Решения задач лекции 29

Задача 1.

```java
public class Triangle {

    private final int ab;
    private final int bc;
    private final int ac;

    public Triangle(int ab, int bc, int ac) {
        this.ab = ab;
        this.bc = bc;
        this.ac = ac;
    }

    public int getSide1() {
        return ab;
    }

    public int getSide2() {
        return bc;
    }

    public int getSide3() {
        return ac;
    }
}
```

```java
public interface Square {

    double calc(Triangle triangle);
}
```

```java
public abstract class TriangleChain implements Square {

    private TriangleChain next;

    public void setNext(TriangleChain next) {
        this.next = next;
    }

    public double getSquare(Triangle triangle) {
        if (satisfyConditions(triangle)) {
            System.out.println("определен тип треугольника: " +
getTriangleType());
            return calc(triangle);
        } else if (next != null) {
            return next.getSquare(triangle);
        } else
            throw new IllegalArgumentException("площадь не была
посчитана");
```

```java
    }

    abstract boolean satisfyConditions(Triangle triangle);
    abstract String getTriangleType();
}
```

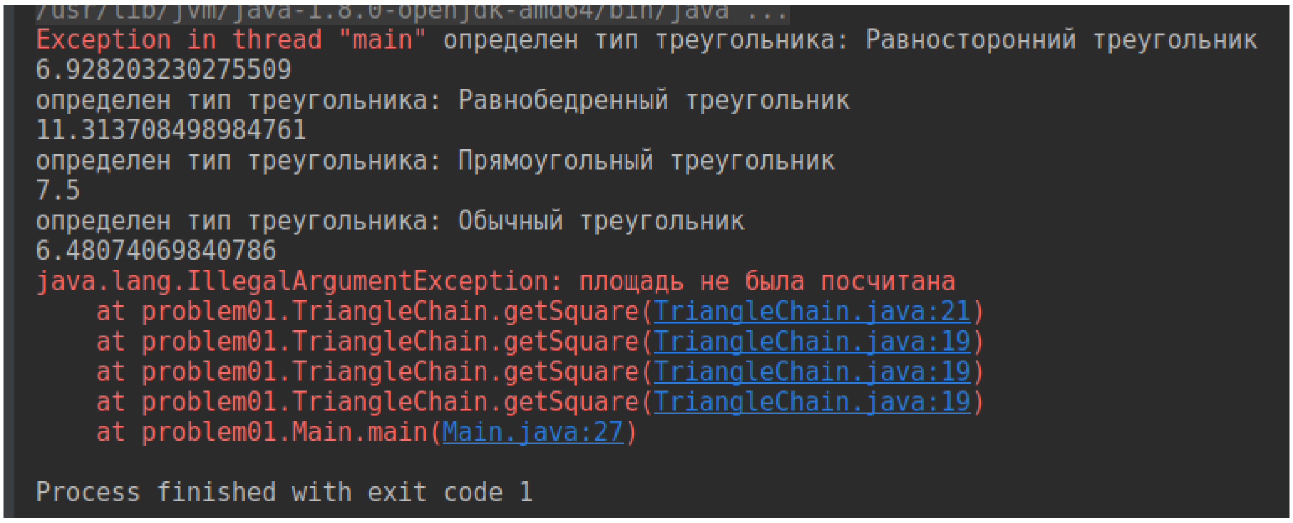
```java
public class EquiliteralTriangleChain extends TriangleChain {

    @Override
    boolean satisfyConditions(Triangle triangle) {
        return triangle.getSide1() == triangle.getSide2() &&
                triangle.getSide2() == triangle.getSide3() &&
                triangle.getSide1() == triangle.getSide3();
    }

    @Override
    String getTriangleType() {
        return "Равносторонний треугольник";
    }

    @Override
    public double calc(Triangle triangle) {
        return triangle.getSide1() * triangle.getSide2() *
Math.sqrt(3) * 0.25;
    }
}
```

```java
public class IsoScelesTriangleChain extends TriangleChain {

    @Override
    boolean satisfyConditions(Triangle triangle) {
        return triangle.getSide1() == triangle.getSide2() ||
                triangle.getSide1() == triangle.getSide3() ||
                triangle.getSide2() == triangle.getSide3();
    }

    @Override
    String getTriangleType() {
        return "Равнобедренный треугольник";
    }

    @Override
    public double calc(Triangle triangle) {
        int base = triangle.getSide1();
        int side = triangle.getSide2();
        if (triangle.getSide1() == triangle.getSide2()) {
            base = triangle.getSide3();
            side = triangle.getSide1();
        } else if (triangle.getSide1() == triangle.getSide3()) {
            base = triangle.getSide2();
            side = triangle.getSide1();
        }
        double half = 0.5 * base;
        double height = Math.sqrt(side * side - half * half);
        return half * height;
    }
}
```

```java
public class RightTriangleChain extends TriangleChain {

    @Override
    boolean satisfyConditions(Triangle tr) {
