Java Pro 2.2

TEL-RAN
by Starta Institute

Множество. Сравнение объектов

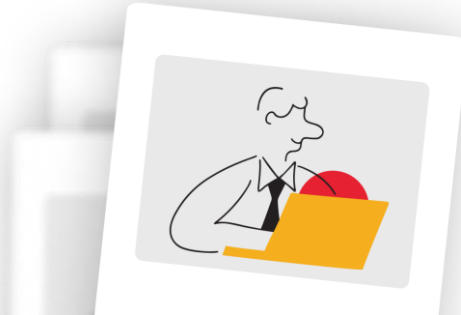
ПРЕПОДАВАТЕЛЬ

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. Дайте определение понятию “коллекция”.
2. Назовите преимущества использования коллекций.
3. Какие данные могут хранить коллекции?
4. Какова иерархия коллекций?
5. Что вы знаете о коллекциях типа List
8. Что разного/общего у классов ArrayList и LinkedList, когда лучше использовать ArrayList, а когда LinkedList?
9. В каких случаях разумно использовать массив, а не ArrayList?

Ответы https://javastudy.ru/interview/collections/

Повторение

TEL-RAN
by Starta Institute

Что будет выведено в консоль?

```
import java.util.List;

public class Main {
    public static void main(String[] args) {
        System.out.println("Начало программы");
        List<Integer> nums = List.of(-1, 0, 1);
        nums.set(0, Math.abs(nums.get(0)));
        System.out.println(nums);
    }
}
```

A.
Начало программы
[1, 0, 1]

B.
Ошибка компиляции

C.
Начало программы
Ошибка выполнения

D.
Начало программы
[-1, 0, 1]

Повторение

TEL-RAN
by Starta Institute

Что будет выведено в консоль?

```
import java.util.List;

public class Main {
    public static void main(String[] args) {
        System.out.println("Начало программы");
        List<Integer> nums = List.of(-1, 0, 1);
        nums.set(0, Math.abs(nums.get(0)));
        System.out.println(nums);
    }
}
```

A.
Начало программы
[1, 0, 1]

B.
Ошибка компиляции

C.
Начало программы
Ошибка выполнения

D.
Начало программы
[-1, 0, 1]

При создании коллекции с помощью метода *of* создаётся неизменяемы коллекция. При попытке изменить такую коллекцию выбрасывается *UnsupportedOperationException*

Повторение

TEL-RAN
by Starta Institute

Что будет выведено в консоль?

```
import java.util.List;

public class Main {
    public static void main(String[] args) {
        List<String> strings;
        for (String s: strings) {
            System.out.println(s);
        }
    }
}
```

A. *[]*

B. *Ошибка компиляции*

C. *Ошибка выполнения*

D. *0*

Повторение

TEL-RAN
by Starta Institute

Что будет выведено в консоль?

A. *[]*

B. *Ошибка компиляции*

C. *Ошибка выполнения*

D. *0*

```
import java.util.List;

public class Main {
    public static void main(String[] args) {
        List<String> strings;
        for (String s: strings) {
            System.out.println(s);
        }
    }
}
```

Переменная strings была создана, но никогда не была проинициализирована, т.е. неизвестно, что в ней лежит.

Повторение

TEL-RAN
by Starta Institute

Исправьте ошибку в коде

```
public static void main(String[] args) {
    List<String> strings = new ArrayList<>();
    strings.add(1);
    strings.add(2);
    strings.add(3);
    System.out.println(strings);
}
```

Повторение

TEL-RAN
by Starta Institute

Исправьте ошибку в коде

```
public static void main(String[] args) {
    List<String> strings = new ArrayList<>();
    strings.add("1");
    strings.add("2");
    strings.add("3");
    System.out.println(strings);
}
```

При создании списка было указано, что в нём будут лежать элементы типа String, поэтому в методе add() должны быть указаны соответствующие литералы строк.

Повторение

TEL-RAN
by Starta Institute

Исправьте ошибку в коде

```
public static void main(String[] args) {
    List<int> nums = new ArrayList<>();
    nums.add(1);
    nums.add(2);
    nums.add(3);
    System.out.println(nums);
}
```

Повторение

TEL-RAN
by Starta Institute

Исправьте ошибку в коде

```
public static void main(String[] args) {
    List<Integer> nums = new ArrayList<>();
    nums.add(1);
    nums.add(2);
    nums.add(3);
    System.out.println(nums);
}
```

Коллекции являются параметризируемыми типами (generic). В качестве параметров могут выступать только ссылочные типы, поэтому вместо int нужно указать Integer.

