Tolik

Viber: 063 392 26 22

Email: trubintolik@gmail.com

SRC:  [**https://drive.google.com/open?id=0B4hO51AiXiZ9bWhVMHBhOEdTWjQ**](https://drive.google.com/open?id=0B4hO51AiXiZ9bWhVMHBhOEdTWjQ)

**UML: https://drive.google.com/open?id=0B4hO51AiXiZ9V0FmVG1pcVM1U2M**

**[07/Mar/2004:16:05:49 -0800] "GET Main.ConfigurationVariables HTTP/1.1"**

**LogToken**

**time = 07/Mar/2004:16:05:49**

**method = GET**

**message = Main.ConfigurationVariables HTTP/1.1**

**[07/Mar/2004:16:06:51 -0800] "GET /twiki/bin/rdiff/TWiki/NewUserTemplate?"**

**[07/Mar/2004:16:10:02 -0800] "POST /mailman/listinfo/hsdivision HTTP/1.1"**

**[07/Mar/2004:16:05:49 -0800] "GET Main.ConfigurationVariables HTTP/1.1"**

**output to XML:**

**<logtokens>**

**<size>3</size>**

**<logtoken>**

**<time>07/Mar/2004:16:06:51</time>**

**<method>GET</method>**

**<message>/twiki/bin/rdiff/TWiki/NewUserTemplate?</message>**

**</logtoken>**

**<logtoken>**

**<time>07/Mar/2004:16:10:02</time>**

**<method>POST</method>**

**<message>/mailman/listinfo/hsdivision HTTP/1.1</message>**

**</logtoken>**

**<logtoken>**

**<time>07/Mar/2004:16:05:49</time>**

**<method>GET</method>**

**<message>Main.ConfigurationVariables HTTP/1.1</message>**

**</logtoken>**

**</logtokens>**

**output to JSON:**

**[**

**{**

**"time" : "07/Mar/2004:16:06:51",**

**"method" : "GET",**

**"message" : "/twiki/bin/rdiff/TWiki/NewUserTemplate?"**

**},**

**{**

**"time" : "07/Mar/2004:16:10:02",**

**"method" : "POST",**

**"message" : "/mailman/listinfo/hsdivision HTTP/1.1"**

**},**

**{**

**"time" : "07/Mar/2004:16:05:49",**

**"method" : "GET",**

**"message" : "Main.ConfigurationVariables HTTP/1.1"**

**}**

**]**

**Задача со звездочкой:**

**Написать класс LogAnalyzer с методом который принимает три параметра String path, java.time.LocalDateTime timeFrom, LocalDateTime timeTo.**

**Метод должен просканировать файл, и вернуть коллекцию из LogToken, которые удовлетворяют условию временного поиска > timeFrom и < timeTo.**

**LogToken содержит поля -> LocalDateTime time, HttpMethod method, String message.**

**HttpMethod - наш enum с константами GET и POST.**

**http://www.monitorware.com/en/logsamples/apache.php**

**первый лог**

**это к третьей задаче**

**static int calculateFiles(String path) {**

**int count = 0;**

**File file = new File(path);**

**File[] fileAndDirs = file.listFiles();**

**for (File fileAndDir : fileAndDirs) {**

**if (fileAndDir.isFile()) {**

**count++;**

**} else {**

**count += *calculateFiles*(fileAndDir.getAbsolutePath());**

**}**

**}**

**return count;**

**}**

**import java.io.IOException;**

**import java.sql.SQLException;**

**public class EmailFileDao implements EmailDao {**

**public String[] getEmails() {**

**try {**

**throw new SQLException();**

**} catch (SQLException e) {**

**throw new RuntimeException(e);**

**}**

**}**

**}**

**import java.io.IOException;**

**import java.sql.SQLException;**

**public class EmailService {**

**EmailDao emailDao;**

**void sendEmails() {**

**try {**

**String[] emails = emailDao.getEmails();**

**for (String email : emails) {**

***//send email***

**}**

**} catch (IOException e) {**

**e.printStackTrace();**

**} catch (SQLException e) {**

**e.printStackTrace();**

**}**

**}**

**}**

**import java.io.IOException;**

**import java.sql.SQLException;**

**public class EmailFileDao implements EmailDao {**

**public String[] getEmails() throws IOException, SQLException {**

***// work with IO***

**return null;**

**}**

**}**

**import java.io.IOException;**

**import java.sql.SQLException;**

**public class EmailDBDao implements EmailDao {**

**public String[] getEmails() throws SQLException, IOException {**

***// work with DB***

**return null;**

**}**

**}**

**import java.io.IOException;**

**import java.sql.SQLException;**

**public interface EmailDao {**

**String[] getEmails() throws SQLException, IOException;**

**}**

**Задача со звездочкой:**

**Написать класс LogAnalyzer с методом который принимает три параметра String path, java.time.LocalDateTime timeFrom, LocalDateTime timeTo.**