        return tr.getSide1() * tr.getSide1() + tr.getSide2() *
tr.getSide2() == tr.getSide3() * tr.getSide3() ||
                tr.getSide1() * tr.getSide1() + tr.getSide3() *
tr.getSide3() == tr.getSide2() * tr.getSide2() ||
                tr.getSide3() * tr.getSide3() + tr.getSide2() *
tr.getSide2() == tr.getSide1() * tr.getSide1();
    }

    @Override
    String getTriangleType() {
        return "Прямоугольный треугольник";
    }

    @Override
    public double calc(Triangle triangle) {
        int biggest = findMax(triangle.getSide1(),
triangle.getSide2(), triangle.getSide3());
        double result;
        if (biggest == triangle.getSide1()) {
            result = triangle.getSide2() * triangle.getSide3();
        } else if (biggest == triangle.getSide2()) {
            result = triangle.getSide2() * triangle.getSide1();
        } else {
            result = triangle.getSide1() * triangle.getSide3();
        }
        return 0.5 * result;
    }

    private int findMax(int a, int b, int c) {
        if (a > b && a > c) {
            return a;
        } else if (b > a && b > c) {
            return b;
        } else {
            return c;
        }
    }
}
```

```java
public class BaseTriangleChain extends TriangleChain {

    @Override
    boolean satisfyConditions(Triangle triangle) {
        return triangle.getSide1() > 0 && triangle.getSide2() > 0 &&
triangle.getSide3() > 0 &&
                triangle.getSide1() + triangle.getSide2() >
triangle.getSide3() &&
                triangle.getSide1() + triangle.getSide3() >
triangle.getSide2() &&
                triangle.getSide3() + triangle.getSide2() >
triangle.getSide1();
    }

    @Override
    String getTriangleType() {
        return "Обычный треугольник";
    }

    @Override
    public double calc(Triangle triangle) {
        int half = (triangle.getSide1() + triangle.getSide2() +
triangle.getSide3()) / 2;
        return Math.sqrt(half * (half - triangle.getSide1()) * (half
- triangle.getSide2()) * (half - triangle.getSide3()));
    }
}
```

```
public class Main {
    public static void main(String[] args) {
        List<Triangle> triangleList = new ArrayList<>();
        triangleList.add(new Triangle(4, 4, 4));
        triangleList.add(new Triangle(4, 6, 6));
        triangleList.add(new Triangle(3, 4, 5));
        triangleList.add(new Triangle(4, 5, 6));
        triangleList.add(new Triangle(4, 1, 6));
        TriangleChain chain0 = new EquiliteralTriangleChain();
        TriangleChain chain1 = new IsoScelesTriangleChain();
        TriangleChain chain2 = new RightTriangleChain();
        TriangleChain chain3 = new BaseTriangleChain();
        chain2.setNext(chain3);
        chain1.setNext(chain2);
        chain0.setNext(chain1);

        for (Triangle triangle : triangleList)
            System.out.println(chain0.getSquare(triangle));
    }
}
```



```
/usr/lib/jvm/java-1.8.0-openjdk-amd64/bin/java ...