Повторение

TEL-RAN
by Starta Institute

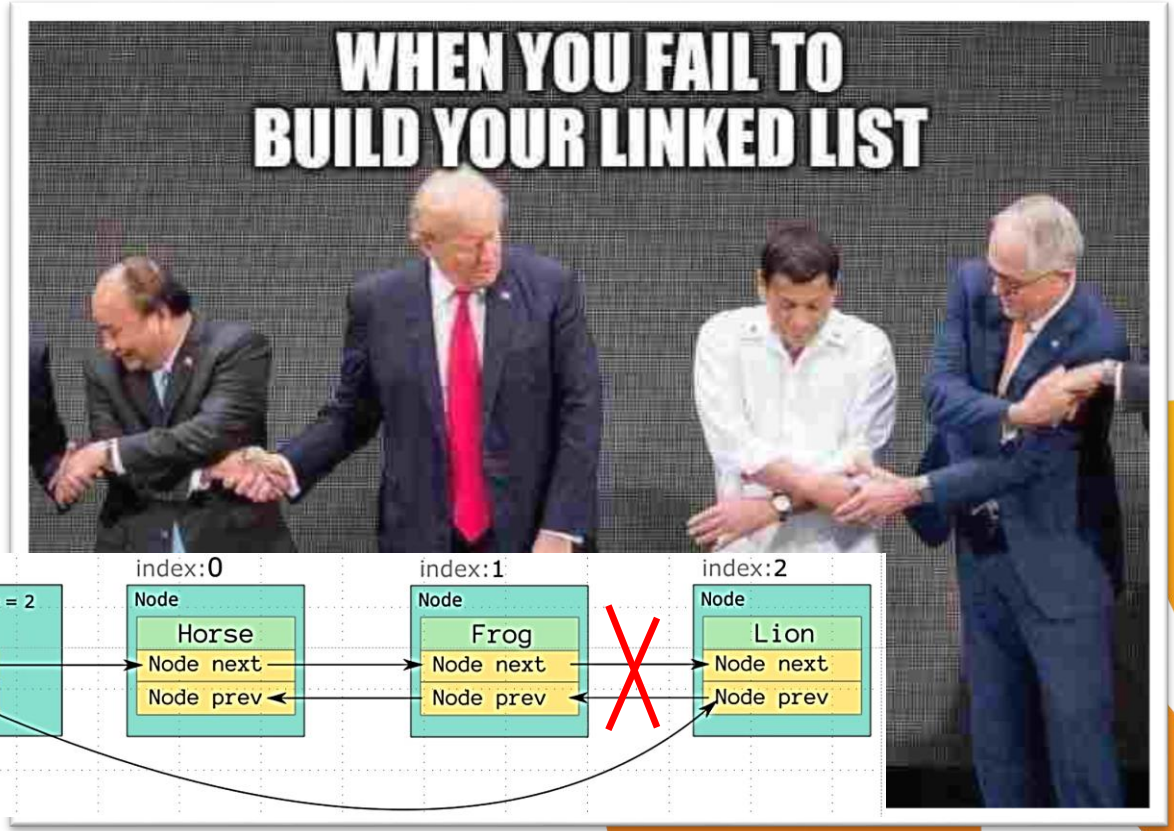
В чём прикол мема?

Повторение

TEL-RAN
by Starta Institute

В чём прикол мема?

Каждый элемент связного списка должен иметь ссылку на следующий элемент, чтобы коллекция оставалась цельной



TEL-RAN
by Starta Institute

2 ОСНОВНОЙ БЛОК

Введение

TEL-RAN
by Starta Institute

- Вас много, а я одна!
- Мы с тобой одной крови – ты и я
- Хеш-магия
- Конкретные множества

Проблема

TEL-RAN
by Starta Institute

Представим, что мы должны написать программу для улучшения работы поликлиник по всему миру.

Первая задача, которую нас попросили решить – это **создать перечень всех пациентов в поликлинике**. Для начала было решено

1 создать сущность *Пациент* с полями *идентификатор, имя, фамилия, дата рождения, адрес, телефон, email*, а также данные о страховании (номер страхового полиса, наименование страховой организации).

2 перенести всех пациентов из текстового файла в программу. Для тестирования нам передали следующий участок файла:

Albert Einstein, Nelson Mandela, Leonardo da Vinci, Mahatma Gandhi, Marie Curie, Martin Luther King Jr., Queen Elizabeth II, Albert Einstein, William Shakespeare, Mahatma Gandhi, Steve Jobs, Steve Jobs, Michael Jackson

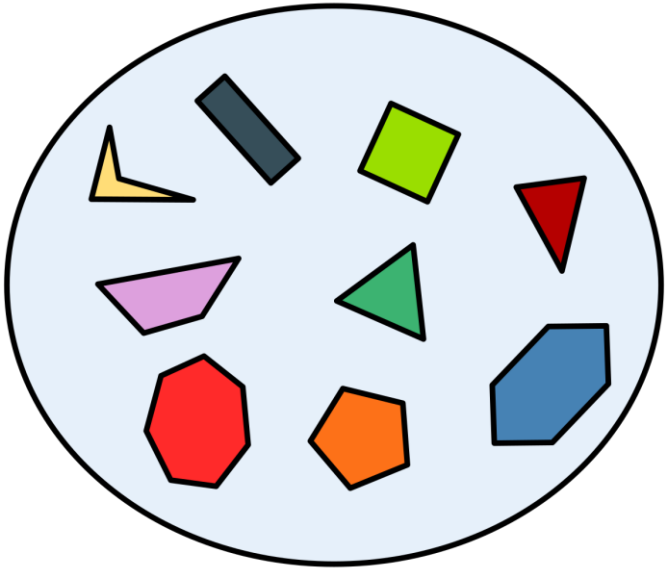
К сожалению, пока есть только имена пациентов, причём некоторые из них повторяются, что недопустимо в будущей программе.