**Метод должен просканировать файл, и вернуть коллекцию из LogToken, которые удовлетворяют условию временного поиска > timeFrom и < timeTo.**

**LogToken содержит поля -> LocalDateTime time, HttpMethod method, String message.**

**HttpMethod - наш enum с константами GET и POST.**

**http://www.monitorware.com/en/logsamples/apache.php**

**первый лог**

**это к третьей задаче**

***Into FileManager***

***- public static void copy(String from, String to) - метод по копированию папок и файлов. Параметр from - путь к файлу или папке, параметр to - путь к папке куда будет производиться копирование.***

***- public static void move(String from, String to) - метод по перемещению папок и файлов. Параметр from - путь к файлу или папке, параметр to - путь к папке куда будет производиться перемещение.***

**package ua.trusin.datastructures.queue;**

**import java.io.\*;**

**public class IOTest {**

**public static void main(String[] args) throws IOException {**

**Reader reader = new FileReader("test.txt");**

**BufferedReader bufferedReader = new BufferedReader(reader);**

**String value;**

**while ((value = bufferedReader.readLine()) != null) {**

**System.*out*.println(value);**

**}**

**reader.close();**

**}**

**}**

**package ua.trusin.datastructures.queue;**

**import java.io.\*;**

**public class IOTest {**

**public static void main(String[] args) throws IOException {**

**InputStream inputStream = null;**

**OutputStream outputStream = null;**

**try {**

**inputStream = new FileInputStream("test.txt");**

**outputStream**

**= new FileOutputStream("copyOfTest.txt");**

**int value;**

**while ((value = inputStream.read()) != -1) {**

**outputStream.write(value);**

**}**

**} finally {**

**inputStream.close();**

**outputStream.close();**

**}**

**}**

**}**

**написать класс FileManager, со следующими методами:**

**- public static int calculateFiles(String path) - принимает путь к папке, возвращает количество файлов в папке и всех подпапках по пути**

**- public static int calculateDirs(String path) - принимает путь к папке, возвращает количество папок в папке и всех подпапках по пути**