Exception in thread "main" определен тип треугольника: Равносторонний треугольник
6.928203230275509
определен тип треугольника: Равнобедренный треугольник
11.313708498984761
определен тип треугольника: Прямоугольный треугольник
7.5
определен тип треугольника: Обычный треугольник
6.48074069840786
java.lang.IllegalArgumentException: площадь не была посчитана
    at problem01.TriangleChain.getSquare(TriangleChain.java:21)
    at problem01.TriangleChain.getSquare(TriangleChain.java:19)
    at problem01.TriangleChain.getSquare(TriangleChain.java:19)
    at problem01.TriangleChain.getSquare(TriangleChain.java:19)
    at problem01.Main.main(Main.java:27)

Process finished with exit code 1
```

Задача 4

```java
public class Article {

    private final int id;
    private final String header;
    private final String body;

    public Article(int id, String header, String body) {
        this.id = id;
        this.header = header;
        this.body = body;
    }


    @Override
    public String toString() {
        return "Article{" +
                "id=" + id +
                ", header='" + header + '\'' +
                ", body='" + body + '\'' +
                '}';
    }
}
```

```java
public class Newspaper {

    private final List<Article> articles;
    private final int id;

    public Newspaper(List<Article> articles, int id) {
        this.articles = articles;
        this.id = id;
    }

    @Override
    public String toString() {
        StringBuilder description = new StringBuilder();
        description.append("Newspaper number ");
        description.append(id);
        description.append(" with articles: ");
        for (Article article : articles) {
            description.append(article.toString());
            description.append("\n");
        }
        return description.toString();
    }
}
```

```
public interface ArticleCallback {

    void provideArticle(Article article);
}
```

```
public class Journalist {

    private static final int ARTICLES_NUMBER_FOR_NEWSPAPER = 4;

    private final List<Article> articles;

    private int newspapersCount;

    public Journalist() {
        articles = new ArrayList<>();
    }

    public void handleArticle(Article article) {
        if (articles.size() == ARTICLES_NUMBER_FOR_NEWSPAPER) {
            System.out.println(new Newspaper(articles,
