Java Pro 2.1

TEL-RAN
by Starta Institute

Коллекции. Список.

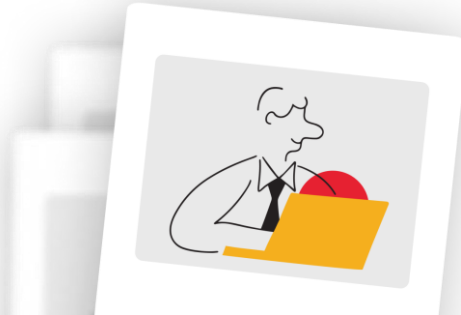
ПРЕПОДАВАТЕЛЬ

TEL-RAN
by Starta Institute

Юрий Костяной

Java/Kotlin backend-разработчик

- 3+ года опыта в коммерческой разработке
- 2+ года опыта в преподавании
- Проекты по интеграции сторонних платформ, CRM
- Проблемно-ориентированный подход в преподавании



ВАЖНО:

- Камера должна быть включена на протяжении всего занятия.
- Если у Вас возник вопрос в процессе занятия, пожалуйста, поднимите руку и дождитесь, пока преподаватель закончит мысль и спросит Вас, также можно задать вопрос в чате или когда преподаватель скажет, что начался блок вопросов.
- Организационные вопросы по обучению решаются с кураторами, а не на тематических занятиях.
- Вести себя уважительно и этично по отношению к остальным участникам занятия.
- Во время занятия будут интерактивные задания, будьте готовы включить камеру или демонстрацию экрана по просьбе преподавателя.

TEL-RAN
by Starta Institute

ПЛАН ЗАНЯТИЯ

1. Основной блок
2. Вопросы по основному блоку
3. Домашняя работа

TEL-RAN
by Starta Institute

TEL-RAN
by Starta Institute

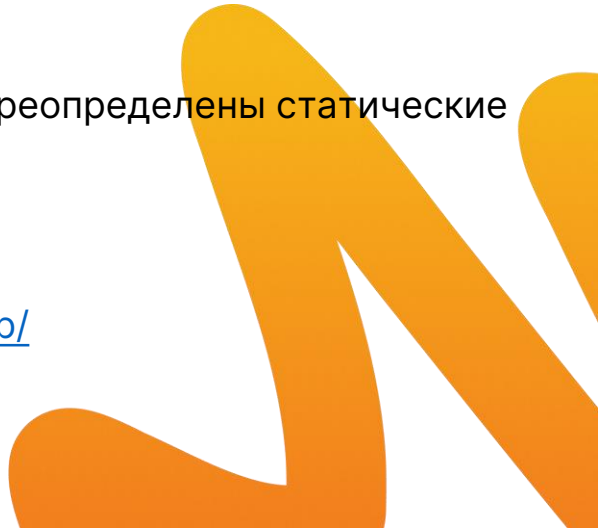
1 ПОВТОРЕНИЕ ИЗУЧЕННОГО

Повторение

TEL-RAN
by Starta Institute

1. Назовите принципы ООП и расскажите о каждом.
2. Дайте определение понятию “класс”.
3. О чем говорят ключевые слова “this”, “super”, где и как их можно использовать?
4. Какие преобразования называются нисходящими и восходящими?
5. Чем отличается переопределение от перегрузки?
6. Зачем нужен оператор instanceof?
7. Каков порядок вызова конструкторов и блоков инициализации двух классов: потомка и его предка?
8. Где и для чего используется модификатор abstract?
9. Расскажите про переопределение методов. Могут ли быть переопределены статические методы?

Ответы https://javastudy.ru/interview/java-oop/



Повторение

TEL-RAN
by Starta Institute

Птицы летают, ящерицы ползают. А птеродактили и летают, и ползают. Что наиболее верно по отношению к этим утверждениям?

A. class Pterodactyl *extends* Crawlable, Flyable{}

B. class Pterodactyl *extends* Crawlable *implements* Flyable{}

C. class Pterodactyl *implements* Crawlable, Flyable{}

D. class Pterodactyl *extends* Flyable *implements* Crawlable{}

Повторение

TEL-RAN
by Starta Institute

Птицы летают, ящерицы ползают. А птеродактили и летают, и ползают. Что наиболее верно по отношению к этим утверждениям?

A. class Pterodactyl *extends* Crawlable, Flyable{}

B. class Pterodactyl *extends* Crawlable *implements* Flyable{}

C. class Pterodactyl *implements* Crawlable, Flyable{}

D. class Pterodactyl *extends* Flyable *implements* Crawlable{}

Повторение

```
public class GoodbyeWorld {
    public static void main(String[] args) {
        System.out.print(A.i);
        System.out.print(B.i);
    }
}

class A {
    static {
        i = 2;
    }
    static int i = 1;
};

class B {
    static int i = 1;
    static {
        i = 2;
    }
};
```

Что будет выведено в консоль?

A. 11

B. 22

C. 21

D. 12

TEL-RAN
by Starta Institute

Повторение

```
public class GoodbyeWorld {
    public static void main(String[] args) {
        System.out.print(A.i);
        System.out.print(B.i);
    }
}

class A {
    static {
        i = 2;
    }
    static int i = 1;
};

class B {
    static int i = 1;
    static {
        i = 2;
    }
};
```

Что будет выведено в консоль?

A. 11

B. 22

C. 21

D. 12

Инициализация всех статических полей и блоки статической инициализации выполняются друг за другом, в том порядке, в котором они записаны. Поэтому в классе A поле i будет иметь значение 1, а в классе B – значение 2.

Повторение

Что будет выведено в консоль?

TEL-RAN
by Starta Institute

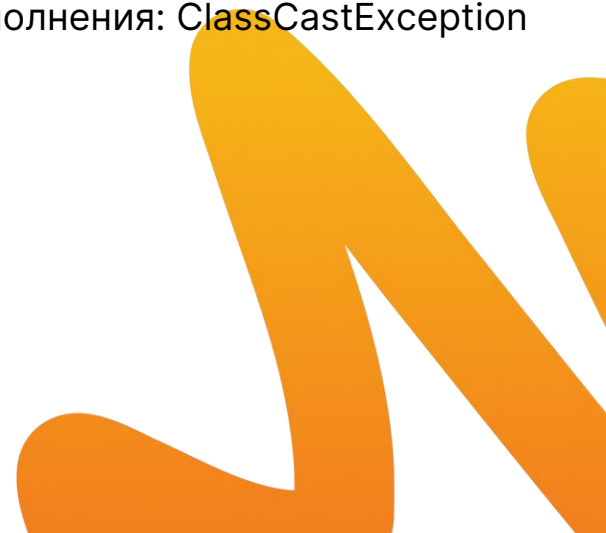
```
class A {
    int i = 0;
    public int increment() {
        return ++i;
    }
}

public class B extends A {
    int i = 10;
    public int increment() {
        return ++i;
    }

    public static void main(String[] args) {
        A obj = (A) new B();
        System.out.println(obj.increment());
    }
}
```

A. 1

B. 11

C. Ошибка компиляции: Incompatible types

D. Ошибка выполнения: ClassCastException



Повторение

Что будет выведено в консоль?

TEL-RAN
by Starta Institute

```
class A {
    int i = 0;
    public int increment() {
        return ++i;
    }
}

public class B extends A {
    int i = 10;
    public int increment() {
        return ++i;
    }

    public static void main(String[] args) {
        A obj = (A) new B();
        System.out.println(obj.increment());
    }
}
```

A. 1

B. 11

C. Ошибка компиляции: Incompatible types

D. Ошибка выполнения: ClassCastException

Метод A.increment() перекрывается в классе B. Поэтому, вне зависимости от типа переменной obj, будет вызываться метод, соответствующий реальному типу объекта - т.е. B.increment(). Этот метод будет использовать переменную i, объявленную в том же классе B и имеющую начальное значение 10. Результирующее значение - 11.

Повторение

Исправьте ошибку в коде

TEL-RAN
by Starta Institute

```
class Class1 {
    Class1(int i) {
        System.out.println("Class1(int)");
    }
}

public class Class2 extends Class1 {
    Class2(double d) {              // 1
        this((int) d);
        System.out.println("Class2(double)");
    }

    Class2(int i) {                 // 2
        System.out.println("Class2(int)");
    }

    public static void main(String[] args) {
        new Class2(0.0);
    }
}
```

Повторение

Исправьте ошибку в коде

TEL-RAN
by Starta Institute

```
class Class1 {
    Class1(int i) {
        System.out.println("Class1(int)");
    }
}

public class Class2 extends Class1 {
    Class2(double d) {              // 1
        this((int) d);
        System.out.println("Class2(double)");
    }

    Class2(int i) {                 // 2
        System.out.println("Class2(int)");
    }

    public static void main(String[] args) {
        new Class2(0.0);
    }
}
```

В строке 2 будет ошибка компиляции, т.к. конструктор должен обязательно вызывать либо другой конструктор этого же класса, либо конструктор суперкласса.