Вас много, а я одна!

Set (множество)

TEL-RAN
by Starta Institute

Set – это неупорядоченный набор уникальных элементов. То же, что *List*, но без повторений и как попало.
Например, мешочек с бочонками для игры в лото: каждый номер от 1 до 90 встречается в нём ровно один раз, и заранее неизвестно, в каком порядке бочонки вынут при игре.



Вас много, а я одна!

Методы Set

TEL-RAN
by Starta Institute

Set содержит те же методы, что и *Collection.*

Set.of() – метод для явного создания неизменяемого множества.

Самый простой способ избавиться от повторов в коллекции - это превратить её в множество. Для этого можно использовать конструктор конкретной реализации интерфейса *Set*.

Вас много, а я одна!

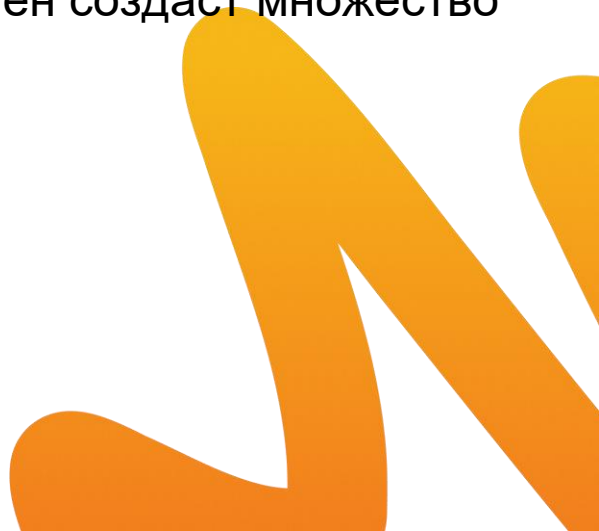
Задание

TEL-RAN
by Starta Institute

1 Напишите метод для распознавания строки с именами в множество полных имён. Используйте *HashSet*.

2 Выведите в консоль полученное множество. Есть ли в нём повторения?

3 Напишите метод, который по множеству полных имён создаст множество пациентов.



Проблема

TEL-RAN
by Starta Institute

Когда *Set* убирает дублирование из исходной коллекции, то как *Set* понимает, что является дубликатом?

Или когда мы вызываем метод *contains()* у *List* или *Set*, как реализации этих интерфейсов «понимают», что они уже содержат такой элемент?

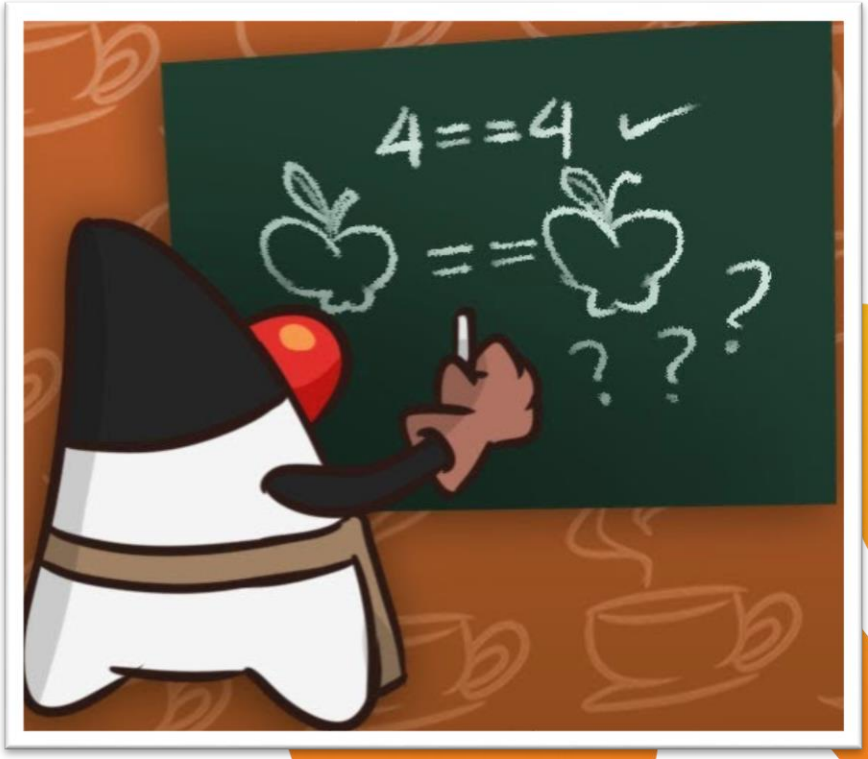
Мы с тобой одной крови – ты и я

Метод equals

TEL-RAN
by Starta Institute

При вызове *contains()* в реализациях *List* происходит обход всех элементов списка. Каждый элемент сравнивается с искомым с помощью метода *equals()*. Если какой-то элемент эквивалентен искомому, то возвращается *true*.