**package ua.trusin.datastructures.queue;**

**import java.io.File;**

**import java.io.IOException;**

**public class FileTest {**

**public static void main(String[] args) {**

***//C:\Users\grey25\IdeaProjects\Test\test.txt***

***//       File file = new File("test.txt"); // C:\\boot.ini***

***//        System.out.println(file.exists());***

***//        try {***

***//            file.createNewFile();***

***//        } catch (IOException e) {***

***//            System.out.println(e.getMessage());***

***//        }***

***//***

***//        System.out.println(file.exists());***

***//        File dir = new File("C://test");***

***//        System.out.println(dir.exists());***

***//        dir.mkdir();***

***//        System.out.println(dir.exists());***

**File file = new File("C://Windows");**

***//        for (String name : file.list()) {***

***//            String path = file.getAbsolutePath() + "/" + name;***

***//            System.out.println(path);***

***//        }***

**File[] files = file.listFiles();**

**for (File fileItem : files) {**

**String output = fileItem.isFile() ? "file" : "dir";**

**output += " -> " + fileItem.getPath();**

**System.*out*.println(output);**

**}**

**}**

**}**

**import java.io.FileNotFoundException;**

**import java.util.Random;**

**public class ExceptionTest {**

**public static void main(String[] args) {**

**System.*out*.println("start");**

***m1*();**

**System.*out*.println("finish");**

**}**

**public static void m1() {**

**System.*out*.println("m1");**

***m2*();**

**}**

**public static void m2() throws Exception {**

***m3*();**

**System.*out*.println("m2");**

**}**

**public static void m3() throws MyCustomException, FileNotFoundException {**

**Random random = new Random();**

**if (random.nextBoolean()) {**

**throw new MyCustomException();**

**}**

**throw new FileNotFoundException();**

**}**

**}**

**https://www.coursera.org/learn/algorithms-part1**

**package ua.trusin.datastructures.queue;**

**public class ArrayQueue implements Queue{**

**static final int *INITIAL\_CAPACITY* = 10;**

**int size;**

**Object[] array = new Object[*INITIAL\_CAPACITY*];**

**int start;**

**int end;**

**@Override**

**public void push(Object value) {**

**}**

**@Override**

**public Object pop() {**

**return null;**

**}**

**@Override**

**public Object peek() {**

**return null;**

**}**

**}**

**package ua.trusin.datastructures.queue;**

**public class LinkedQueue implements Queue {**

**int size;**

**Node head;**

**@Override**

**public void push(Object value) {**

**}**

**@Override**

**public Object pop() {**

**if (size == 0) {**

**throw new NullPointerException("Queue is empty");**

**}**

**Object value = head.value;**

**head = head.next;**

**size--;**

**return value;**

**}**

**@Override**

**public Object peek() {**

**return null;**

**}**

**}**

**package ua.trusin.datastructures.queue;**

**public class LinkedQueueTest {**

**public static void main(String[] args) {**

**LinkedQueue linkedQueue = new LinkedQueue();**

**linkedQueue.push("A");**

**linkedQueue.push("B");**

**linkedQueue.push("C");**

**}**

**}**

**package** ua.trusin.datastructures.queue;

**public class** Node {

  Object **value**;

  Node **next**;

}

**package ua.trusin.datastructures.queue;**

**public interface Queue {**

**void push(Object value);**

**Object pop();**

**Object peek();**

**}**

**public class** ArrayList {

  Object[] **array** = **new** Object[5];

**int size**;

**void** add(Object value) {

**array**[**size**] = value;

**size**++;

  }

*// 0 .. < size*

Object get(**int** index) {

**if** (index < 0 || index >= **size**) {

          String msg = **"Index -> "** + index + **"  should be "** +

**"between 0 and "** + **size** + **" exclusive"**;

**throw new** IndexOutOfBoundsException(msg);

      }

**return array**[index];

  }

}

**public class** ArrayListTest {

**public static void** main(String[] args) {

      ArrayList arrayList = **new** ArrayList();

      arrayList.add(**"hello"**);

      arrayList.add(**"world"**);

      System.***out***.println(arrayList.**size**);

      System.***out***.println(arrayList.get(0));

      System.***out***.println(arrayList.get(1));

      arrayList.get(10);

  }

}

**<http://www.bookzone.com.ua/Netshop/catalogue/catalogue_32260.html?gclid=CjwKEAiAs_PCBRD5nIun9cyu01kSJAA-WD-rdl5dyqwEjLYwHD84P1NFe8Ptxddp8r9FqtiOSaxQnhoCacXw_wcB>**

**Глава 1, Паттерн стратегия**

**Дан холодильник закрытый на замок. На передней панели расположено 16 рычагов матрицей 4 на 4. Рычаги могут принимать как горизонтальное, так и вертикальное положение.**

**При повороте одного из рычагов, все рычаги по вертикали и горинтали от поворачиваемого меняют положение на противоположное.**