Повторение

Что будет в результате выполнения программы?

TEL-RAN
by Starta Institute

```
class Go extends A {
     public static  void main(String[] args)  {
        new Go().start();
     }

   private void start() {
      check(new A(), new Go());
      check((Go)new A(), new Go());
   }

   private void check(A a, A a1) {
       Go go = (Go) a;  // 1
       A a2 = (A) a1;   //  2
   }
}

class A{

}
```

A. Ошибка компиляции

B. Ошибка выполнения в строке 1

C. Код успешно скомпилируется и выполнится

D. Ошибка выполнения в строке 2

Повторение

Что будет в результате выполнения программы?

TEL-RAN by Starta Institute

```
class Go extends A {
     public static  void main(String[] args)  {
        new Go().start();
     }

   private void start() {
      check(new A(), new Go());
      check((Go)new A(), new Go());
   }

   private void check(A a, A a1) {
       Go go = (Go) a;  // 1
       A a2 = (A) a1;   //  2
   }
}

class A{

}
```

A. Ошибка компиляции

B. Ошибка выполнения в строке 1

C. Код успешно скомпилируется и выполнится

D. Ошибка выполнения в строке 2

Преобразование переменной a типа A к типу наследника Go не может быть выполнено, но исключение будет брошено только на этапе выполнения.

Повторение

Что будет в результате выполнения программы?

TEL-RAN
by Starta Institute

```
public class Funcs extends java.lang.Math {
    public int add(int x, int y) {
        return x + y;
    }
    public int sub(int x, int y) {
        return x - y;
    }
    public static void main(String[] a) {
        Funcs f = new Funcs();
        System.out.println("" + f.add(1, 2));
    }
}
```

A. Код не скомпилируется

B. Код скомпилируется, но ничего не напечатает

C. Код скомпилируется и напечатает 3

D. Код скомпилируется, но во время выполнения возникнет исключение

Повторение

Что будет в результате выполнения программы?

TEL-RAN
by Starta Institute

```
public class Funcs extends java.lang.Math {
    public int add(int x, int y) {
        return x + y;
    }
    public int sub(int x, int y) {
        return x - y;
    }
    public static void main(String[] a) {
        Funcs f = new Funcs();
        System.out.println("" + f.add(1, 2));
    }
}
```

A. Код не скомпилируется

B. Код скомпилируется, но ничего не напечатает

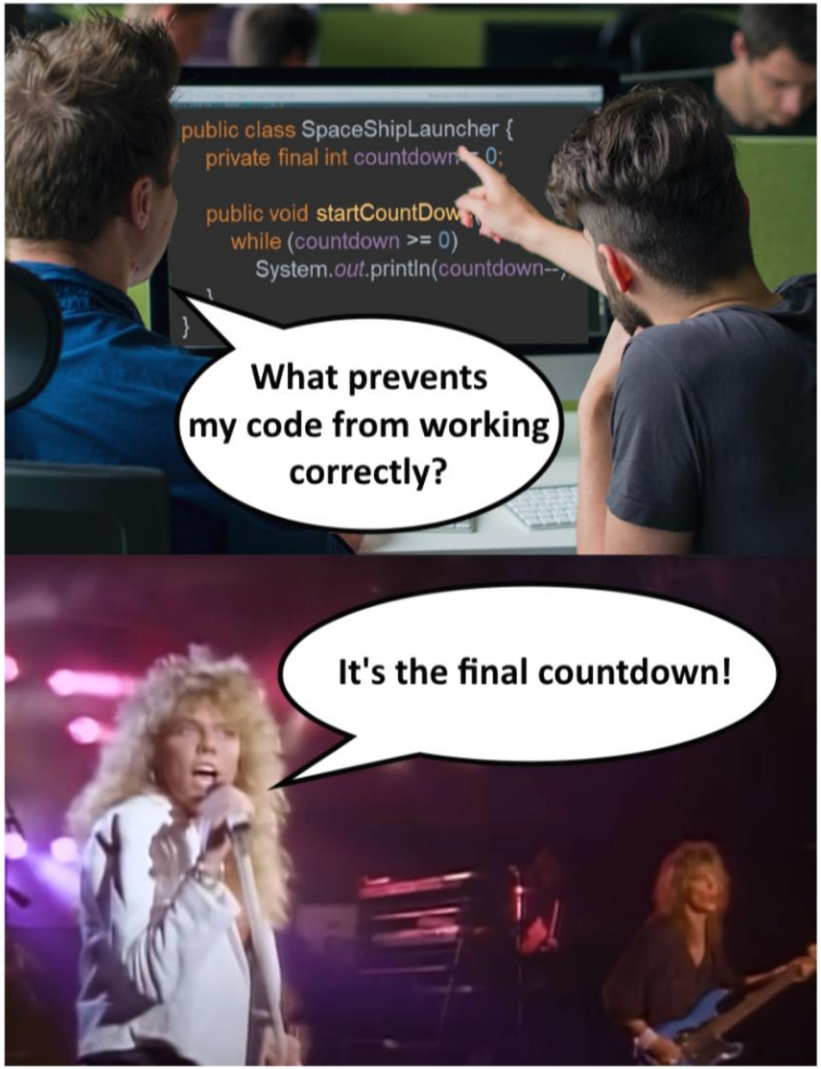
C. Код скомпилируется и напечатает 3

D. Код скомпилируется, но во время выполнения возникнет исключение

Класс java.lang.Math объявлен как final, то есть наследоваться от него нельзя.

Повторение

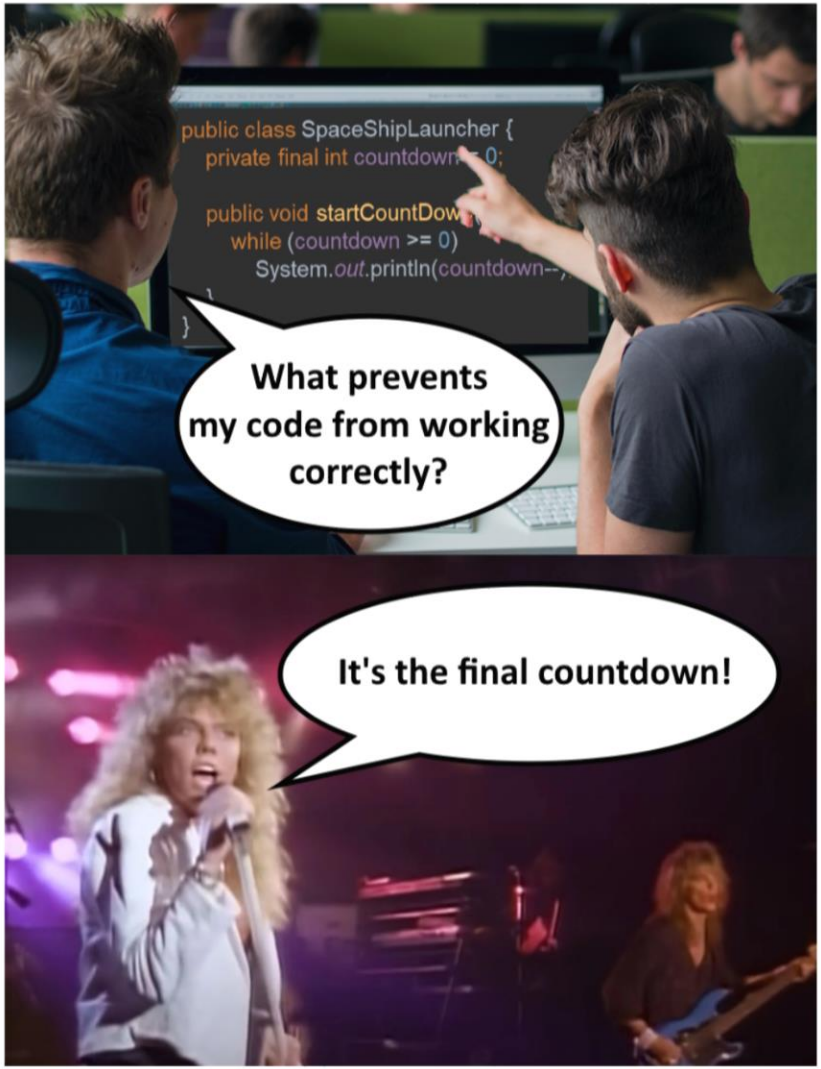
В чём прикол мема?



