(**"У " **+** getName**()** **+** " текут слюнки от " **+** prop**.**getName**());**  
  36 │  **}**  
  37 │   
  38 │  **public** **void** think**(String** thought**)** **{**  
  39 │  **System.**out**.**println**(**"\"" **+** thought **+** "\" - думает " **+** getName**());**  
  40 │  **}**  
  41 │   
  42 │  **public** **void** say**(String** speach**)** **{**  
  43 │  **System.**out**.**println**(**getName**()** **+** "говорит: " **+** speach**);**  
  44 │  **}**  
  45 │   
  46 │  **public** **void** sitOn**(Furniture** furniture**)** **{**  
  47 │  **System.**out**.**println**(**getName**()** **+** " сидит на " **+** furniture**.**getName**());**  
  48 │  **}**  
  49 │   
  50 │  **public** **void** observeScene**(String[]** sceneElements**)** **{**  
  51 │  ***// Локальный класс для элемента сцены***  
  52 │  **class** **SceneElement** **{**  
  53 │  **private** **String** name**;**  
  54 │   
  55 │  **SceneElement(String** name**)** **{**  
  56 │  **this.**name **=** name**;**  
  57 │  **}**  
  58 │   
  59 │  **public** **void** describe**()** **{**  
  60 │  **System.**out**.**println**(Character.this.**name **+** " рассматривает " **+** **this.**name**);**  
  61 │  **}**  
  62 │  **}**  
  63 │   
  64 │  **for** **(String** element **:** sceneElements**)** **{**  
  65 │  ***// Использование локального класса***  
  66 │  **new** **SceneElement(**element**).**describe**();**  
  67 │  **}**  
  68 │   
  69 │  **}**  
  70 │   
  71 │  **public** **void** walkTo**(Direction** direction**)** **{**  
  72 │  **System.**out**.**println**(**getName**()** **+** " идет " **+** direction**.**getDirection**());**  
  73 │  **}**  
  74 │   
  75 │  @Override  
  76 │  **public** **boolean** equals**(Object** obj**)** **{**  
  77 │  **if** **(this** **==** obj**)** **return** true**;**  
  78 │  **if** **(**obj **==** null **||** getClass**()** **!=** obj**.**getClass**())** **return** false**;**  
  79 │  **Character** character **=** **(Character)** obj**;**  
  80 │  **return** name**.**equals**(**character**.**name**);**  
  81 │  **}**  
  82 │   
  83 │  @Override  
  84 │  **public** **int** hashCode**()** **{**  
  85 │  **return** name**.**hashCode**();**  
  86 │  **}**  
  87 │   
  88 │  @Override  
  89 │  **public** **String** toString**()** **{**  
  90 │  **return** "Character{" **+**  
  91 │  "name='" **+** name **+** '\'' **+**  
  92 │  '}'**;**  
  93 │  **}**  
  94 │   
  95 │  **public** **void** hold**(Prop** object**)** **{**  
  96 │  **this.**objectInHand **=** object**;**  
  97 │  **System.**out**.**println**(**getName**()** **+** " держит " **+** **this.**objectInHand**.**getName**());**  
  98 │  **}**  
  99 │ **}**  
 100 │   
 101 │

#### src/characters/Gryphon.java

 1 │ **package** characters**;**  
 2 │   
 3 │ **public** **class** **Gryphon** **extends** **Character** **{**  
 4 │  **public** **Gryphon(String** name**){**  
 5 │  **super(**name**);**  
 6 │  **}**  
 7 │ **}**  
 8 │   
 9 │

#### src/characters/King.java

  1 │ **package** characters**;**  
  2 │   
  3 │ **public** **class** **King** **extends** **Character** **{**  
  4 │  **public** **King(String** name**){**  
  5 │  **super(**name**);**  
  6 │  **}**  
  7 │  **public** **void** command**(String** command**){**  
  8 │  **System.**out**.**println**(**getName**()** **+** " приказывает: " **+** command**);**  
  9 │  **}**  
 10 │ **}**