newspapersCount).toString());
            newspapersCount++;
            articles.clear();
        } else {
            articles.add(article);
        }
    }
}
```

```
public class ArticleFactory {

    private final ArticleCallback articleCallback;
    private final Timer timer;

    private int count;

    public ArticleFactory(ArticleCallback articleCallback) {
        this.articleCallback = articleCallback;
        this.timer = new Timer();
    }

    void start() {
        final TimerTask timerTask = new TimerTask() {
            @Override
            public void run() {
                articleCallback.provideArticle(new Article(count,
"Article header " + count, "Article body"));
                if (++count == 40) {
                    timer.cancel();
                }
            }
        };
        timer.scheduleAtFixedRate(timerTask, 1000, 3000);
    }
}
```

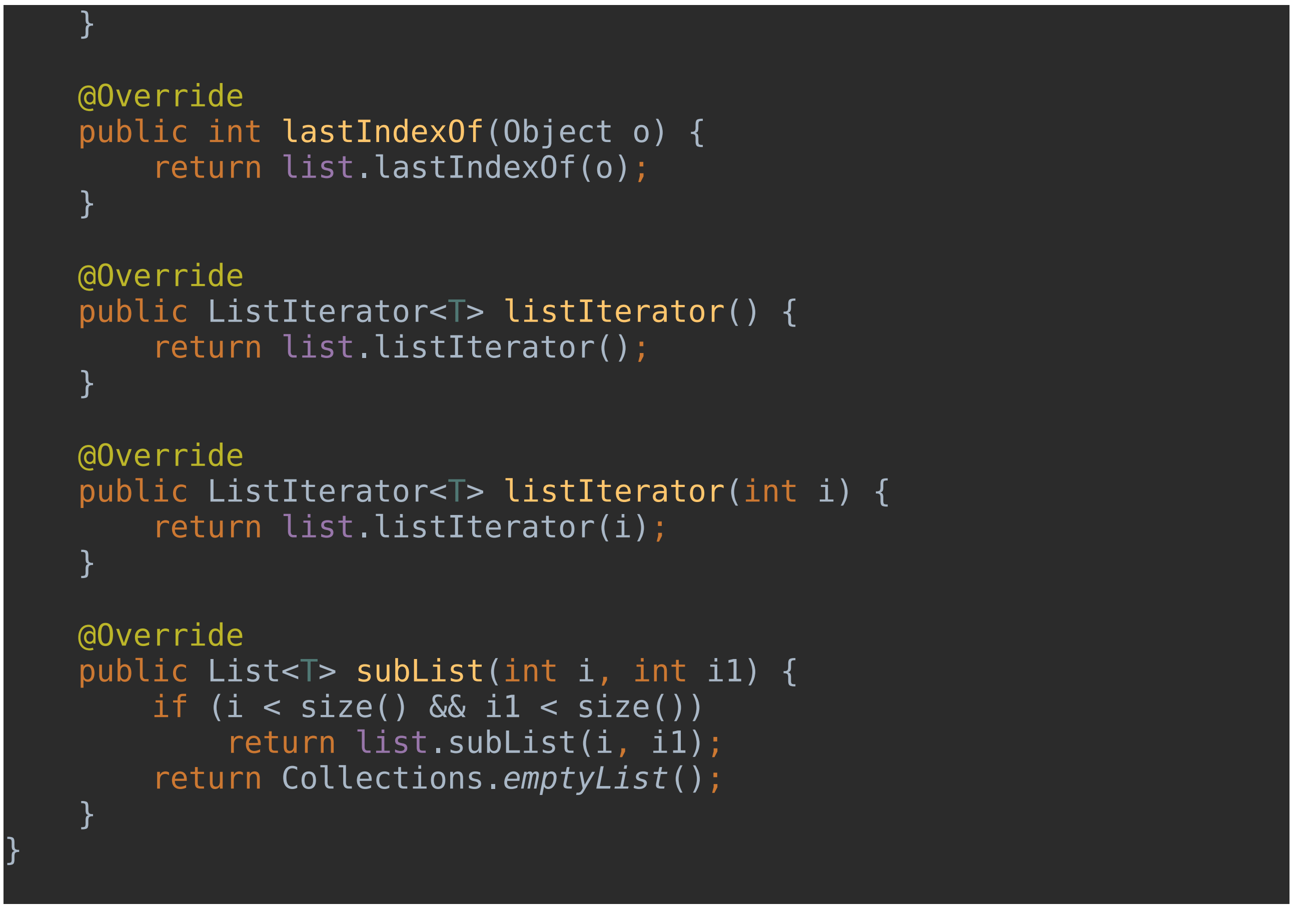
```
public static void main(String[] args) {
    Journalist journalist = new Journalist();
    ArticleCallback callback = journalist::handleArticle;
    ArticleFactory factory = new ArticleFactory(callback);
    factory.start();
}
```

```
Newspaper number 0 with articles: Article{id=0, header='Article header 0', body='Article body'}
Article{id=1, header='Article header 1', body='Article body'}
Article{id=2, header='Article header 2', body='Article body'}
Article{id=3, header='Article header 3', body='Article body'}

…

Newspaper number 7 with articles: Article{id=35, header='Article header 35', body='Article body'}
Article{id=36, header='Article header 36', body='Article body'}
Article{id=37, header='Article header 37', body='Article body'}
Article{id=38, header='Article header 38', body='Article body'}


Process finished with exit code 0
```

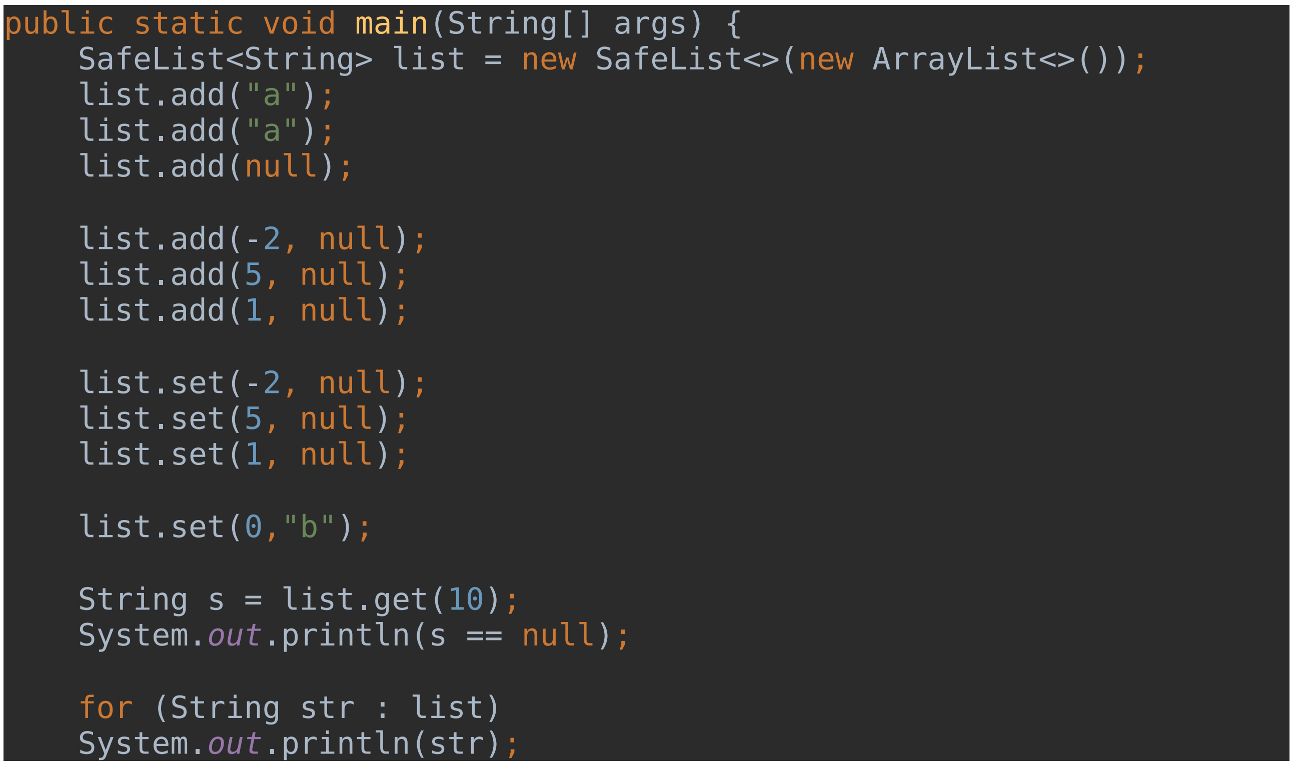
Задача 8

```
public class SafeList<T> implements List<T> {

    private final List<T> list;

    public SafeList(List<T> list) {
        this.list = list;
    }

    @Override
    public int size() {
        return list.size();
    }

    @Override
    public boolean isEmpty() {
        return list.isEmpty();
```

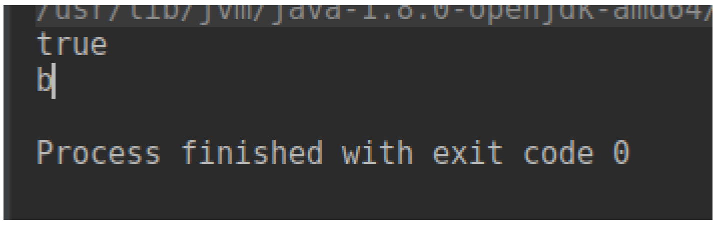
```java
    }

    @Override
    public boolean contains(Object o) {
        return list.contains(o);
    }

    @Override
    public Iterator<T> iterator() {
        return list.iterator();