Повторение

В чём прикол мема?

```
public class SpaceShipLauncher {
    private final int countdown = 0;

    public void startCountDown() {
        while (countdown >= 0)
            System.out.println(countdown--);
    }
}
```

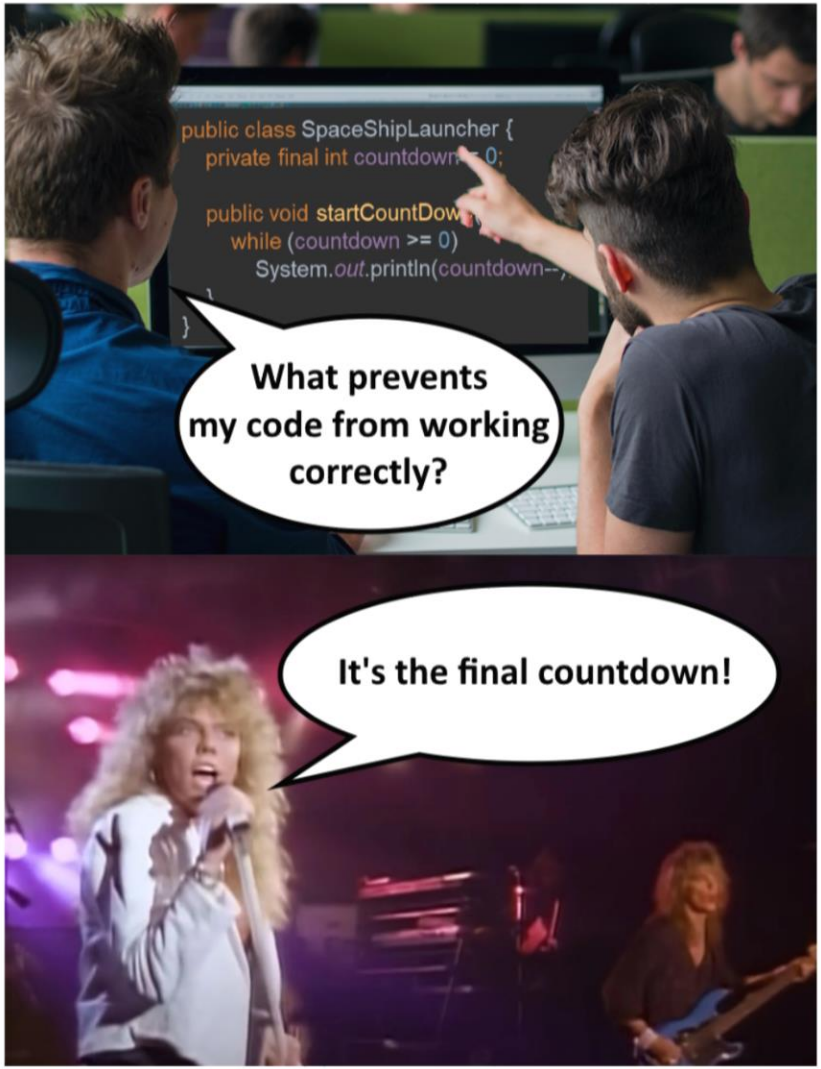


Повторение

В чём прикол мема?

```
public class SpaceShipLauncher {
    private final int countdown = 0;

    public void startCountDown() {
        while (countdown >= 0)
            System.out.println(countdown--);
    }
}
```

Cannot assign a value to final variable 'countdown'



TEL-RAN
by Starta Institute

2 ОСНОВНОЙ БЛОК

Введение

TEL-RAN
by Starta Institute

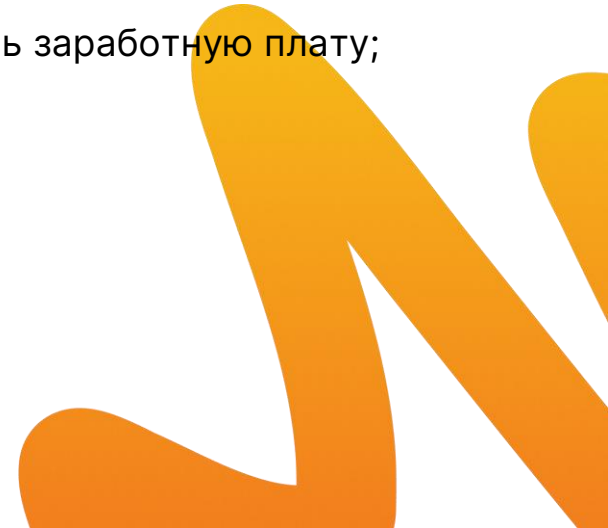
- Перечисляя то, что было
- Полиморфы атакуют!
- Лицом к лицу

Проблема

TEL-RAN
by Starta Institute

Необходимо решить задачи по хранению данных в разных сферах жизни:

- данные о том, на какое занятие какой из студентов записался, чтобы преподаватели знали состав групп;
- данные о том, кто первым из пациентов попадёт к врачу;
- данные о том, кому из сотрудников фирмы нужно выплатить заработную плату;
- бумажные карты пациентов в поликлинике;
- населённые пункты и дороги между ними.



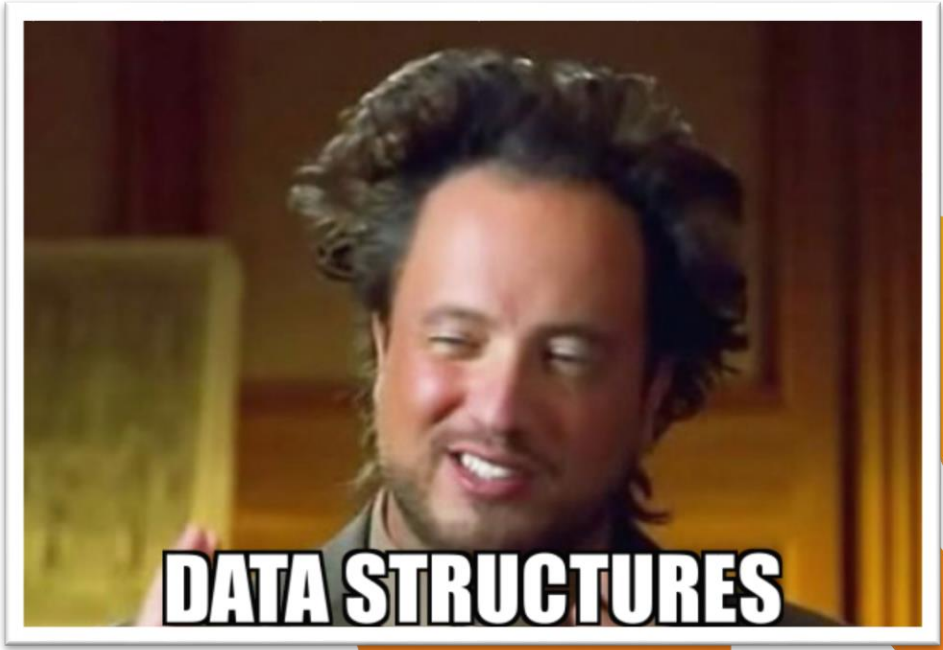
По полочкам

Структуры данных

TEL-RAN
by Starta Institute

Структура данных – это способ хранения и организации наборов данных таким образом, чтобы данные можно было быстро:

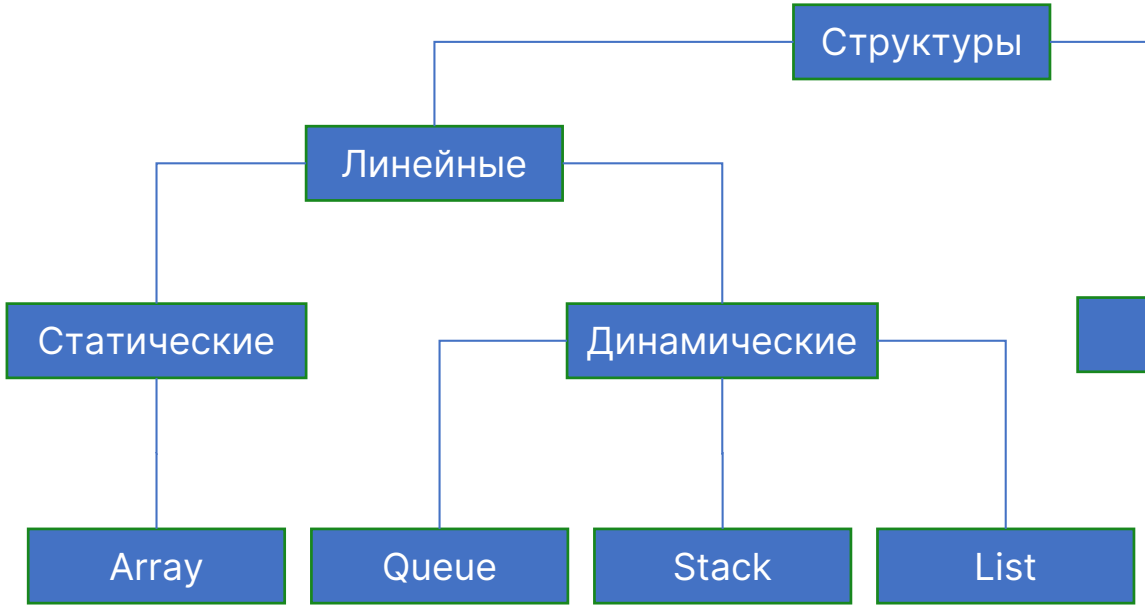
- получить из структуры и обработать
- обновлять и добавлять
- искать в структуре