Метод *equals()*, наследованный из класса *Object* сравнивает объекты по ссылке (==).
Если мы хотим иметь возможность сравнивать объекты по значению их полей, то в собственном классе следует переопределить метод *equals()*, чтобы учитывались интересующие нас поля.

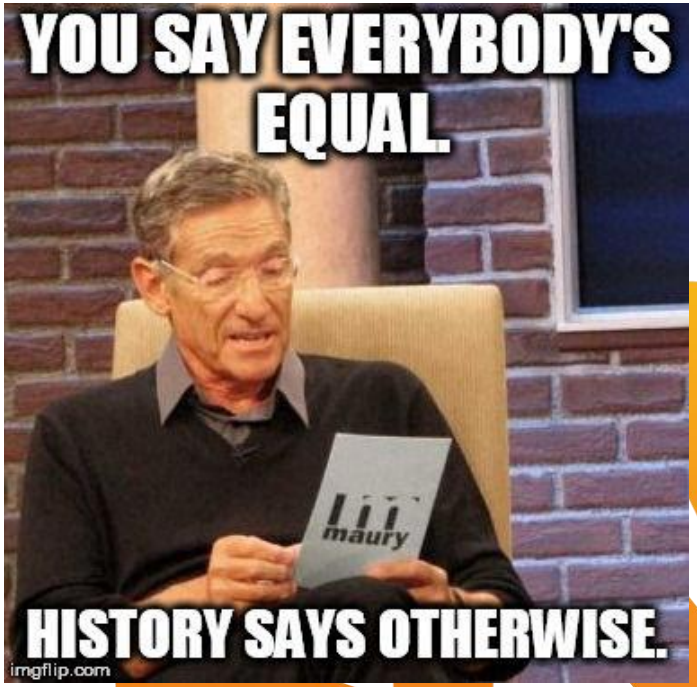


Мы с тобой одной крови – ты и я

Метод equals

TEL-RAN
by Starta Institute

Метод **equals()** подтверждает или отрицает факт, что два объекта одного происхождения являются *логически равными*, то есть, эквивалентны ли их *значимые* поля (не обязательно все поля должны быть идентичны).