#### src/characters/KnaveOfHearts.java

  1 │ **package** characters**;**  
  2 │   
  3 │ **import** characters**.**movement**.Direction;**  
  4 │   
  5 │ **public** **class** **KnaveOfHearts** **extends** **Character** **{**  
  6 │  **private** **Soldier** guardLeft**;**  
  7 │  **private** **Soldier** guardRight**;**  
  8 │   
  9 │  **public** **KnaveOfHearts(String** name**)** **{**  
 10 │  **super(**name**);**  
 11 │  **}**  
 12 │   
 13 │  ***// Вложенный нестатический класс Солдата***  
 14 │  **public** **class** **Soldier** **extends** **Character** **{**  
 15 │  **public** **Soldier(String** name**)** **{**  
 16 │  **super(**name**);**  
 17 │  **}**  
 18 │   
 19 │  **public** **void** guard**(KnaveOfHearts** knave**)** **{**  
 20 │  **System.**out**.**println**(**getName**()** **+** " охраняет " **+** knave**.**getName**());**  
 21 │  **}**  
 22 │  **}**  
 23 │   
 24 │  ***// Методы для создания солдат-охранников***  
 25 │  **public** **void** assignGuardLeft**(Soldier** guard**)** **{**  
 26 │  **this.**guardLeft **=** guard**;**  
 27 │  guard**.**guard**(this);**  
 28 │  guard**.**standBy**(this,** **Direction.**BY\_LEFT\_HAND**);**  
 29 │  **}**  
 30 │   
 31 │  **public** **void** assignGuardRight**(Soldier** guard**)** **{**  
 32 │  **this.**guardRight **=** guard**;**  
 33 │  guard**.**guard**(this);**  
 34 │  guard**.**standBy**(this,** **Direction.**BY\_RIGHT\_HAND**);**  
 35 │  **}**  
 36 │ **}**  
 37 │

#### src/characters/Queen.java

 1 │ **package** characters**;**  
 2 │   
 3 │ **public** **class** **Queen** **extends** **Character** **{**  
 4 │  **public** **Queen(String** name**)** **{**  
 5 │  **super(**name**);**  
 6 │  **}**  
 7 │ **}**  
 8 │   
 9 │

#### src/characters/WhiteRabbit.java

  1 │ **package** characters**;**  
  2 │   
  3 │ **import** props**.Prop;**  
  4 │   
  5 │ **public** **class** **WhiteRabbit** **extends** **Character** **{**  
  6 │  **private** **Prop** objectInHand**;**  
  7 │  **public** **WhiteRabbit(String** name**){**  
  8 │  **super(**name**);**  
  9 │  **}**  
 10 │  @Override  
 11 │  **public** **void** hold**(Prop** object**){**  
 12 │  **this.**objectInHand **=** object**;**  
 13 │  **System.**out**.**println**(**getName**()** **+** " держит в лапке " **+** **this.**objectInHand**.**getName**());**  
 14 │  **}**  
 15 │ **}**

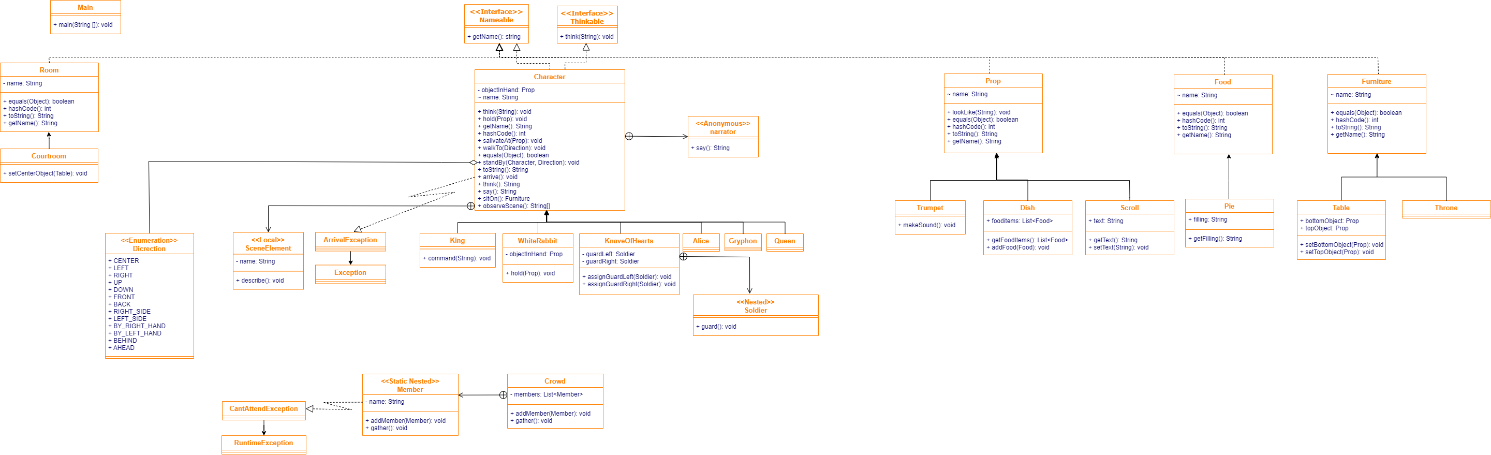
#### src/characters/movement/Direction.java