    }

    @Override
    public Object[] toArray() {
        return list.toArray();
    }

    @Override
    public <T1> T1[] toArray(T1[] t1s) {
        return list.toArray(t1s);
    }

    @Override
    public boolean add(T t) {
        if (t == null)
            return false;
        if (contains(t))
            return false;

        return list.add(t);
    }

    @Override
    public boolean remove(Object o) {
        return list.remove(o);
    }

    @Override
    public boolean containsAll(Collection<?> collection) {
        return list.containsAll(collection);
    }

    @Override
    public boolean addAll(Collection<? extends T> collection) {
        for (T t : collection) {
            add(t);
        }
        return true;
    }

    @Override
    public boolean addAll(int i, Collection<? extends T> collection){
        for (T t : collection) {
            add(i, t);
        }
        return true;
    }

    @Override
    public boolean removeAll(Collection<?> collection) {
        return list.removeAll(collection);
    }

    @Override
    public boolean retainAll(Collection<?> collection) {
        return list.retainAll(collection);
    }

    @Override
    public void clear() {
        list.clear();
    }

    @Override
    public T get(int i) {
        if (isEmpty() || i >= size() || i < 0)
            return null;
        else
            return list.get(i);
    }

    @Override
    public T set(int i, T t) {
        if (isEmpty() || i >= size() || i < 0 || t == null ||
contains(t))
            return null;
        else
            return list.set(i, t);
    }

    @Override
    public void add(int i, T t) {
        if (i >= 0 && i < size()) {
            list.add(i, t);
        }
    }

    @Override
    public T remove(int i) {
        if (i >= 0 && i < size())
            return list.remove(i);
        else
            return null;
    }

    @Override
    public int indexOf(Object o) {
        return list.indexOf(o);
```



```java
    }

    @Override
    public int lastIndexOf(Object o) {
        return list.lastIndexOf(o);
    }

    @Override
    public ListIterator<T> listIterator() {
        return list.listIterator();
    }

    @Override
    public ListIterator<T> listIterator(int i) {
        return list.listIterator(i);
    }

    @Override
    public List<T> subList(int i, int i1) {