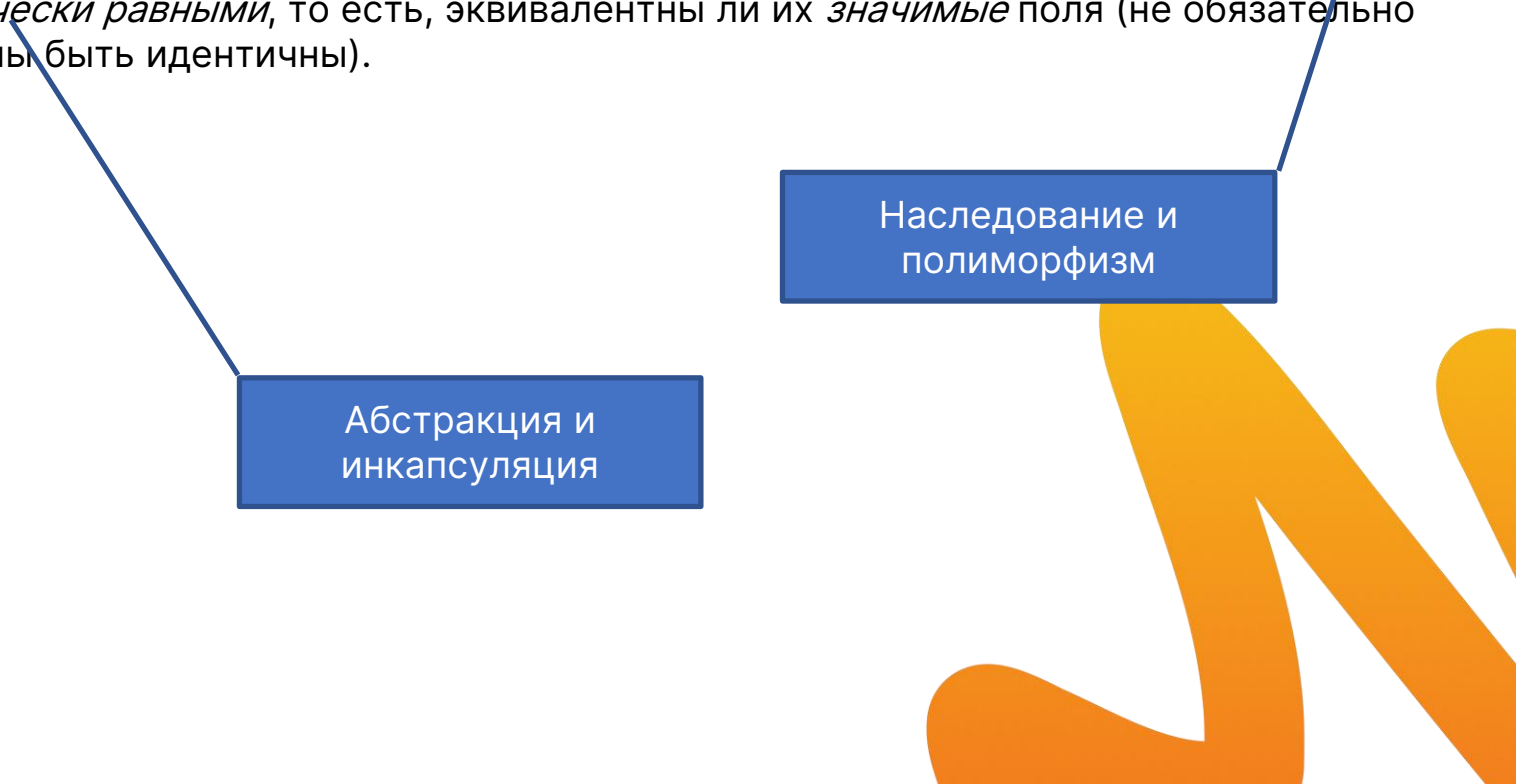


Мы с тобой одной крови – ты и я

Метод equals

TEL-RAN
by Starta Institute

Метод **equals()** подтверждает или отрицает факт, что два объекта одного происхождения являются *логически равными*, то есть, эквивалентны ли их *значимые* поля (не обязательно все поля должны быть идентичны).



Мы с тобой одной крови – ты и я

Контракт equals

TEL-RAN
by Starta Institute

Контракт – это правила, определенные в спецификации языка Java:

Рефлексивность – для любого заданного (*x != null*) значения x, выражение *x.equals(x)* должно возвращать *true*.

Симметричность – для любых заданных значений x и y, *x.equals(y)* должно возвращать *true* только в том случае, когда *y.equals(x)* возвращает *true*.

Транзитивность – для любых заданных значений x, y и z, если *x.equals(y)* возвращает *true* и *y.equals(z)* возвращает *true*, *x.equals(z)* должно вернуть значение *true*.

Согласованность – для любых заданных значений x и y повторный вызов *x.equals(y)* будет возвращать значение предыдущего вызова этого метода при условии, что поля, используемые для сравнения этих двух объектов, не изменялись между вызовами.

Сравнение с null – для любого заданного значения x вызов *x.equals(null)* должен возвращать *false*.

Мы с тобой одной крови – ты и я

Переопределение equals

TEL-RAN
by Starta Institute

1 Проверить на равенство ссылки объектов *this* и параметра метода o.

2 Проверить, определена ли ссылка o, т. е. является ли она *null*.

Если в дальнейшем при сравнении типов объектов будет использоваться оператор *instanceof*, этот пункт можно пропустить, т. к. этот параметр возвращает *false*.

3 Сравнить типы объектов *this* и o с помощью оператора *instanceof* или метода *getClass()*.

```
import person.Person;
import java.util.Arrays;
import java.util.Objects;

public class EqualsMethodExample {
    private String s;
    private int[] array;
    private double d;
    private Person person;

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        EqualsMethodExample other = o instanceof
EqualsMethodExample ? (EqualsMethodExample) o : null;
        if (other == null) return false;
        return other.s.equals(s) &&
                Double.compare(this.d, other.d) == 0 &&
                Arrays.equals(this.array, other.array) &&
                Objects.equals(this.person, other.person);
    }
}
```

Мы с тобой одной крови – ты и я

Переопределение equals

TEL-RAN by Starta Institute

4 Если метод equals переопределяется в подклассе, не забудьте сделать вызов *super.equals(o)*.

5 Выполнить преобразование типа параметра *o* к требуемому классу.

6 Выполнить сравнение всех значимых полей объектов:

- для примитивных типов (кроме *float* и *double*), используя оператор *==*;
- для ссылочных полей необходимо вызвать их метод *equals*;
- для массивов можно воспользоваться перебором по циклу, либо методом *Arrays.equals()*;

```
import person.Person;
import java.util.Arrays;
import java.util.Objects;

public class EqualsMethodExample {
    private String s;
    private int[] array;
    private double d;
    private Person person;

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        EqualsMethodExample other = o instanceof
EqualsMethodExample ? (EqualsMethodExample) o : null;
        if (other == null) return false;
        return other.s.equals(s) &&
                Double.compare(this.d, other.d) == 0 &&
                Arrays.equals(this.array, other.array) &&
                Objects.equals(this.person, other.person);
    }
}
```

Мы с тобой одной крови – ты и я

Переопределение equals

TEL-RAN
by Starta Institute

- для типов *float* и *double* необходимо использовать методы сравнения соответствующих оберточных классов *Float.compare()* и *Double.compare()*;

7 Проверить соответствие контракту переопределения этого метода (см. выше).

```
import person.Person;
import java.util.Arrays;
import java.util.Objects;

public class EqualsMethodExample {
    private String s;
    private int[] array;
    private double d;
    private Person person;

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        EqualsMethodExample other = o instanceof
EqualsMethodExample ? (EqualsMethodExample) o : null;
        if (other == null) return false;
        return other.s.equals(s) &&
                Double.compare(this.d, other.d) == 0 &&
                Arrays.equals(this.array, other.array) &&
                Objects.equals(this.person, other.person);
    }
}
```

Мы с тобой одной крови – ты и я

Когда не стоит переопределять equals

1 Когда каждый экземпляр класса является уникальным. Например, классы перечислений (*Enum*) и класс потоков *Thread* (изучим позже).

