

# Университет ИТМО

Программирование  
Отчет по лабораторной работе №4

ФИО студента: Готовко Алексей Владимирович

Номер варианта: 322040

Направление подготовки: 09.03.04 (СППО)

Учебная группа: Р3119

ФИО преподавателя: Письмак Алексей Евгеньевич

Санкт-Петербург  
2021 г.

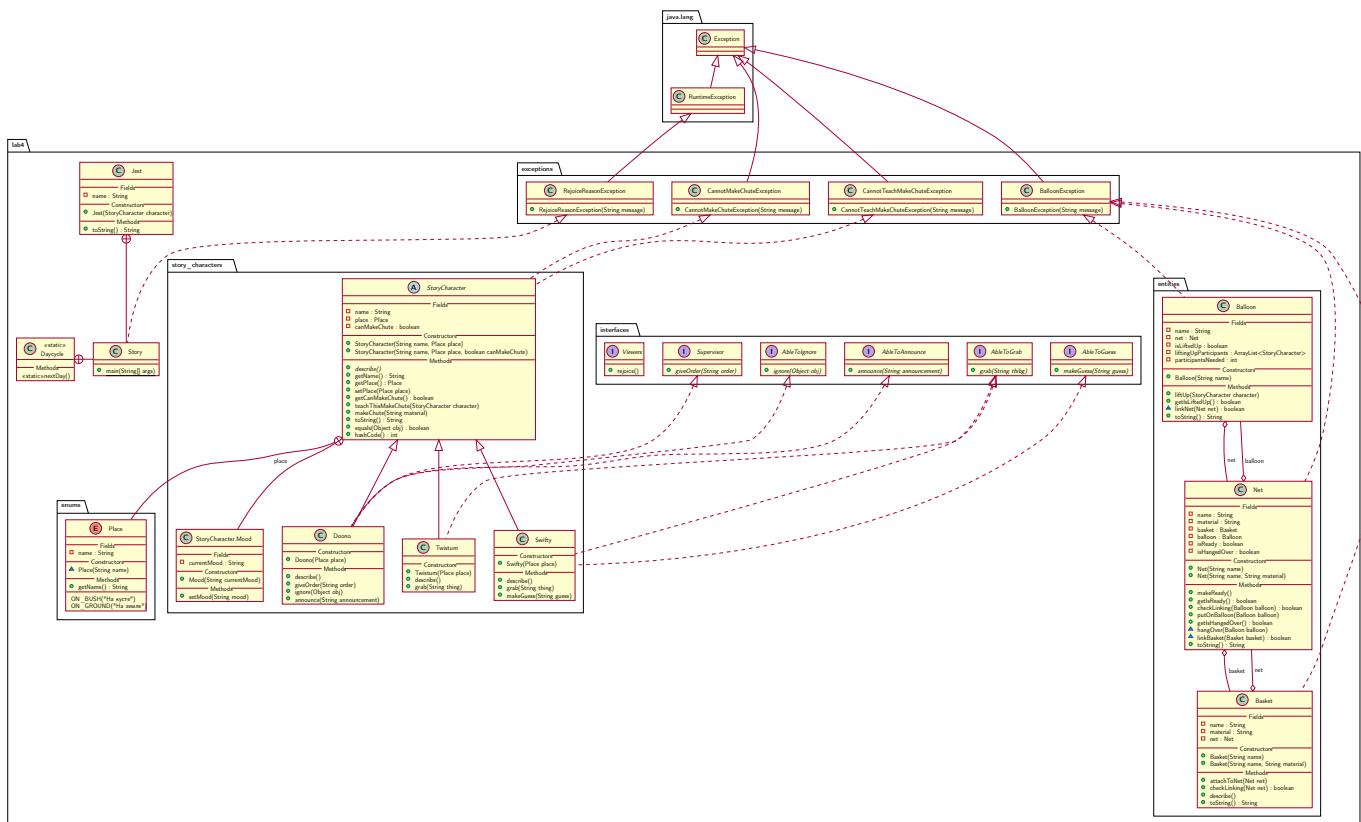
## **Содержание**

<b>1 Текст задания</b>	<b>2</b>
<b>2 Диаграмма классов</b>	<b>2</b>
<b>3 Результат работы программы</b>	<b>3</b>
<b>4 Исходный код</b>	<b>4</b>
<b>5 Вывод</b>	<b>10</b>