По полочкам

Структуры данных

TEL-RAN
by Starta Institute



По полочкам

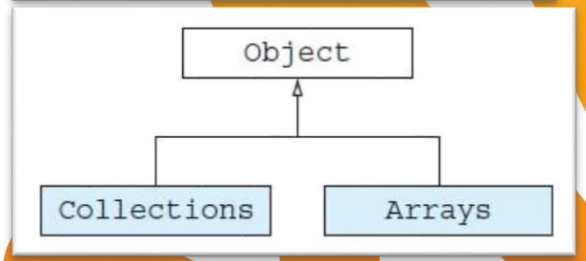
Коллекции Java

TEL-RAN
by Starta Institute

Простейшая структура данных – это *переменная*. *Массив* – это набор однотипных переменных. *Экземпляр класса* – это набор разнотипных переменных.

Коллекции – это реализация структур данных в виде классов *Java Collection Framework (JCF)*. Коллекции хранят наборы объектов. Далее мы будем говорить о структурах данных и их реализациях в JCF.

Java Collection Framework – это собрание классов и интерфейсов в Java, предназначенных для хранения и обработки данных в оперативной памяти



По полочкам

Зачем нужны коллекции, если есть массивы?

TEL-RAN
by Starta Institute

Сравним массив с ближайшей по функциональности коллекцией – *List*:

1. Размер массива фиксирован. В *List* можно добавлять и удалять элементы динамически.
2. Скорость доступа к элементу массива – фиксирована, в *List* – зависит от реализации списка (фиксированная или хуже).
3. Использование в режиме «только для чтения» в массиве отсутствует, т.е. поученный массив можно изменить, даже если бы мы этого не хотели. У *List* есть *immutable*-обертка, позволяющая запретить редактирование *List*.
4. Массив может содержать как примитивные, так и ссылочные типы. Коллекции могут содержать только ссылочные типы, поэтому для хранения примитивных типов используются обёртки (*Long*, *Integer*, *Character* и т.д.)

По полочкам

Зачем нужны коллекции, если есть массивы?

TEL-RAN
by Starta Institute

5. Коллекции часто работают медленнее простого массива, но предоставляют **дополнительные удобные способы хранения данных и методы доступа**:

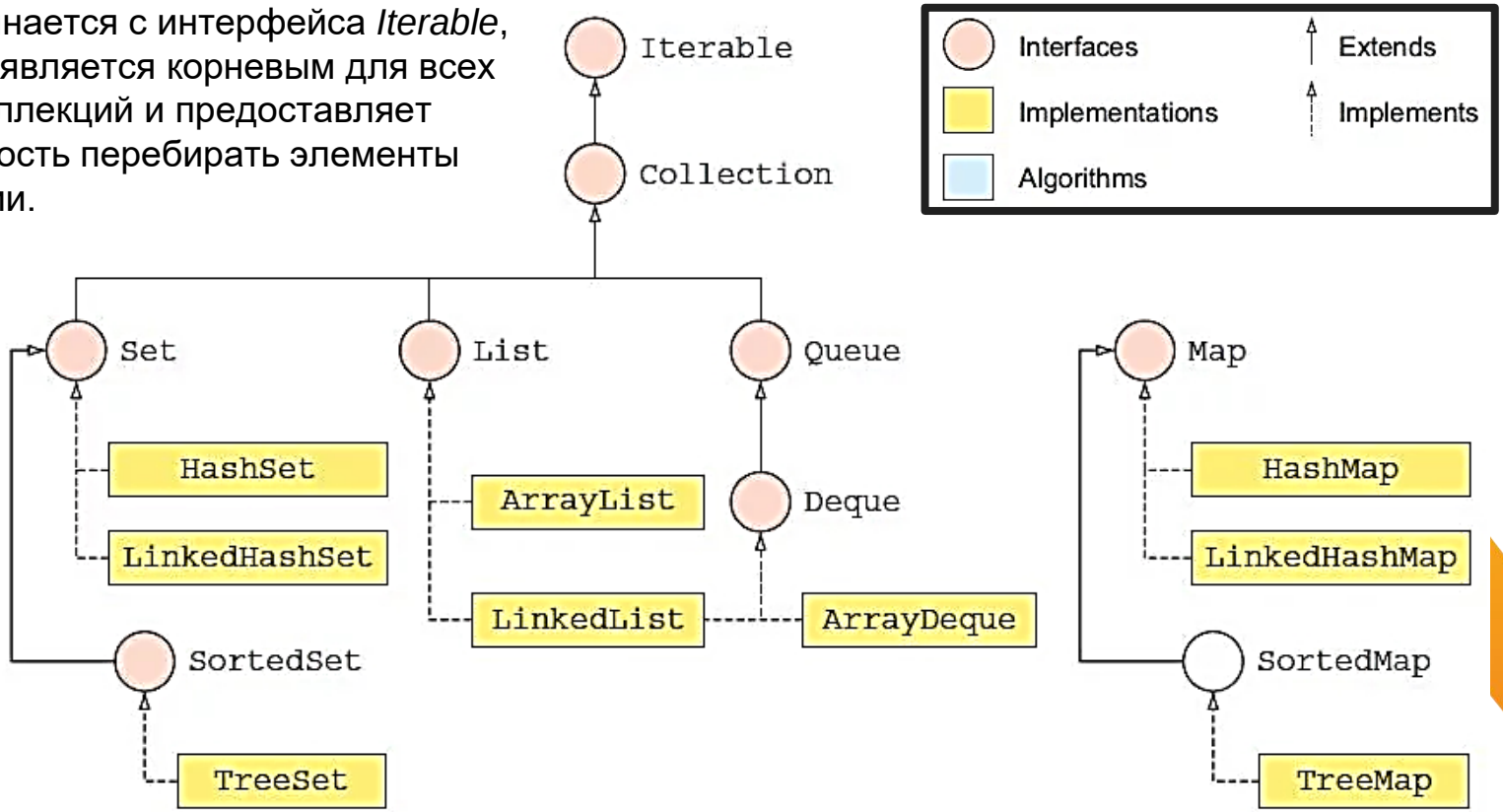
- Простые способы добавления и удаления элементов.
- Разные структуры данных для хранения (словарь, дерево, множество и др.)
- Служебные методы. Например, *toString* выводит состав коллекции, а *equals* сравнивает состав и порядок коллекции, а не только ссылки.
- Обход в циклах в нужном порядке с помощью вспомогательного класса *Iterator*. Такой класс можно написать самому, если нужен особенный порядок обхода коллекции.