2 Когда от класса не требуется определять эквивалентность его экземпляров. Например для класса *java.util.Random* вообще нет необходимости сравнивать между собой экземпляры класса, определяя, могут ли они вернуть одинаковую последовательность случайных чисел. Просто потому, что природа этого класса даже не подразумевает такое поведение.

3 Когда класс, который вы расширяете, уже имеет свою реализацию метода *equals* и поведение этой реализации вас устраивает.

4 Когда область видимости вашего класса является *private* или *package-private* и вы уверены, что этот метод никогда не будет вызван.

Мы с тобой одной крови – ты и я

Задание

TEL-RAN
by Starta Institute

1 Превратите распознанное множество в список.

2 Создайте нового клиента с теми же именем и фамилией, как у одно из клиентов в списке. С помощью метода *contains()* проверьте, содержит ли список такого пациента.

3 В классе *Пациент* переопределите метод *equals()*. Эквивалентными считаются любые пациенты, у которых совпадают имена, фамилии и даты рождения.

4 Повторите программу. Изменился ли результат выполнения метода *contains()*? Почему?

Проблема

TEL-RAN
by Starta Institute

При поиске в коллекциях типа List используется сравнение объектов с помощью *equals()*. Если в объекте будет много полей, а в коллекции будет много объектов, то придётся брать каждый элемент коллекции и сравнивать все поля.

Есть ли способ ускорить этот процесс?

Хеш-магия

Метод hashCode

TEL-RAN by Starta Institute

При поиске в коллекциях типа List используется сравнение объектов с помощью *equals()*. Но в других коллекциях (*HashMap*, *HashSet*) сперва объекты сравнивают по значению хеша. **Хеш** – это число, которое хранит в себе краткое представление значений всех полей объекта. Поэтому при поиске нет нужды сравнивать каждое поле, а достаточно сравнить только хэши объектов. Соответственно, нужно переопределить и метод *hashCode()* (см. исходный код класса *Object*).

Сравнить хеши (два числа) значительно быстрее, чем все поля объекта.

Хеш-магия

Метод hashCode

TEL-RAN
by Starta Institute

При этом хэш-функция получения хэша должна быть реализована таким образом, чтобы обеспечить минимальную вероятность повторения хеша у двух неэквивалентных объектов (коллизий).

Хэш-функция – это реализация метода *hashCode()* в своём классе.

Хеш-магия

Связь equals и hashCode

TEL-RAN
by Starta Institute

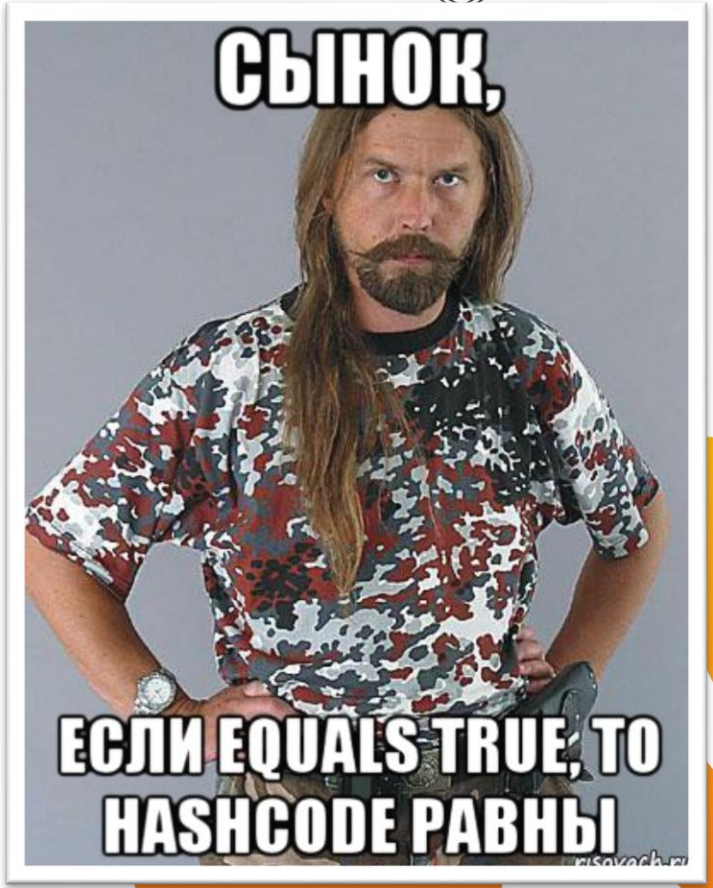
Два логически эквивалентных объекта, должны возвращать одинаковое значение хеш-функции.

Что из этого следует:

1 если хеши двух объектов отличаются, то объекты гарантированно неэквивалентны;

2 неэквивалентные объекты могут иметь одинаковый хеш, если хеш-функция написана не оптимально (*коллизии*).