# 1 Текст задания

Но Знайка не обращал внимания на эти насмешки. Когда шелковая сеть была готова, он велел накинуть ее сверху на шар. Сеть растянули и накрыли ею шар сверху. Знайка велел подцепить шар веревкой снизу, привязать к ветке орехового куста и подтянуть кверху. Сейчас же Торопыжка и Шпунтик взобрались с веревкой на куст и стали подтягивать шар кверху. Это очень обрадовало зрителей. Когда шар приподняли над землей, сетка по краям его свесилась вниз, и Знайка велел привязать к углам сетки корзину из березовой коры. Корзина была четырехугольная. С каждой стороны в ней было сделано по лавочке, и на каждой лавочке могло поместиться по четыре малыша. Корзину привязали к сетке за четыре угла, и Знайка объявил, что работа по постройке шара закончена. Торопыжка вообразил, что уже можно лететь, но Знайка сказал, что еще надо подготовить для всех парашюты. На следующий день Знайка и его товарищи были заняты изготовлением парашютов. Каждый сам для себя мастерил парашют из пушинок одуванчика, а Знайка всем показывал, как надо делать.

# 2 Диаграмма классов



### 3 Результат работы программы

Персонажи:

Знайка – умный

Торопыжка – быстрый

Шпунтик – сообразительный

Над персонажем Знайка насмехаются

Персонаж Знайка игнорирует объект 'Насмешка над персонажем Знайка'

'Сетка из материала Шелк' готова

Персонаж Знайка велит накинуть объект 'Сетка из материала Шелк' на объект 'Воздушный шар'

Объект 'Сетка из материала Шелк' теперь находится на объекте 'Воздушный шар'

Персонаж Знайка велит подцепить объект 'Воздушный шар' веревкой снизу, привязать к ветке орехового куста и подтянуть кверху

Персонаж Торопыжка схватил предмет 'Веревка'

Персонаж Торопыжка перешел в локацию 'На кусте'

Персонаж Торопыжка начал приподнимать объект 'Воздушный шар'

Чтобы приподнять объект Воздушный шар необходимо 2 персонажа (-ей). Сейчас: 1

Персонаж Шпунтик схватил предмет 'Веревка'

Персонаж Шпунтик перешел в локацию 'На кусте'

Персонаж Шпунтик начал приподнимать объект 'Воздушный шар'

Чтобы приподнять объект Воздушный шар необходимо 2 персонажа (-ей). Сейчас: 2

Объект 'Воздушный шар' приподнялся

Объект 'Сетка из материала Шелк' свесился вниз

Зрители обрадовались

Персонаж Знайка велит привязать к объекту 'Сетка из материала Шелк' объект 'Корзина из материала Березовая кора'

Теперь к объекту 'Сетка из материала Шелк' прикреплен объект 'Корзина из материала Березовая кора'

'Корзина из материала Березовая кора': четырехугольная; с каждой ее стороны есть лавочка, и на каждой лавочке может поместиться по четыре малыша

Персонаж Знайка объявляет, что работа по постройке шара закончена

Персонаж Торопыжка предполагает, что уже можно лететь

Персонаж Знайка объявляет, что еще надо приготовить для всех парашюты

Наступает следующий день

Персонаж Знайка хочет научить персонажа Торопыжка делать парашют

Персонаж Знайка научил персонажа Торопыжка делать парашют

Персонаж Знайка хочет научить персонажа Шпунтик делать парашют

Персонаж Знайка научил персонажа Шпунтик делать парашют

Персонаж Знайка хочет сделать парашют

Персонаж Знайка сделал парашют из материала 'Пушинки одуванчика'

Персонаж Торопыжка хочет сделать парашют

Персонаж Торопыжка сделал парашют из материала 'Пушинки одуванчика'

Персонаж Шпунтик хочет сделать парашют

Персонаж Шпунтик сделал парашют из материала 'Пушинки одуванчика'