По полочкам

Иерархия коллекций

TEL-RAN
by Starta Institute

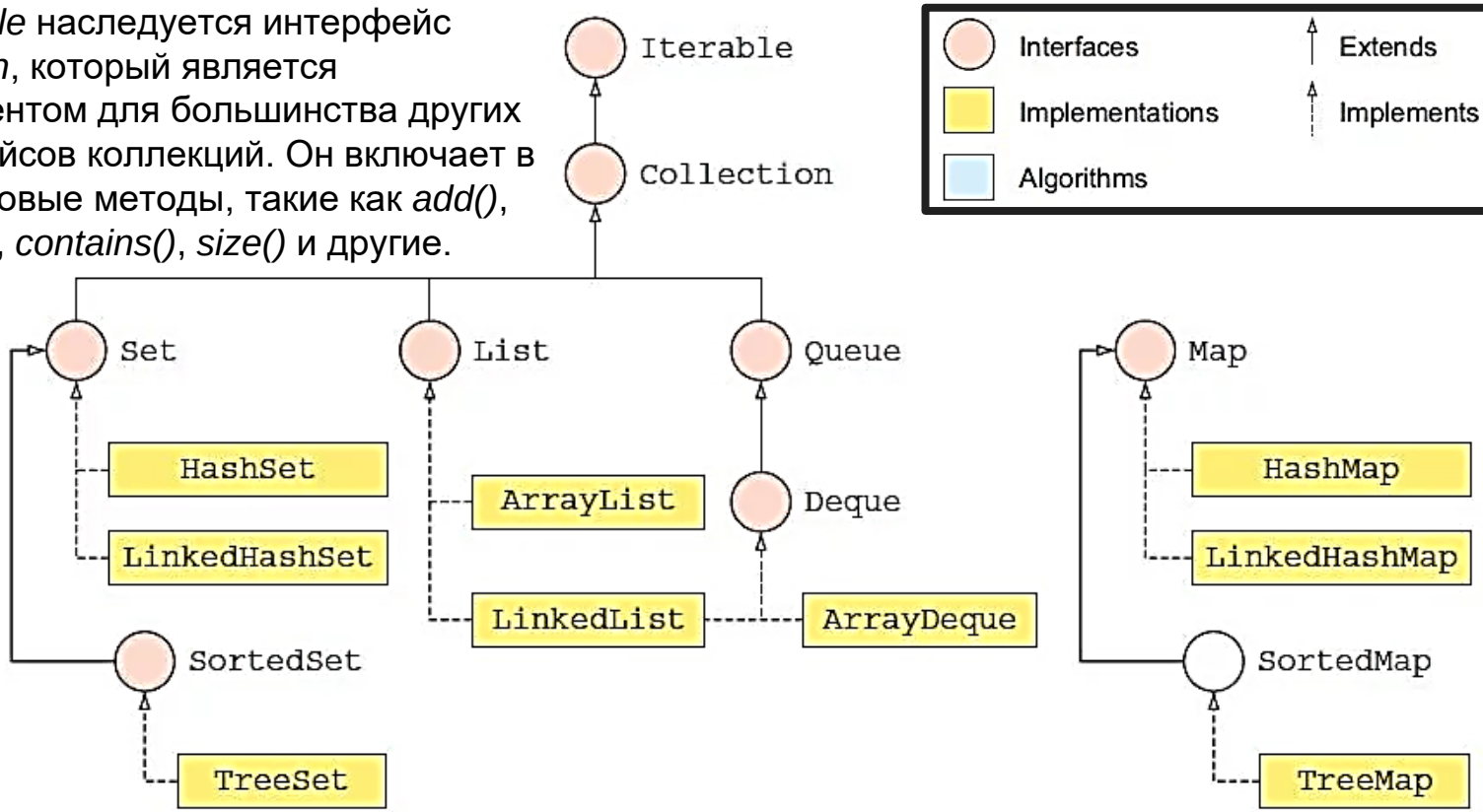
Все начинается с интерфейса *Iterable*, который является корневым для всех типов коллекций и предоставляет возможность перебирать элементы коллекции.



По полочкам

Иерархия коллекций

От *Iterable* наследуется интерфейс *Collection*, который является фундаментом для большинства других интерфейсов коллекций. Он включает в себя базовые методы, такие как *add()*, *remove()*, *contains()*, *size()* и другие.



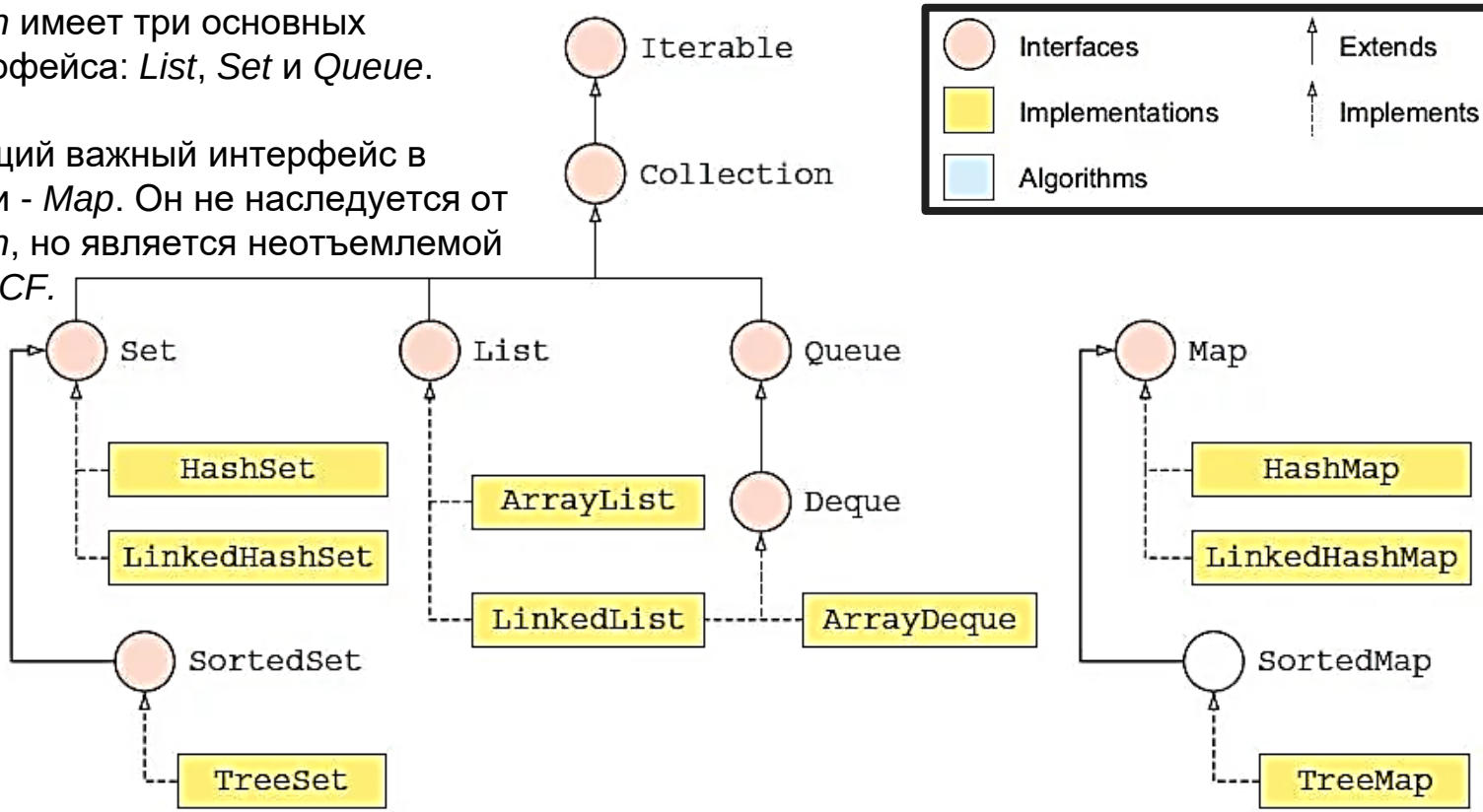
По полочкам

Иерархия коллекций

TEL-RAN
by Starta Institute

Collection имеет три основных подинтерфейса: *List*, *Set* и *Queue*.

Следующий важный интерфейс в иерархии - *Map*. Он не наследуется от *Collection*, но является неотъемлемой частью *JCF.*



По полочкам

Иерархия коллекций

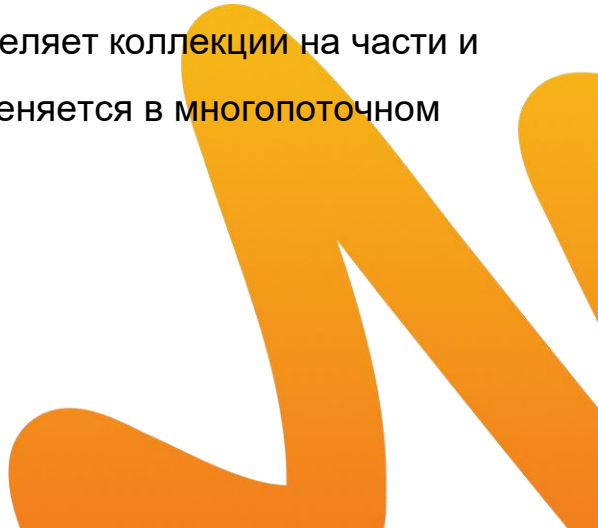
TEL-RAN
by Starta Institute

Базовые методы *Iterable*:

forEach() – выполнение заданного действия для каждого из элементов коллекции, пока не поучаствуют все элементы или не будет брошено исключение.

iterator() – возвращает объект итератора (класса, который позволяет обходить коллекцию поэлементно)

spliterator() – возвращает объект сплитератора (класса, который разделяет коллекции на части и организует обход элементов в заданном порядке). Сплитератор применяется в многопоточном программировании.