Если коллекция допускает коллизии, то в ней дополнительно сравнивают объекты по equals.



Хеш-магия

Метод hashCode

TEL-RAN
by Starta Institute

Переопределяя в своем коде метод *equals*, необходимо всегда переопределять и метод *hashCode*

Хеш-магия

Контракт hashCode

TEL-RAN
by Starta Institute

Для реализации хэш-функции в спецификации языка определены следующие правила:

1 вызов метода *hashCode* один и более раз над одним и тем же объектом должен возвращать одно и то же хэш-значение, при условии что поля объекта, участвующие в вычислении значения, не изменялись.

2 вызов метода *hashCode* над двумя объектами должен всегда возвращать одно и то же число, если эти объекты равны (вызов метода *equals* для этих объектов возвращает *true*).

3 вызов метода *hashCode* над двумя неравными между собой объектами должен возвращать разные хеш-значения. Хотя это требование и не является обязательным, следует учитывать, что его выполнение положительно повлияет на производительность работы хеш-таблиц.

Хеш-магия

Переопределение hashCode

TEL-RAN
by Starta Institute

Рекомендуется генерировать *hashCode* средствами IDE.

```
public class EqualsMethodExample {
    private String s;
    private int[] array;
    private double d;
    private Person person;

@Override
    public int hashCode() {
        int result;
        long temp;
        result = s != null ? s.hashCode() : 0;
        result = 31 * result + Arrays.hashCode(array);
        temp = Double.doubleToLongBits(d);
        result = 31 * result + (int) (temp ^ (temp >>> 32));
        result = 31 * result + (person != null ? person.hashCode() : 0);
        return result;
    }
}
```

Для желающих вникнуть в способы переопределения hashCode необходимо изучить статью.

Хеш-магия

Задание

TEL-RAN
by Starta Institute

Переопределите метод hashCode в классе Пациент.

Конкретные множества

HashSet

TEL-RAN
by Starta Institute

Класс **HashSet** использует для хранения данных хеш-таблицу. Это значит, что при манипуляциях с элементами используется хеш-функция - *hashCode()* в Java.

Хеш-таблица – структура данных, в которой все элементы помещаются в корзины (buckets), соответствующие результату вычисления хеш-функции – хешу.

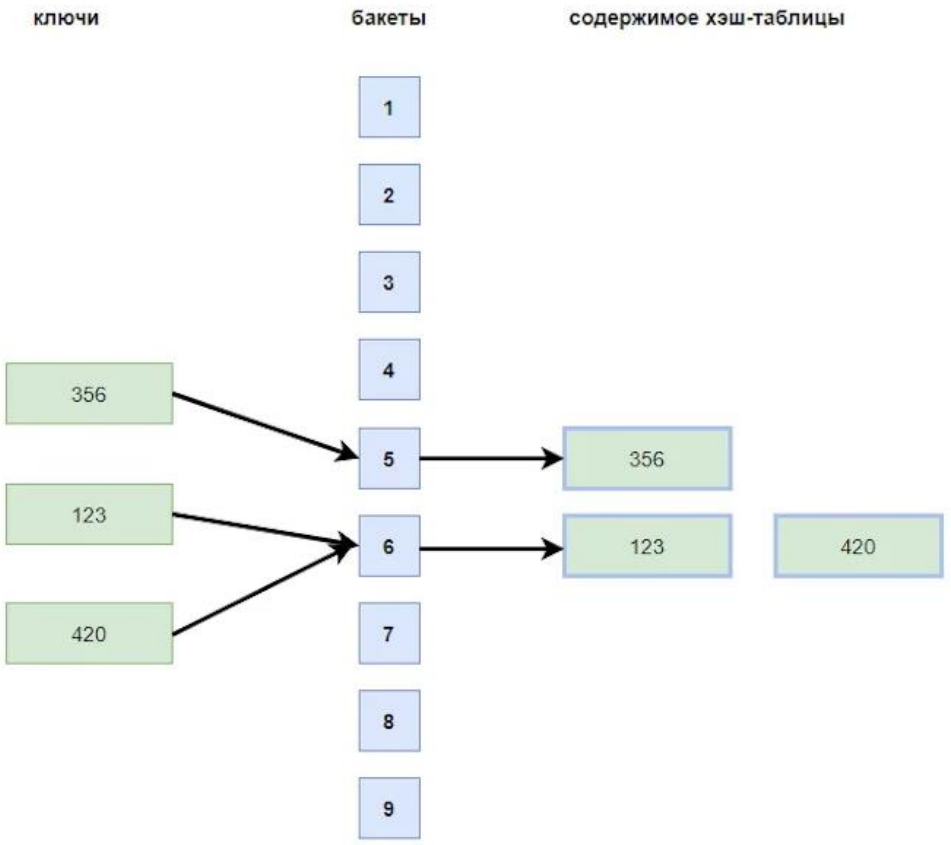
Добавление, поиск и удаление элементов *HashSet* происходит за постоянное время *O(1)*, независимо от числа элементов в коллекции.

```
Set<Long> hSet = new HashSet<>();
```

Конкретные множества
HashSet

Например, администратор в гостинице может класть ключ в коробку с номером от 1 до 9, вычисляя его по такому алгоритму: *складывать все цифры номера, пока не получится одноразрядное число* (искать по одной цифре проще, чем по трём, да и количество коробок для хранения ключей ограничено).