## 4 Исходный код

```
Story.java

1 package lab4;
2
3 import lab4.entities.Balloon;
4 import lab4.entities.Basket;
5 import lab4.entities.Net;
6 import lab4.exceptions.BalloonException;
7 import lab4.exceptions.CannotMakeChuteException;
8 import lab4.exceptions.CannotTeachMakeChuteException;
9 import lab4.exceptions.NoBalloonException;
10 import lab4.interfaces.Viewers;
11 import lab4.story_characters.StoryCharacter;
12 import lab4.story_characters.Doono;
13 import lab4.story_characters.Swifty;
14 import lab4.story_characters.Twistum;
15
16 import static lab4.enums.Place.*;
17
18 public class Story {
19
20     public static class Daycycle { // <-- Static nested class
21         static void nextDay() {
22             System.out.println("Наступает следующий день");
23         }
24     }
25
26     public static void main(String[] args) {
27
28         // ...
29
30         Doono doono = new Doono(ON_GROUND);
31         Swifty swifty = new Swifty(ON_GROUND);
32         Twistum twistum = new Twistum(ON_GROUND);
33
34         System.out.println("Персонажи:");
35         StoryCharacter[] characters = {doono, swifty, twistum};
36         for (StoryCharacter character : characters) {
37             character.describe();
38         }
39         System.out.println();
40
41     // ...
42
43     class Jest { // <-- Local class
44         private String name = "Насмешка над персонажем";
45
46         public Jest(StoryCharacter character) {
47             name += character.getName();
48         }
49         System.out.println("Над персонажем " + character.getName() + " насмехается");
50     }
51
52         @Override
53         public String toString() {
54             return name;
55         }
56     }
57
58     Jest jest = new Jest(doono);
59     doono.ignore(jest);
60     System.out.println();
61
62     // ...
63
64     Balloon balloon = new Balloon("Воздушный шар");
65     Net net = new Net("Сетка", "Мяч");
66     Basket basket = new Basket("Корзина", "Березовая кора");
67 }
```

```

Story.java

67     net.makeReady();
68
69     if (!net.getIsReady()) {
70         throw new RuntimeException("Невозможно продолжить историю. Сетка еще не готова");
71     }
72
73     doono.giveOrder("накинуть объект " + net + " на объект " + balloon);
74     try {
75         net.pushOnBalloon(balloon);
76     } catch (BalloonException e) {
77         System.out.println(e.getMessage());
78     }
79
80     if (!net.checkLinking(balloon)) {
81         throw new RuntimeException("Невозможно продолжить историю. Не удалось накинуть сетку на
82     }
83
84     doono.giveOrder("подцепить объект " + balloon + " веревкой снизу, привязать к ветке
85     деревьев и подтянуть крахм");
86
87     swifty.grab("Веревка");
88     swifty.setPlace(ON_BUSH);
89     try {
90         balloon.liftUp(swifty);
91     } catch (BalloonException e) {
92         System.out.println(e.getMessage());
93
94     twiustum.grab("Веревка");
95     twiustum.setPlace(ON_BUSH);
96     try {
97         balloon.liftUp(twiustum);
98     } catch (BalloonException e) {
99         System.out.println(e.getMessage());
100    }
101
102
103    Viewers viewers = new Viewers() { }; // <<< Anonymous class
104
105    if (balloon.getIsLiftedUp()) {
106        viewers.rejoice();
107    }
108    else {
109        throw new RejoiceReasonException("Cannot continue story. Viewers must rejoice but there
110    is nothing to rejoice at");
111    }
112
113    doono.giveOrder("привязать к объекту " + net + " объект " + basket);
114    try {
115        basket.attachToNet(net);
116    } catch (BalloonException e) {
117        System.out.println(e.getMessage());
118    }
119
120    if (!basket.checkLinking(net)) {
121        throw new RuntimeException("Невозможно продолжить историю. Не удалось прикрепить
122    корзину к сетке");
123
124    doono.announce("работа по постройке шара закончена");
125    swifty.makeGuess("уши можно легеть");
126    doono.announce("еще надо приготовить для всех парашюты");
127
128    System.out.println();