**Когда все рычаги принимают горизонтальное положение холодильник открывается.**

**1) Разработать формат хранения матрицы. Тоесть предусмотреть возможность решения заранее заданой матрицы.**

**2) Найти минимальное количество шагов за которое можно решить матрицу и открыть холодильник.**

**3) Вывести комбинацию рычагов которую нужно повернуть для того чтобы открыть холодильник.**

**п.с. рекурсия, ООП стиль.**

**http://questgame.ru/images/pilots/2.jpg**

**public class Test {**

**public static void main(String[] args) {**

**EmployeeService employeeService = new EmployeeService();**

**employeeService.generateEmployees(10);**

**employeeService.printEmployees();**

**System.*out*.println("======================================");**

**for (Employee employee : employeeService.getByName("Alex")) {**

**System.*out*.println(employee);**

**}**

**}**

**}**

**import java.util.Random;**

**public class Developer extends Employee {**

**int fixedBugs;**

**double calculateSalary() {**

**return (salary + fixedBugs \* 1.5) \* (new Random().nextBoolean() ? 2 : 0);**

**}**

**public String toString() {**

**return "Developer{" +**

**super.toString() +**

**", fixedBugs=" + fixedBugs +**

**'}';**

**}**

**}**

**import java.util.Random;**

**public class EmployeeService {**

**Employee[] employees;**

**void generateEmployees(int size) {**

**String[] names = {"Tolik", "Vasya", "Vika", "Alex"};**

**Employee[] employees = new Employee[size];**

**Random random = new Random();**

**for (int i = 0; i < employees.length; i++) {**

**boolean isDeveloper = random.nextBoolean();**

**if (isDeveloper) {**

**Developer developer = new Developer();**

**developer.fixedBugs = random.nextInt(100);**

**employees[i] = developer;**

**} else {**

**Cleaner cleaner = new Cleaner();**

**cleaner.rate = 1.5;**

**cleaner.workedDays = random.nextInt(30);**

**employees[i] = cleaner;**

**}**

**Employee employee = employees[i];**

**employee.id = i + 1;**

**employee.age = random.nextInt(25) + 25;**

**employee.salary = random.nextInt(250) + 2500;**

**employee.gender = random.nextBoolean() ? 'M' : 'F';**

**employee.name = names[random.nextInt(names.length)];**

**}**

**this.employees = employees;**

**}**

**void printEmployees() {**

**for (Employee employee : employees) {**

**System.*out*.println(employee);**

**}**

**}**

**Employee[] getByName(String name) {**

**int count = 0;**

**int index = 0;**

**for (Employee employee : employees) {**

**String employeeName = employee.name;**

**if (employeeName.equals(name)) {**

**count++;**

**}**

**}**

**Employee[] result = new Employee[count];**

**for (Employee employee : employees) {**

**String employeeName = employee.name;**

**if (employeeName.equals(name)) {**

**result[index++] = employee;**

**}**

**}**

**return result;**

**}**

**double calculateSalary() {**

**double sum = 0;**

**for (Employee employee : employees) {**

**sum += employee.calculateSalary();**

**}**

**return sum;**

**}**

**}**

**public abstract class Employee {**

**long id;**

**String name;**

**int age;**

**double salary;**

**char gender;**

**abstract double calculateSalary();**

**public String toString() {**

**return "id=" + id +**

**", name=" + name +**

**", gender=" + gender +**

**", age=" + age +**

**", salary=" + salary;**

**}**

**}**

**Разработать приложение по учету сотрудников предприятия.**

**Типы сотрудников + характеристики для хранения:**

**Developer (id, name, age, salary, gender, fixedBugs)**

**Manager (id, name, age, salary, gender)**

**Cleaner (id, name, age, salary, gender, rate, workedDays)**

**ставка -> salary**

**Расчет итоговой зарплаты для Developer происходит по формуле:**

**(ставка + fixedBugs \* коэффициент) \* (randomBoolean ? 2 : 0)**

**Итоговой зарплатой Manager-а является его ставка:**

**Расчет итоговой зарплаты для Cleaner происходит по формуле:**

**ставка + rate \* workedDays**

**Разработать класс EmployeeService который будет хранить всех сотрудников предприятия в массиве.**

**Класс должен содержать методы для работы с сотрудниками (которые представлены массивом внутри класса):**

**void generateEmployees(size) -> генерирует случайным образом заданное количество сотрудников, и присваивает их внутреннему массиву (смотри AnimalService пример)**

**void printEmployees() -> вывод на экран информации о сотрудниках**

**double calculateSalaryAndBonus() -> возвращает количество денег необходимое для выплаты зарплат для всех сотрудников в этом месяце**

**Employee getById(long) -> возвращает сотрудника по заданному id**

**Employee[] getByName(String) -> возвращает сотрудников по заданному имени**

**Employee[] sortByName()**

**Employee[] sortByNameAndSalary() -> возвращают отсортированный массив с сотрудниками по критерию**

**Employee edit(Employee) -> находит сотрудника по id, и подменяет информацию о нем на новую. Старую версию сотрудника метод возвращает.**

