





Windows'ams

0. кидаем кряк JetbrainsCrack.jar в папку бин или C:\Program Files\JetBrains\IntelliJ IDEA 2018.3.3\bin

1. открываем папку C:\Users\ваше имя\.IntelliJIdea2018.3\config и открываем папку C:\Program Files\JetBrains\IntelliJ IDEA 2018.3.3\bin (Смотрите актуальную версию вашей директории)

2. открываем файл idea64.exe.vmoptions , который находится в папке C:\Program Files\JetBrains\IntelliJ IDEA 2018.3.3\bin

3. добавляем строку -javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2018.3.3\bin\JetbrainsCrack.jar

4. сохраняем

5. копируем файл idea64.exe.vmoptions и добавляем в папку C:\Users\ваше имя\.IntelliJIdea2018.3\config

6. запускаем прогу, вводим любой код активации и на этом все (edited)

Susikuriu sarasas

Map<Integer, Persone> persons = **new** HashMap<>();

Pasemu faila

Path peopleFile = Paths.*get*(**"C:\\Users\\W10\\IdeaProjects\\Akv\\src\\failai\\people"**);

Nuskaityti duomenis

List<String> lines = Files.*readAllLines*(peopleFile);

Pasitikrinam

**for** (String line: lines){  
 System.***out***.println(line);  
}

sarasa suskaidom I grupes nuo kablelio. Tada yra grazinamas masyvas, bet kartu ir panaikina ta kableli

Persone(Integer.*parseInt*(strings[0]),strings [1].trim(), strings[2].trim());

String[] strings = line.split(**","**);

Java - String split() Method

Nuimam tarpus ir paverciame I Person objekta

Integer.*parseInt t.y. inta paverciam stringu*

Persone person = **new** Persone(Integer.*parseInt*(strings[0]),strings [1].trim(), strings[2].trim());

Sudedam I sarasa

persons.put(person.getID(), person);

4. Pridėkite Person klasei du naujus laukus: receivedMoney ir sentMoney

Sukuriam kintamuosius ir get set jiems tik, nes jau dalis lauku buvo panaudota ir negalime I konstruktoriu ideti

BuferReade naudojam nes failas didelis (mes traktuojam kad didelis)

**try** (BufferedReader reader = Files.*newBufferedReader*(paymentsFile)) {  
 String line;  
 **while** ((line = reader.readLine()) != **null**) {  
 String [] paymentsLine =line.split(**","**);

pasemam tinkamus duomenis

**int** senderId = Integer.*parseInt*(paymentsLine[3].trim());  
**int** amount = Integer.*parseInt*(paymentsLine[2].trim());  
**int** sum = Integer.*parseInt*(paymentsLine[1].trim());

tada naudoti comparator implement ir override compare metoda

rūšiavimas

rusiuoja

Collections.sort(employees);

Čia nustatom ketamąjį. Pagal vieną tik rūšiuojam.

public int compareTo(Employee anotherEmployee) {

return this.getId() - anotherEmployee.getId();

import java.time.LocalDate;

import java.util.Objects;

class Employee implements Comparable<Employee> {

private int id;

private String name;

private double salary;

private LocalDate joiningDate;

public Employee(int id, String name, double salary, LocalDate joiningDate) {

this.id = id;

this.name = name;

this.salary = salary;

this.joiningDate = joiningDate;

}

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public double getSalary() {

return salary;

}

public void setSalary(double salary) {

this.salary = salary;

}

public LocalDate getJoiningDate() {

return joiningDate;

}

public void setJoiningDate(LocalDate joiningDate) {

this.joiningDate = joiningDate;

}

@Override

public int compareTo(Employee anotherEmployee) {

return this.getId() - anotherEmployee.getId();

}

@Override

public boolean equals(Object o) {

if (this == o) return true;

if (o == null || getClass() != o.getClass()) return false;

Employee employee = (Employee) o;

return id == employee.id;

}

@Override

public int hashCode() {

return Objects.hash(id);

}

@Override

public String toString() {

return "Employee{" +

"id=" + id +

", name='" + name + '\'' +

", salary=" + salary +

", joiningDate=" + joiningDate +

'}';

}

}

import java.time.LocalDate;

import java.util.ArrayList;

import java.util.Collections;

import java.util.List;

public class ComparableExample {

public static void main(String[] args) {

List<Employee> employees = new ArrayList<>();

employees.add(new Employee(1010, "Rajeev", 100000.00, LocalDate.of(2010, 7, 10)));

employees.add(new Employee(1004, "Chris", 95000.50, LocalDate.of(2017, 3, 19)));

employees.add(new Employee(1015, "David", 134000.00, LocalDate.of(2017, 9, 28)));

System.out.println("Employees (Before Sorting) : " + employees);

Collections.sort(employees);

System.out.println("\nEmployees (After Sorting) : " + employees);

}

}

**public** V gauti(K raktas) {  
 **for** (Pora <K,V> pora : **sarasasPoru**) {  
 **if** (raktas.equals(pora.getRaktas())) {  
 **return** pora.getReiksme();

V nurodomas grazinimo tipas. Paduodam rakta ir grazinam reiksme.

# Collections

Klase turinti metosdus

Collection objektas

Map ir Set reikia perasyti hachcod ir equals metus kad butu galima patikrinti ar yra toks raktas

#### java.util.TreeMap

geriau tik tada kai reikia strukturos. T.y pagal nustatyta tvarka norim surikiuoti

UUID.*randomUUID*().toString();

Tai sugeneruoja unikalu koda

System.*currentTimeMillis*()

Mili sekundes gauti

Sukurkite statinį metodą, kuris priimtų int tipo parametrą. Parametras nurodo, kiek Payment reikšmių reikia sugeneruoti. Dinamiškai generuokite Payment objektus ir grąžinkite sąrašą su nurodytu reikšmių kiekiu

**public static** List<Payment> getPaymentIdRandum(**int** number) {

transactions.stream().filter(transaction -> transaction.getId()%2==0);

transaction automatiskai Predikcate susikuriama mes jos nematome

.**collect(**Collectors.*toList*()); collect jam reikia perduoti

Iterable klase kuria implementuoja visos kolekcijos r jis turi metoda iterator

Optional<ShopView> shopView

Optional sako gali egzistuoti objektas arba neturi (.findFirst(); sis metodas grzina opsional)

flatMap sujungia di mapus

.orElseGet(()->Item.*clothingItem*(0, **"Cheap"**, 0, 1));

Kad null nemestu

Kitaip get galimti

**public static void** main(String[] args) {  
  
 NameGenerator generator = **new** NameGenerator();  
  
 List<Name> names = generator.generateNames(1000);  
  
 names.forEach(System.***out***::println);  
  
}

atspausdina sarasa

vienas paskui kita

C:\Users\W10\.m2\repository

Maveno saugykla

C:\Users\W10\.m2\repository\lt\codeacademy

messages.forEach(**this**::handleSingleMessage);

**this jai toje pat klaseje**

**jai kitoje tai rasome tos klases pavadinima**

**jai nenaudojame toj kklasej static zodzio tai turime susikurti objekta**

**A a= new A(); ir per ji pasiekti** messages.forEach(**a**::handleSingleMessage);

Arba (-> message)

Norint naudoti tėviniame POM'e dependencyManagement aprašytą priklausomybę, vaikiniame POM'e užtenka:

<dependencies>

<dependency>

<groupId>com.github.javafaker</groupId>

<artifactId>javafaker</artifactId>

</dependency>

</dependencies>

Jei nerasom dependencyManagement

Tai vaikui turiu pateikti versija