```

```
Story.java
129 // ...
130
131
132 Story.Daycycle.nextDay();
133
134 try {
135     swifty.teachThisMakeChute(doono);
136 } catch (CannotTeachMakeChuteException e) {
137     System.out.println(e.getMessage());
138 }
139
140 try {
141     twilstum.teachThisMakeChute(doono);
142 } catch (CannotTeachMakeChuteException e) {
143     System.out.println(e.getMessage());
144 }
145
146 for (StoryCharacter character : characters) {
147     System.out.println("Персонаж " + character.getName() + " хочет сделать парашют");
148     try {
149         character.makeChute("рушки одуванчика");
150     } catch (CannotMakeChuteException e) {
151         System.out.println(e.getMessage());
152     }
153 }
154
155 }
156 }
```

```
StoryCharacter.java

1 package lab4.story_characters;
2
3 import lab4.enums.Place;
4 import lab4.exceptions.CannotMakeChuteException;
5 import lab4.exceptions.CannotTeachMakeChuteException;
6
7 abstract public class StoryCharacter {
8     private final String name;
9     private Place place;
10    private boolean canMakeChute = false;
11
12    // _____ Constructors _____
13
14    public StoryCharacter(String name, Place place) {
15        this.name = name;
16        this.place = place;
17    }
18
19    public StoryCharacter(String name, Place place, boolean canMakeChute) {
20        this.name = name;
21        this.place = place;
22        this.canMakeChute = canMakeChute;
23    }
24
25
26    // _____ Fields access methods _____
27
28    public String getName() {
29        return name;
30    }
31
32    public Place getPlace(){
33        return place;
34    }
35
36    public void setPlace(Place place) {
37        this.place = place;
38        System.out.println("Персонаж " + name + " перешел в локацию " + place.getName() + "\n");
39    }
40
41    public boolean getCanMakeChute() {
42        return canMakeChute;
43    }
44
45
46    public void teachThisMakeChute(StoryCharacter character) throws CannotTeachMakeChuteException {
47        System.out.println("Персонаж " + character.getName() + " хочет научить персонажа " + name +
48            " делать парашюты");
49        if (character.getCanMakeChute()) {
50            throw new CannotTeachMakeChuteException("Персонаж " + character.getName() + " не умеет
51            делать парашют и не может научить персонажа " + name + " делать парашют");
52        }
53        if (canMakeChute) {
54            System.out.println("Персонаж " + this.name + " уже умеет делать парашют");
55        }
56        else {
57            canMakeChute = true;
58            System.out.println("Персонаж " + character.getName() + " научил персонажа " + name + "
59            делать парашют");
60        }
61    }
62    // _____ Misc methods _____
63    public void makeChute(String material) throws CannotMakeChuteException {
```

```

StoryCharacter.java
1 package lab4.story_characters;
2
3 import lab4.enums.Place;
4 import lab4.interfaces.AbleToGrab;
5 import lab4.interfaces.AbleToGuess;
6
7 public class StoryCharacter implements AbleToGrab, AbleToGuess {
8     public String name;
9     public Place place;
10
11     public StoryCharacter(String name, Place place) {
12         this.name = name;
13         this.place = place;
14     }
15
16     @Override
17     public void makeChute() {
18         System.out.println("Персонаж " + name + " не умеет делать парашют");
19     }
20
21     // _____ Abstract methods _____
22
23     abstract public void describe();
24
25     // _____ Overridden methods _____
26
27     @Override
28     public String toString() {
29         return String.format("lab4.story_characters.StoryCharacter = {name: %s, currentPlace: %s}"
30             , name, place);
31     }
32
33     @Override
34     public boolean equals(Object obj) {
35         if (obj == null) return false;
36         if (obj instanceof StoryCharacter) {
37             return (name.equals(((StoryCharacter) obj).getName()) &&
38                 place.equals(((StoryCharacter) obj).getPlace()));
39         }
40         return false;
41     }
42
43     @Override
44     public int hashCode() {
45         return (name.hashCode() + place.hashCode());
46     }
47
48     // _____ Nested non-static class _____
49     public class Mood {
50         private String currentMood;
51
52         public Mood(String currentMood) {
53             this.currentMood = currentMood;
54         }
55         public void setMood(String mood) {
56             this.currentMood = mood;
57         }
58     }
59
60 }

```

Page 2 of 2

```

Doonoo.java
1 package lab4.story_characters;
2
3 import lab4.enums.Place;
4 import lab4.interfaces.AbleToAnnounce;
5 import lab4.interfaces.AbleToIgnore;
6 import lab4.interfaces.Supervisor;
7
8 public class Doonoo extends StoryCharacter implements Supervisor, AbleToIgnore, AbleToAnnounce {
9     public Doonoo(Place place) {
10         super("Энайка", place, true);
11     }
12
13     @Override
14     public void describe() {
15         System.out.println("Энайка -- умный");
16     }
17
18     @Override
19     public void giveOrder(String order) {
20         System.out.println("Персонаж Энайка велит " + order);
21     }
22
23     @Override
24     public void ignore(Object obj) {
25         System.out.println("Персонаж Энайка игнорирует объект '" + obj + "'");
26     }
27
28     @Override
29     public void announce(String announcement) {
30         System.out.println("Персонаж Энайка объявляет, что " + announcement);
31     }
32
33 }