**Использовать класс Test для создания объекта EmploeeService и тестирования описаных выше методов.**

**public class EmployeeService {**

**Employee[] employees;**

**void generateEmployees(int size) {**

**Employee[] employees = new Employee[size];**

***// code***

**this.employees = employees;**

**}**

**void printEmployees() {**

**for (Employee employee : employees) {**

***// print***

**}**

**}**

**}**

***/\*\****

***\* Created by user on 21.12.2016.***

***\*/***

**public class Test {**

**public static void main(String[] args) {**

**EmployeeService employeeService = new EmployeeService();**

**employeeService.generateEmployees(10);**

**employeeService.printEmployees();**

**Employee e1 = new Developer();**

**Employee e2 = new Manager();**

**}**

**static void printType(Employee employee) {**

**if (employee instanceof Manager) {**

**System.*out*.println("Manager");**

**} else if (employee instanceof Developer) {**

**Developer dev = (Developer) employee;**

**System.*out*.println("Developer");**

**}**

**}**

**}**

**public class Dog extends Animal {**

**double killedEnemies;**

**public Dog() {**

**}**

**public Dog(String name, int age, int weight, double killedEnemies) {**

**this.name = name;**

**this.age = age;**

**this.weight = weight;**

**this.killedEnemies = killedEnemies;**

**}**

**void eat() {**

**System.*out*.println("I am dog and I am EATING!!!!");**

**killedEnemies++;**

**}**

**void describe() {**

**System.*out*.print("hi I am dog [");**

**super.describe();**

**System.*out*.print(", killedEnemies = " + killedEnemies);**

**System.*out*.println("]");**

**}**

**}**

**import java.util.Random;**

**public class Cat extends Animal {**

**boolean inMyayProcess;**

**Cat() {**

**age = 1;**

**weight = 0.1;**

**name = "Unknown";**

**}**

**Cat(int age, String name, double weight) {**

**this.age = age;**

**this.name = name;**

**this.weight = weight;**

**}**

**void eat() {**

**if (inMyayProcess) {**

**System.*out*.println("myay");**

**}**

**weight += 0.5;**

**}**

**void play() {**

**super.play();**

**System.*out*.println("I am not eating");**

**weight += 1;**

**}**

**void describe() {**

**System.*out*.print("hi I am cat [");**

**super.describe();**

**System.*out*.print(", inMyayProcess = " + inMyayProcess);**

**System.*out*.println("]");**

**}**

**Cat makeCat(Cat cat) {**

**Cat newCat = new Cat();**

**String fatherName = cat.name;**

**String motherName = name;**

**String firstPart = fatherName.substring(0, fatherName.length() / 2);**

**String secondPart = motherName.substring(motherName.length() / 2);**

**newCat.name = firstPart + secondPart;**

**return newCat;**

**}**

**}**

**import java.util.Random;**

**public class Animal {**

**String name;**

**int age;**

**double weight;**

**void play() {**

**Random random = new Random();**

**System.*out*.println(name + " is playing");**

**if (random.nextBoolean()) {**

**weight -= 0.25;**

**} else {**

**age += 1;**

**}**

**}**

**void eat() {**

**System.*out*.println("eat process");**

**}**

**void describe(){**

**System.*out*.print("name = " + name);**

**System.*out*.print(", age = " + age);**

**System.*out*.print(", weight = " + weight);**

**}**

**}**

**public class Test {**

**public static void main(String[] args) {**

**Cat cat = new Cat(5, "Kuzya", 10);**

**Animal animal = cat;**

**animal.age = 12;**

**Animal dog = new Dog();**

**Animal[] animals = new Animal[2];**

**animals[0] = cat;**

**animals[1] = dog;**

***play*(animals);**

***feed*(animals);**

**for (Animal animal1 : animals) {**

**animal1.describe();**

**}**

**}**

**static void play(Animal[] animals) {**

**for (Animal animal : animals) {**

**animal.play();**

**}**

**}**

**static void feed(Animal[] animals) {**

**for (Animal animal : animals) {**

**animal.eat();**

**}**

**}**

**}**

**public class Test {**

**public static void main(String[] args) {**

**Cat cat = new Cat(5, "Kuzya", 10);**

**cat.describe();**

**cat.play();**

**Cat secondCat = new Cat();**

**secondCat.name = "Murka";**

**secondCat.age = 3;**

**secondCat.weight = 3.5;**

**Cat newCat = secondCat.makeCat(cat);**

**newCat.describe();**

***//        print(secondCat);***

**}**

**static void print(Cat cat) {**

**System.*out*.print("cat[");**

**System.*out*.print("name = " + cat.name);**

**System.*out*.print(", age = " + cat.age);**

**System.*out*.print(", weight = " + cat.weight);**

**System.*out*.println("]");**

**}**

**}**

**import** java.util.Random;

**public class** Cat {

  String **name**;