Здесь алгоритм вычисления – это хеш-функция, а результат вычисления — хеш-код. Тогда ключ от номера 356 попадёт в коробку 5 (3 + 5 + 6 = 14; 1 + 4 = 5), а ключ от номера 123 — в коробку с номером 6.



Конкретные множества
HashSet

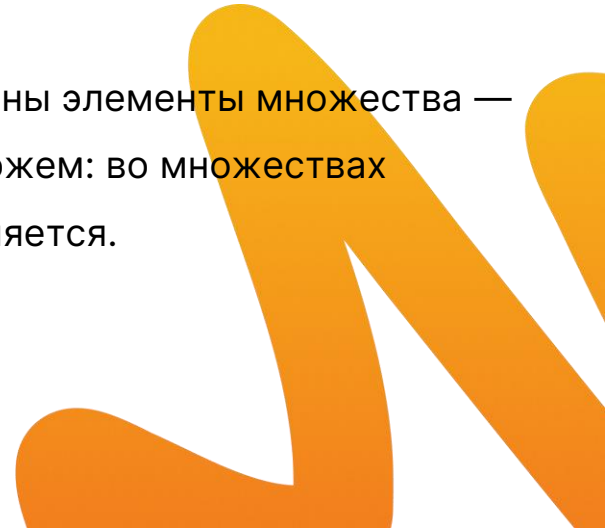
TEL-RAN
by Starta Institute

```
import java.util.HashSet;

public class SimpleTestSet {

    public static void main (String[] args){
        HashSet<String> set = new HashSet<>();
        set.add("Иван");
        set.add("Марья");
        set.add("Пётр");
        set.add("Иван");
        System.out.print(set.size()+":");
        for (String s: set){
            System.out.print(s+" ");
        }
    }
}
```

Вывод: *3:, а дальше точно не известно.*

Так как дубликаты во множествах недопустимы, второй «Иван» не станет частью множества. В итоге размер множества будет равен 3.

В каком порядке будут выведены элементы множества — определённо мы сказать не можем: во множествах порядок добавления не сохраняется.



Конкретные множества

HashSet

TEL-RAN
by Starta Institute

```
import java.util.HashSet;

class Person {
   String name;
   Person(String name) { this.name = name; }
   public String toString() { return name; }
}

class TestHashSet {
   public static void main(String args[]) {
       HashSet<Person> set = new HashSet<>();
       Person p1 = new Person("Иван");
       Person p2 = new Person("Мария");
       Person p3 = new Person("Пётр");
       Person p4 = new Person("Мария");
       set.add(p1);
       set.add(p2);
       set.add(p3);
       set.add(p4);
       System.out.print(set.size());
   }
}
```

Вывод: 4.

Прежде чем добавить новый элемент в множество, вычисляется его *hashCode(),* чтобы определить *bucket*, куда он может быть помещён. Если *bucket* пуст, элемент будет добавлен. Иначе уже добавленные элементы с таким же значением *хеша* сравниваются с кандидатом при помощи метода *equals()*. Если дубликат не найден, новый элемент становится частью множества. Он попадёт в тот же *bucket*.
Мы добавляем в *Set* объекты созданного нами класса *Person*. Так как мы не переопределили метод *hashCode()*, будет использована родительская реализация. В ней хеш вычисляется на основе данных адреса.
Метод *equals()* тоже не переопределён. В классе-родителе этот метод просто сравнивает ссылки на объекты. Это значит, что даже если хеш случайно совпадёт для каких-то из четырёх элементов, *equals()* в любом случае вернёт *false*.

Конкретные множества

Задание

TEL-RAN
by Starta Institute

1 Создайте класс FileCabinet (картотека), который будет хранить множество пациентов.

2 В классе FileCabinet создайте методы для добавления карточки пациента.

3 В случае, если пациент переходит в другую поликлинику, то его данные должны быть переданы в архив текущей поликлиники. Создайте класс Archive, который будет хранить информацию об открепившихся пациентах. Создайте в классе FileCabinet метод передачи данных в архив.

TEL-RAN
by Starta Institute

4 Домашнее задание

Домашнее задание

TEL-RAN
by Starta Institute

№1 Напишите метод, который убирает из списка целых чисел все дубликаты.

№2 Создайте класс правильной дроби (например, 1/2, 7/8 и т.д.). Числитель и знаменатель будут храниться в отдельных полях типа int. Самостоятельно (без генерации в IDE) реализуйте методы equals и hashCode для класса. Причём дроби с одинаковым результатом необходимо считать эквивалентными, т.е. 1/2 эквивалентна 2/4, эквивалентна 5/10 и т.д.

ЗАКЛЮЧЕНИЕ

TEL-RAN
by Starta Institute