По полочкам

Иерархия коллекций

TEL-RAN
by Starta Institute

Базовые методы *Collection*:

add() – добавление элемента в коллекцию

remove() – удаление элемента из коллекции

removeAll() – удаляет все элементы, которые содержатся в переданной коллекции

removeIf() – удаляет все элементы, удовлетворяющие заданному условию

contains() – проверка, содержит ли коллекция такой элемент

containsAll() – проверка, что коллекция содержит все элементы переданной коллекции

size() – возвращает количество элементов в коллекции

addAll() – добавляет все элементы переданной коллекции в текущую коллекцию

clear() – очищает коллекцию

isEmpty() – проверяет, что коллекция пуста

retainAll() – удаляет все элементы, кроме тех, что содержатся в заданной коллекции

stream() – создаёт поток (конвейер) для обработки элементов коллекции (*изучим позже*)

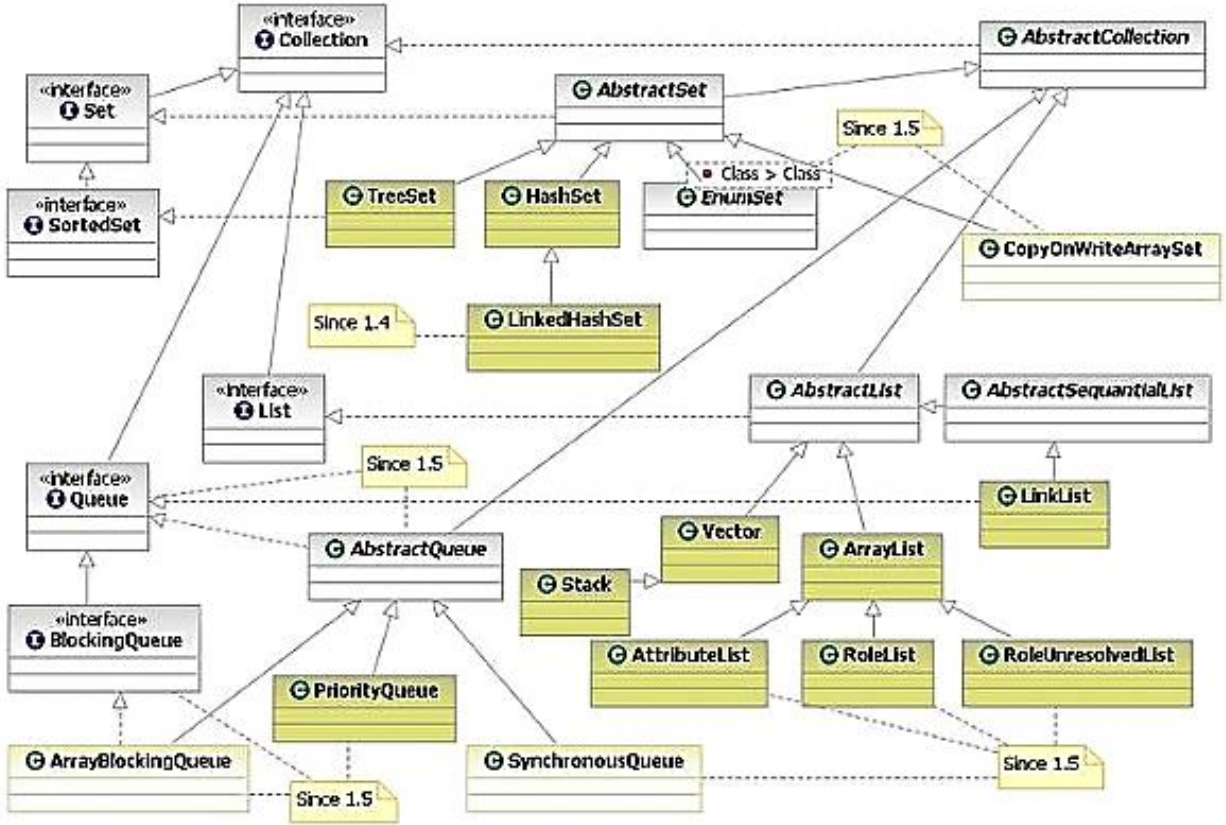
toArray() – превращает коллекцию в массив с типом *Object[]*.

toArray(T[] a) – превращает коллекцию в массив с типом *T[]*, где T – тип элементов коллекции

По полочкам

Иерархия коллекций (подробно)

TEL-RAN
by Starta Institute



Списываем

List (список)

TEL-RAN
by Starta Institute

List – интерфейс, описывающий упорядоченный список (последовательность), в котором у каждого элемента есть индекс (порядковый номер). Дубликаты значений допускаются.

Например, последовательность букв в слове: буквы могут повторяться, при этом их порядок важен.

Списываем

Методы List

TEL-RAN
by Starta Institute

add(int index, object obj) – вставляет элемент *obj* в позицию *index*. Старые элементы, начиная с позиции index, сдвигаются, их индексы увеличиваются на единицу.

addAll(int index, Collection coll) – вставляет все элементы коллекции *coll*

get(int index) – получение элемента из списка по указанному индексу.

indexOf(Object obj) – возвращает индекс первого вхождения указанного элемента в этом списке или -1, если этот список не содержит элемента.

lastindexOf(object obj) – возвращает индекс последнего вхождения указанного элемента в этом списке или -1, если этот список не содержит элемента.

set(int index, object obj) – замена элемента в этом списке на указанной позиции на переданный элемент.

subList(int from, int to) – возвращает часть коллекции от позиции *from* включительно до позиции *to* исключительно.

List.of() – метод для явного создания неизменяемого списка.

Списываем

Задание

TEL-RAN
by Starta Institute

1 Создайте 5 переменных типа *int*.

2 Из созданных переменных создайте список с помощью *List.of()*.

3 Проверьте, содержит ли список число 5.

4 Организуйте цикл по списку с помощью цикла *for*. Для получения элемента используйте метод get().

5 Получите список, состоящий из первых трёх элементов исходного списка.

6 Попробуйте добавить элемент в список с помощью метода add(). Почему было получено исключение?

Списываем
ArrayList

TEL-RAN
by Starta Institute

Класс **ArrayList** – реализация интерфейса *List*, которая подойдёт в большинстве случаев.

Строится на базе *обычного массива*. Если при создании не указать размерность, то под значения выделяется 10 ячеек. При попытке добавить элемент, для которого места уже нет, массив автоматически расширяется – программисту об этом специально заботиться не нужно. Обычно при заполнении массива «под капотом» *ArrayList* создаётся новый массив с двойной длинной.

```
List<Animal> arList = new ArrayList<>();
```

- Тип интерфейса переменной
- Тип хранимых в коллекции элементов
- Название переменной
- Повторно тип элементов можно не указывать
- Конкретная реализация (конструктор класса)
- Оператор создания объекта

Списываем
ArrayList

TEL-RAN
by Starta Institute

Класс **ArrayList** – реализация интерфейса *List*, которая подойдёт в большинстве случаев.