```

Page 1 of 1

```

Swiftly.java
1 package lab4.story_characters;
2
3 import lab4.enums.Place;
4 import lab4.interfaces.AbleToGrab;
5 import lab4.interfaces.AbleToGuess;
6
7 public class Swiftly extends StoryCharacter implements AbleToGrab, AbleToGuess {
8     public Swiftly(Place place) {
9         super("Торопычка", place);
10    }
11
12    @Override
13    public void describe() {
14        System.out.println("Торопычка -- быстрый");
15    }
16
17    @Override
18    public void grab(String thing) {
19        System.out.println("Персонаж Торопычка схватил предмет '" + thing + "'");
20    }
21
22    @Override
23    public void makeGuess(String guess) {
24        System.out.println("Персонаж Торопычка предполагает, что " + guess);
25    }
26
27 }
28

```

Page 1 of 1

```

Twistum.java
1 package lab4.story_characters;
2
3 import lab4.enums.Place;
4 import lab4.interfaces.AbleToGrab;
5
6 public class Twistum extends StoryCharacter implements AbleToGrab {
7     public Twistum(Place place) {
8         super("Шпунтик", place);
9     }
10
11    @Override
12    public void describe() {
13        System.out.println("Шпунтик -- сообразительный");
14    }
15
16    @Override
17    public void grab(String thing) {
18        System.out.println("Персонаж Шпунтик схватил предмет '" + thing + "'");
19    }
20
21 }
22

```

Page 1 of 1

AbleToAnnounce.java

```
1 package lab4.interfaces;
2
3 public interface AbleToAnnounce {
4     void announce(String announcement);
5 }
6
```

Page 1 of 1

AbleToGrab.java

```
1 package lab4.interfaces;
2
3 public interface AbleToGrab {
4     void grab(String thing);
5 }
6
```

Page 1 of 1

AbleToGuess.java

```
1 package lab4.interfaces;
2
3 public interface AbleToGuess {
4     void makeGuess(String guess);
5 }
6
```

Page 1 of 1

AbleToIgnore.java

```
1 package lab4.interfaces;
2
3 public interface AbleToIgnore {
4     void ignore(Object obj);
5 }
6
```

Page 1 of 1

```
Supervisor.java
1 package lab4.interfaces;
2
3 public interface Supervisor {
4     void giveOrder(String order);
5 }
6
```

Page 1 of 1

```
Viewers.java
1 package lab4.interfaces;
2
3 public interface Viewers {
4     default void rejoice() {
5         System.out.println("Зрители обрадовались");
6     }
7 }
```

Page 1 of 1

```
Place.java
1 package lab4.enums;
2
3
4 public enum Place {
5     ON_GROUND("На земле"),
6     ON_BUSH("На кусте");
7
8     private final String name;
9
10    Place(String name) {
11        this.name = name;
12    }
13
14    public String getName() {
15        return this.name;
16    }
17 }
18
```

Page 1 of 1

```
Balloon.java
1 package lab4.entities;
2
3 import lab4.exceptions.BalloonException;
4 import lab4.story_characters.storyCharacter;
5
6 import java.util.ArrayList;
7
8 public class Balloon {
9     private final String name;
10    private Net net;
11    private boolean isLiftedUp = false;
12
13    private ArrayList<StoryCharacter> liftingUpParticipants = new ArrayList<StoryCharacter>();
14    private final int participantsNeeded = 2;
15
16
17    public Balloon(String name) {
18        this.name = name;
19    }
20
21
22    public void liftUp(StoryCharacter character) throws BalloonException {
23        if (!liftingUpParticipants.contains(character)) {
24            throw new BalloonException("Не получилось. Этот персонаж уже начал приподнимать объект " +
25                + name);
26        }
27        if (isLiftedUp) {
28            throw new BalloonException("Не получилось. Объект " + name + " уже поднят");
29        }
30
31        liftingUpParticipants.add(character);
32        System.out.println("Персонаж " + character.getName() + " начал приподнимать объект " + this
33        );
34        System.out.printf("Чтобы приподнять объект %s необходимо %d персонажа (-ей). Сейчас: %d\n",
35            name, participantsNeeded, liftingUpParticipants.size());
36        if (liftingUpParticipants.size() > participantsNeeded) {
37            isLiftedUp = true;
38            System.out.println("Объект " + this + " приподнялся");
39            net.hangOver(this);
40        }
41
42    public boolean getIsLiftedUp() {
43        return isLiftedUp;
44    }
45
46
47    boolean linkNet(Net net) {
48        if (this.net == null) {
49            this.net = net;
50            return true;
51        }
52        return false;
53    }
54
55    @Override
56    public String toString() {
57        return "!" + name + "!";
58    }
59 }
```