  1 │ **package** characters**.**movement**;**  
  2 │   
  3 │ **public** **enum** **Direction** **{**  
  4 │  CENTER**(**"в центр"**),**  
  5 │  LEFT**(**"налево"**),**  
  6 │  RIGHT**(**"направо"**),**  
  7 │  UP**(**"вверх"**),**  
  8 │  DOWN**(**"вниз"**),**  
  9 │  FRONT**(**"вперед"**),**  
 10 │  BACK**(**"назад"**),**  
 11 │  RIGHT\_SIDE**(**"с правой стороны от"**),**  
 12 │  LEFT\_SIDE**(**"с левой стороны от"**),**  
 13 │  BY\_RIGHT\_HAND**(**"по правую руку от"**),**  
 14 │  BY\_LEFT\_HAND**(**"по левую руку от"**),**  
 15 │  BEHIND**(**"сзади"**),**  
 16 │  AHEAD**(**"спереди"**);**  
 17 │   
 18 │  **private** **String** direction**;**  
 19 │   
 20 │  **Direction(String** direction**)** **{**  
 21 │  **this.**direction **=** direction**;**  
 22 │  **}**  
 23 │   
 24 │  **public** **String** getDirection**()** **{**  
 25 │  **return** direction**;**  
 26 │  **}**  
 27 │   
 28 │ **}**

#### src/characters/groups/Crowd.java

  1 │ **package** characters**.**groups**;**  
  2 │   
  3 │ **import** java**.**util**.ArrayList;**  
  4 │ **import** java**.**util**.List;**  
  5 │ **import** exceptions**.CantAttendException;**  
  6 │   
  7 │ **public** **class** **Crowd** **{**  
  8 │  **private** **List<Member>** members **=** **new** **ArrayList<>();**  
  9 │   
 10 │  ***// Статический вложенный класс Member***  
 11 │  **public** **static** **class** **Member** **{**  
 12 │  **private** **String** name**;**  
 13 │  **public** **Member(String** name**)** **{**  
 14 │  **this.**name **=** name**;**  
 15 │  **if** **(**name **==** "Карта Король"**)** **{**  
 16 │  **throw** **new** **CantAttendException(**"Короли не присоединяются к толпе"**);**  
 17 │  **}**  
 18 │  **}**  
 19 │   
 20 │  **public** **void** attend**()** **{**  
 21 │  **System.**out**.**println**(this.**name **+** " собирается в толпу"**);**  
 22 │  **}**  
 23 │  **}**  
 24 │   
 25 │  ***// Метод для добавления нового участника в толпу***  
 26 │  **public** **void** addMember**(Member** member**)** **{**  
 27 │  members**.**add**(**member**);**  
 28 │  **}**  
 29 │   
 30 │  ***// Метод для иллюстрации присутствия толпы***  
 31 │  **public** **void** gather**()** **{**  
 32 │  **for** **(Member** member **:** members**)** **{**  
 33 │  member**.**attend**();**  
 34 │  **}**  
 35 │  **}**  
 36 │ **}**  
 37 │

Диаграмма классов:



Вывод программы:

Король сидит на Трон

Королева сидит на Трон

Алиса прибежал

Грифон ещё раз пытается найти путь

Грифон прибежал

Пичужка 1 собирается в толпу

Пичужка 2 собирается в толпу

Зверушка 1 собирается в толпу

Зверушка 2 собирается в толпу

Карта Валет собирается в толпу

Карта Туз собирается в толпу

Левый солдат охраняет Червонный Валет

Левый солдат находится по левую руку от Червонный Валет

Правый солдат охраняет Червонный Валет

Правый солдат находится по правую руку от Червонный Валет

Белый Кролик находится по правую руку от Король

Белый Кролик держит в лапке Труба

Белый Кролик держит в лапке Пергаментный свиток

В центре Судебный зал стоит Стол

На Стол стоит Большое блюдо с пирожками

Большое блюдо с пирожками выглядит аппетитно

У Алиса текут слюнки от Большое блюдо с пирожками

"Хорошо бы, суд уже кончился и позвали к столу!" - думает Алиса

Алиса рассматривает судья

Алиса рассматривает трон

Алиса рассматривает зверюшки

Алиса рассматривает пичужки

Алиса рассматривает карты

"Вот это судья. Кто в большом парике, тот и судья" - думает Алиса

Хотя Алиса раньше никогда не бывала в суде, она устала про суд в книжках, и ей было очень приятно отметить, что она знает, как тут все - или почти все - называется.

# **Выводы**:

В результате проделанной работы, я изучил виды исключений и научился их обрабатывать. Кроме того, я узнал о локальных, анонимных и вложенных классах и поработал с ними на практике.