        if (i < size() && i1 < size())
            return list.subList(i, i1);
        return Collections.emptyList();
    }
}
```



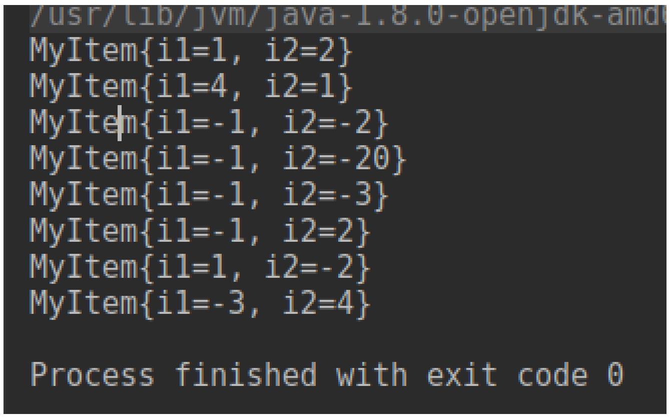
```java
public static void main(String[] args) {
    SafeList<String> list = new SafeList<>(new ArrayList<>());
    list.add("a");
    list.add("a");
    list.add(null);

    list.add(-2, null);
    list.add(5, null);
    list.add(1, null);

    list.set(-2, null);
    list.set(5, null);
    list.set(1, null);

    list.set(0,"b");

    String s = list.get(10);
    System.out.println(s == null);

    for (String str : list)
    System.out.println(str);
```



```
/usr/lib/jvm/java-1.8.0-openjdk-amd64/
true
b

Process finished with exit code 0
```

Задача 10

```
public class Main {

    public static void main(String[] args) {
        List<MyItem> list = new ArrayList<>();
        list.add(new MyItem(-1, -2));
        list.add(new MyItem(-1, 2));
        list.add(new MyItem(1, 2));
        list.add(new MyItem(1, -2));
        list.add(new MyItem(-3, 4));
        list.add(new MyItem(4, 1));
        list.add(new MyItem(-1, -20));
        list.add(new MyItem(-1, -3));

        list.sort(new Comparator<MyItem>() {
            @Override
            public int compare(MyItem next, MyItem current) {
                return current.sign() - next.sign();
            }
        });
        for (MyItem item : list) {
            System.out.println(item);
        }
    }
    static class MyItem {
        final int i1;
        final int i2;

        public MyItem(int i1, int i2) {
            this.i1 = i1;
            this.i2 = i2;
        }

        public int sign() {
            int result;
            if (i1 > 0 && i2 > 0)
                result = 1;
            else if (i1 < 0 && i2 < 0)
                result = 0;
            else
                result = -1;
            return result;
        }

        @Override
        public String toString() {
            return "MyItem{" +
                    "i1=" + i1 +
                    ", i2=" + i2 +
                    '}';
        }
    }
}
```



```
/usr/lib/jvm/java-1.8.0-openjdk-amd
MyItem{i1=1, i2=2}
MyItem{i1=4, i2=1}
MyItem{i1=-1, i2=-2}
MyItem{i1=-1, i2=-20}
MyItem{i1=-1, i2=-3}
MyItem{i1=-1, i2=2}
MyItem{i1=1, i2=-2}
MyItem{i1=-3, i2=4}

Process finished with exit code 0
```