Page 1 of 1

```

Basket.java
1 package lab4.entities;
2
3 import lab4.exceptions.BalloonException;
4
5 public class Basket {
6     private final String name;
7     private String material;
8     private Net net;
9
10    public Basket(String name) {
11        this.name = name;
12    }
13    public Basket(String name, String material) {
14        this.name = name;
15        this.material = material;
16    }
17
18    public void attachToNet(Net net) throws BalloonException {
19        if (this.net != null) {
20            throw new BalloonException("Невозможно прикрепить объект " + this + ", так как он уже
21            прикреплен");
22        }
23        if (net.linkBasket(this)) {
24            this.net = net;
25            System.out.println("Теперь к объекту " + net + " прикреплен объект " + this);
26            describe();
27        } else throw new BalloonException("Не получилось. К объекту " + net + " уже прикреплен
28        другой объект");
29    }
30
31    public boolean checkLinking(Net net) {
32        return this.net == net;
33    }
34    public void describe() {
35        System.out.println(this + ": четырехугольная; с каждой ее стороны есть лавочка, и на каждой
36        лавочке может поместиться по четыре мальши");
37    }
38    @Override
39    public String toString() {
40        return (material != null) ? ("'" + name + " из материала " + this.material + "'") : ("'" +
41        name + "'");
42    }

```

Page 1 of 1

```

Net.java
1 package lab4.entities;
2
3 import lab4.exceptions.BalloonException;
4
5 public class Net {
6     private final String name;
7     private String material;
8     private Basket basket;
9     private Balloon balloon;
10    private boolean isReady = false;
11    private boolean isHangedOver = false;
12
13
14    public Net(String name) {
15        this.name = name;
16    }
17    public Net(String name, String material) {
18        this.name = name;
19        this.material = material;
20    }
21
22    public void makeReady() {
23        isReady = true;
24        System.out.println(this + " готова");
25    }
26
27    public boolean getIsReady() {
28        return isReady;
29    }
30
31    public boolean checkLinking(Balloon balloon) {
32        return this.balloon == balloon;
33    }
34
35
36    public void putOnBalloon(Balloon balloon) throws BalloonException {
37        if (isReady && !balloon.getIsLiftedUp() && balloon.linkNet(this)) {
38            this.balloon = balloon;
39            System.out.println("Объект " + this + " теперь находится на объекте " + balloon);
40        } else {
41            throw new BalloonException("Не получилось. Чтобы положить сетку на шар, сетка должна
42            быть готова, шар не должен быть поднят и на шаре не должно быть другой сетки");
43        }
44    }
45
46    public boolean getIsHangedOver() {
47        return isHangedOver;
48    }
49
50
51    void hangOver(Balloon balloon) {
52        if (balloon.getIsLiftedUp() && balloon.equals(this.balloon)) {
53            isHangedOver = true;
54            System.out.println("Объект " + this + " весит вниз");
55        }
56    }
57
58    boolean linkBasket(Basket basket) {
59        if (this.basket == null && isHangedOver) {
60            this.basket = basket;
61            return true;
62        }
63        return false;
64    }
65

```

Page 1 of 2

```

Net.java
66
67    @Override
68    public String toString() {
69        return (material != null) ? ("'" + name + " из материала " + this.material + "'") : ("'" +
70        name + "'");
71    }
72

```

Page 2 of 2

```

BalloonException.java
1 package lab4.exceptions;
2
3 public class BalloonException extends Exception {
4     public BalloonException(String message) {
5         super(message);
6     }
7 }
8

```

Page 1 of 1

CannotMakeChuteException.java

```
1 package lab4.exceptions;
2
3 public class CannotMakeChuteException extends Exception {
4     public CannotMakeChuteException(String message) {
5         super(message);
6     }
7 }
```

Page 1 of 1

CannotTeachMakeChuteException.java

```
1 package lab4.exceptions;
2
3 public class CannotTeachMakeChuteException extends Exception{
4     public CannotTeachMakeChuteException(String message) {
5         super(message);
6     }
7 }
```

Page 1 of 1

RejoiceReasonException.java

```
1 package lab4.exceptions;
2
3 public class RejoiceReasonException extends RuntimeException {
4     public RejoiceReasonException(String message) {
5         super(message);
6     }
7 }
```

Page 1 of 1

## **5 Вывод**

В результате выполнения лабораторной работы были изучены классы исключений в Java (checked exceptions, unchecked exceptions, runtime exceptions) и их обработка, анонимные классы, вложенные классы (static nested classes и non-static nested classes) и локальные классы; были созданы собственные классы исключений.