Строится на базе *обычного массива*. Если при создании не указать размерность, то под значения выделяется 10 ячеек. При попытке добавить элемент, для которого места уже нет, массив автоматически расширяется – программисту об этом специально заботиться не нужно. Обычно при заполнении массива «под капотом» *ArrayList* создаётся новый массив с двойной длинной.

```
List<Animal> arList = new ArrayList<>();
```

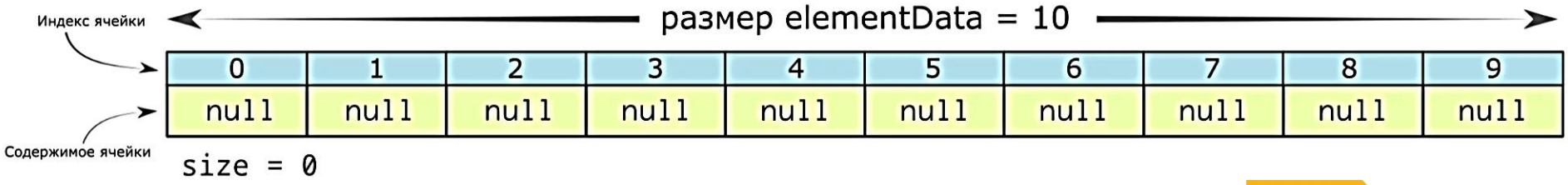
- Не является потокобезопасным;
- Быстрый доступ к элементам по индексу за время O(1);
- Доступ к элементам по значению за линейное время O(n);
- Позволяет хранить любые значения в том числе и *null*;

Списываем
ArrayList

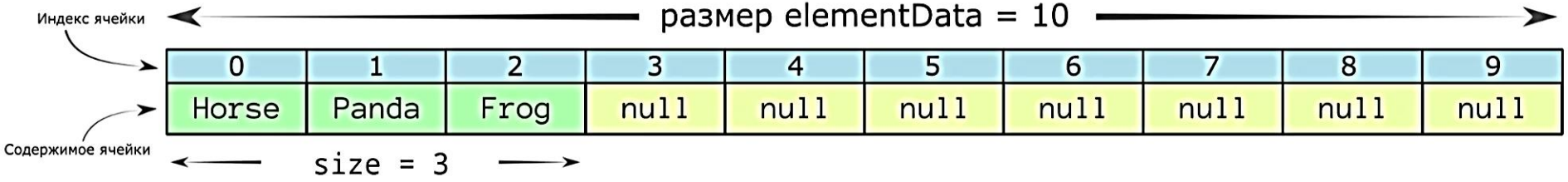
TEL-RAN
by Starta Institute

При включении нового элемента в середину **ArrayList** все элементы с большим индексом сдвигаются вправо:

При создании списка используя конструктор по умолчанию:



После добавления трех элементов в список:



arList.add(new Panda()); // добавление в конец списка
arList.add(1, new Panda()); // добавление в нужную позицию

Списываем
ArrayList

TEL-RAN
by Starta Institute

При удалении элемента **ArrayList** все остальные с бóльшим индексом сдвигаются влево:

0	1	2	3	4	5	6	7	8	9
Horse	~~Panda~~	Frog	null	null	null	null	null	null	null

size = 3

0	1	2	3	4	5	6	7	8	9
Horse	Frog	null	null	null	null	null	null	null	null

size = 2

arList.remove(panda); // удаление по значению элемента (по equals)
arList.remove(1); // удаление по индексу элемента

Списываем

Методы ArrayList

TEL-RAN
by Starta Institute

Эти методы есть только у *ArrayList* и отсутствуют у *List*.

Размер внутреннего массива автоматически увеличивается, но никогда не уменьшается, поэтому если из массива часто удаляются элементы, и при этом не добавляется новых, имеет смысл использовать метод *trimToSize()*.

ensureCapacity(int minCapacity) – увеличивает ёмкость внутреннего массива объекта *ArrayList*, если необходимо, чтобы гарантированно хранить не меньше *minCapacity* элементов.

Клонирующий конструктор создаёт новый *ArrayList* с переданным набором элементов (с набором ссылок на те же объекты другой коллекции).

Списываем

Методы ArrayList

TEL-RAN
by Starta Institute

```
import java.util.ArrayList;


public class SimpleTestArrayList {


   public static void main (String[] args){
       ArrayList<String> list = new ArrayList<>();
       list.add("test1");
       list.add("test2");
       list.add("test3");
       System.out.print(list.get(1)+":");
       list.add(1, "test4");
       System.out.print(list.get(1)+":");
       for (int i=0; i< list.size(); i++){
           System.out.print(list.get(i)+":");
       }
   }
}
```

Вывод: test2:test4:test1:test4:test2:test3:

Элементы в *ArrayList* нумеруются начиная с нуля. Поэтому элемент с номером 1 – это test2. Следующим действием мы добавляем строку «test4» в ячейку с индексом 1. При этом элементы с бóльшим индексом сдвигаются вправо. Вторая часть вывода (test4) показывает, что теперь по индексу 1 извлекается именно test4. Далее мы обходим все элементы списка и убеждаемся, что они выводятся именно в порядке добавления.

Списываем

Методы ArrayList

TEL-RAN
by Starta Institute

```java
import java.util.ArrayList;


public class TestArrayList {
    public static void main(String args[]) {
        ArrayList<String> list = new ArrayList<>();
        String t1 = "test1";
        String t2 = "test2";
        list.add(t1);
        list.add(t2);
        System.out.print(list.size() + ":");
        t1 = "test3";
        list.remove(t1);
        System.out.print(list.size());
    }
}
```

Вывод: 2:2

Первая часть: добавили два элемента, поэтому размер списка равен двум.

Перед удалением элемента его нужно найти в списке. *ArrayList* и остальные коллекции, которые не используют алгоритмы хеширования, применяют для поиска метод equals().

Строки сравниваются по значению, поэтому «test3» не эквивалентно «test1» и «test2». А раз ни один элемент не соответствует критерию поиска, ничего не удалится - размер списка останется прежним.

Списываем

Задание

1 Создать класс *HeavyBox*. Класс содержит поля уникального идентификатора, массы и размеров коробки.

2 Создать *ArrayList*, содержащий не менее трёх объектов класса *HeavyBox*.

3 Распечатать его содержимое.

4 Изменить вес первого ящика на 1.

5 Удалить последний ящик.

6 Получить массив содержащий ящики из коллекции и вывести его в консоль.

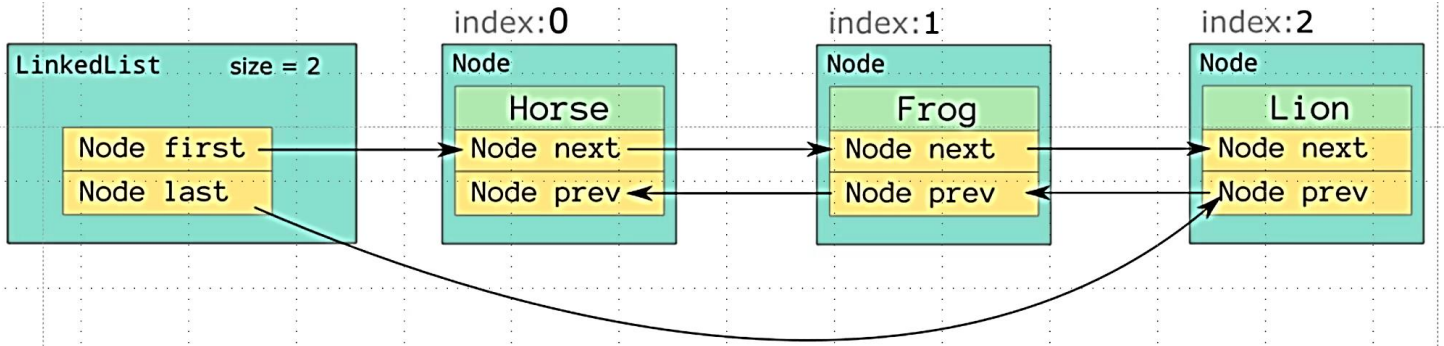
7 Увеличить габариты каждого ящика в 2 раза, используя цикл foreach или метод forEach().

8 Удалить все ящики

Списываем
LinkedList

TEL-RAN
by Starta Institute

Связный список – это структура данных, где каждый элемент является отдельным объектом. Связанный список состоит из двух элементов: данных и ссылки на следующий узел.
Класс **LinkedList** реализует одновременно интерфейсы *List* и *Deque*. Это список, в котором у каждого элемента есть ссылка на предыдущий и следующий элементы:



Благодаря этому добавление и удаление элементов выполняется быстро - времязатраты не зависят от размера списка, так как элементы при этих операциях не сдвигаются: просто перестраиваются ссылки.

Конкретная реализация (конструктор класса)

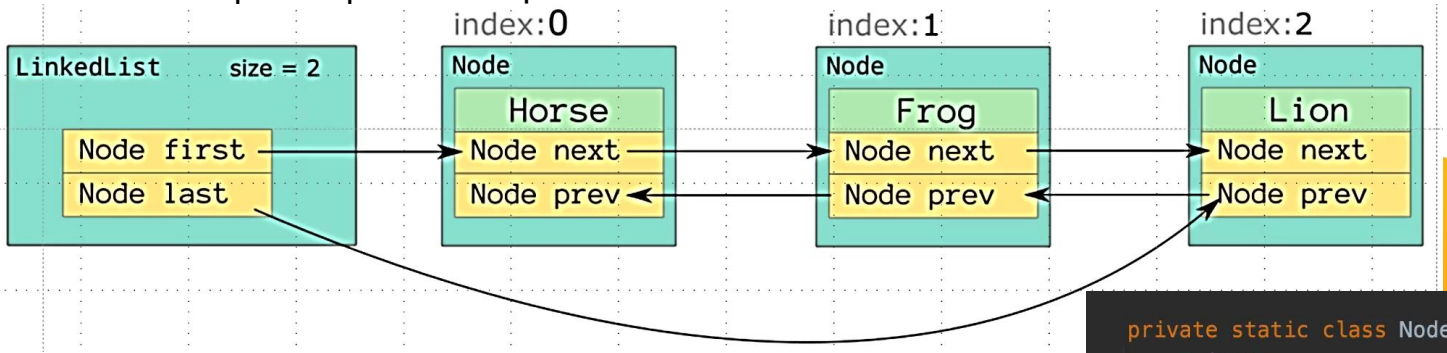
```
List<Animal> lkList = new LinkedList<>();
```

Списываем
LinkedList

TEL-RAN
by Starta Institute

Класс *LinkedList* содержит три основных поля:

1. *Node<E> first* – ссылка на первый элемент списка
2. *Node<E> last* – ссылка на второй элемент списка
3. *int size* – размер коллекции.



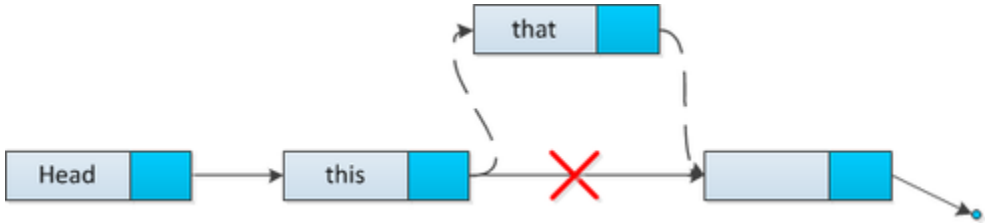
Каждый узел (Node) в LinkedList хранит два элемента: данные и ссылку на следующий и предыдущий узел.

```
private static class Node<E> {
    E item;
    Node<E> next;
    Node<E> prev;

    Node(Node<E> prev, E element, Node<E> next) {
        this.item = element;
        this.next = next;
        this.prev = prev;
    }
}
```

Списываем
LinkedList

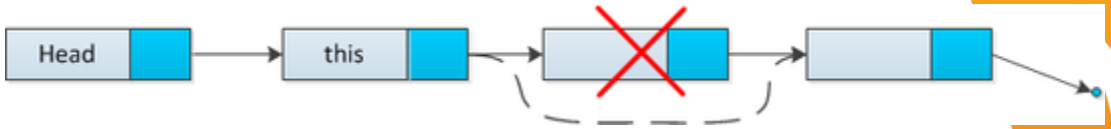
TEL-RAN
by Starta Institute

Добавление элемента



Удаление первого элемента

Удаление элемента из середины

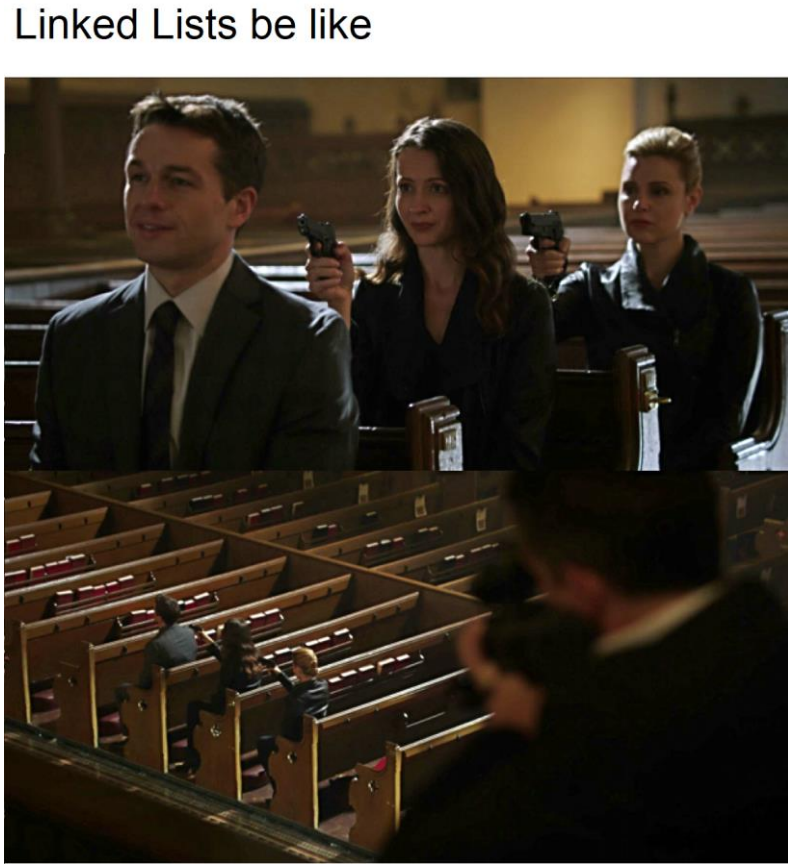


Списываем
LinkedList

TEL-RAN
by Starta Institute

Характеристики *LinkedList*:

- не является потокобезопасным;
- позволяет хранить любые объекты, в том числе *null* и повторяющиеся;
- за константное время O(1) выполняются операции вставки и удаления первого и последнего элемента, операции вставки и удаления элемента из середины списка. Не учитывая время поиска позиции элемента, который осуществляется за линейное время O(n);
- за линейное время O(n) выполняются операции поиска элемента по индексу и по значению.



Списываем

ArrayList vs LinkedList

TEL-RAN
by Starta Institute

На собеседованиях часто спрашивают, **когда выгоднее использовать LinkedList**, а когда - **ArrayList**.
Правильный ответ таков: если добавлять и удалять элементы с произвольными индексами в списке нужно чаще, чем перебирать элементы, то лучше *LinkedList*. В остальных случаях - *ArrayList*.

Однако при добавлении элементов в *ArrayList* вызывается нативный метод *System.arraycopy*. В нём используются ассемблерные инструкции для копирования блоков памяти. Так что даже для больших массивов эти операции выполняются за приемлемое время.

```
public void add(int index, E element) {
    rangeCheckForAdd(index);

    ensureCapacityInternal(size + 1);  // Increments modCount!!
    System.arraycopy(elementData, index, elementData, index + 1,
                     size - index);
    elementData[index] = element;
    size++;
}
```

Списываем

Задание

TEL-RAN
by Starta Institute

Создайте *LinkedList*, положить в переменную *List<String>*. Наполните элементами.

Добавьте 2-3 элемента в середину.

Организуйте цикл по списку с помощью цикла *foreach*.

Списываем

Дополнительные задания

TEL-RAN
by Starta Institute

Напишите свою реализацию связанного списка для типа String (не обязательно наследовать какие-либо интерфейсы).

TEL-RAN
by Starta Institute

3 ВОПРОСЫ ПО ОСНОВНОМУ БЛОКУ

TEL-RAN
by Starta Institute

4 Домашнее задание

Домашнее задание

TEL-RAN
by Starta Institute

№1

Создайте программу, которая принимает от пользователя неограниченное количество строк. Ввод строк должен закончится, когда пользователь введёт слово quit. Выведите в консоль все строки, которые ввёл пользователь. Реализуйте два метода – один находит самую длинную строку в списке, второй – самую короткую строку. Выведите самую длинную и самую короткую из строк в консоль.

№ 2

Напишите метод, который создаёт список с произвольным количеством элементов. Список должен быть заполнен случайными числами в диапазоне от -100 до 100 включительно. Напишите второй метод, который принимает список чисел и удаляет из него все отрицательные числа. В main вызовите оба метода.

№3

Создайте класс Group, который хранит фамилии всех студентов в учебной группе. В классе напишите метод, который по заданным первым буквам фамилии находит всех студентов и возвращает результат в виде списка. Вызовите метод в main.

Дополнительная практика

TEL-RAN
by Starta Institute

1 Создайте класс Контейнер, который имеет внутренние габариты и максимальную массу хранимого груза, а также хранит набор грузов (*HeavyBox*). Пусть все грузы имеют одинаковый размер и массу. Создайте метод в классе Контейнер, который будет принимать груз до тех пор, пока внутренне пространство контейнера не будет заполнено или не будет достигнута максимальная масса груза, допустимого для контейнера.

2 В *main* создайте несколько контейнеров разного размера. Затем создавайте грузы и заполняйте контейнеры, пока все контейнеры не будут максимально загружены.

3 Выведите в консоль, сколько грузов с какими идентификаторами были загружены в какой каждый контейнер.

ЗАКЛЮЧЕНИЕ

TEL-RAN
by Starta Institute