**int age**;

**double weight**;

  Cat() {

**age** = 1;

**weight** = 0.1;

**name** = **"Unknown"**;

  }

  Cat(**int** catAge, String catName, **double** catWeight) {

**age** = catAge;

**name** = catName;

**weight** = catWeight;

  }

**void** eat() {

**weight** += 0.5;

  }

**void** play() {

      Random random = **new** Random();

      System.***out***.println(**"I am playing"**);

**if** (random.nextBoolean()) {

**weight** -= 0.25;

      } **else** {

          eat();

**age** += 1;

      }

  }

**void** describe() {

      System.***out***.print(**"hi I am cat ["**);

      System.***out***.print(**"name = "** + **name**);

      System.***out***.print(**", age = "** + **age**);

      System.***out***.print(**", weight = "** + **weight**);

      System.***out***.println(**"]"**);

  }

  Cat makeCat(Cat cat) {

      Cat newCat = **new** Cat();

      String fatherName = cat.**name**;

      String motherName = **name**;

      String firstPart = fatherName.substring(0, fatherName.length() / 2);

      String secondPart = motherName.substring(motherName.length() / 2);

      newCat.**name** = firstPart + secondPart;

**return** newCat;

  }

}

import java.util.\*;

class GameOne {

public static void main(String [] args){

int [][] array = {{1,2,3},

 {10,20},

 {-11}};

invert(array);

print(array);

}

// принимает 2м массив интов,  ивертирует

// массив интов (каждое значение в передаваемом

// массиве меняет знак)

static void invert(int[][] array) {

for(int i = array.length - 1; i >=0; i--){

for(int j = 0; j < array[i].length; j++){

int value = array[i][j];

array[i][j] = value \* -1;

}

}

}

static void print(int[][] array){

for(int i = 0; i < array.length; i++){

int[] innerArray = array[i];

for(int j = 0; j < innerArray.length; j++){

System.out.print(innerArray[j] + " ");

}

System.out.println();

}

}

}

|  |
| --- |
| 1. **принимает 2м массив строк, выводит его на экран** |
| **2) принимает 2м массив интов, возвращает 2мерный массив чаров, каждый символ в позиции массива соответствует коду символа передаваемого инта** |
| **3) принимает 2м массив интов, ивертирует массив интов (каждое значение в передаваемом массиве меняет знак)** |
| **4) приминает 2 инта, а и б, возвращает большее их этих 2х чисел** |
| **5) принимает 3 инта, возвращает большее из них** |
| **6) приминает 5 интов, возвращает большее из них** |
| **7) принимает массив чаров, возвращает строку состоящую из символов массива** |
| **8) принимает 2 массива чаров, проверяет есть ли в 1 массиве, такая же последовательность символов которую представляет собой второй массив. Возвращает булеан** |
| **9) принимает массив интов, и значение типа инт, возвращает индекс массива в котором значение совпадает с передаваемым, начиная с начала массива. Если значения в массиве нет возвращает -1** |
| **10) принимает массив интов, и значение типа инт, возвращает индекс массива в котором значение совпадает с передаваемым, начиная с конца массива. Если значения в массиве нет возвращает -1** |
| **11) метод принимает инт, и возвращает факториал от заданого инта** |
| **12) принимает инт год, и возвращает тру если год высокосный** |
| **13) Принимает массив строк и просто строку, возвращает массив строк которые содержат данную подстроку. т.е. отфильтрованый из первого** |
| **14) приминает массив интов и число, выводит на екран только елементы массива которые кратны этому числу** |
| **15) метод приминает дабл, выводит на екран этот дабл с точностью не больше чем 3 знака после точки** |
| **16) метод принимает массив интов сортирует его по возрастанию** |
| **17) метод приминимает массив интов сортирует его по убыванию (нельзя использовать методы сторонних классов)** |
| **18) метод принимает массив интов и булен, если булен тру то сортирует по убываниию, иначе по возрастанию** |
| **19) принимает массив байт, если в массиве есть повторяющиеся елементы, возвращает тру**  **20) Написать 2 метод для конвертации :**  **int -> String**  **double -> String**  **1) Нельзя использовать стандартные методы для конвертации.**  **2) При контеканации можно за раз использовать не больше 1 символа (т.е. при переводе из числа в строку, из числа нужно достать по одной цифре)**  **3) Числа могут быть отрицательными**  **4) Числа с плавающей точкой могут и не содержать дробную часть (12345)**  **п.с. Можно использовать методы класса Math** |

public static void main(String [] args){

int[][] array  = {{1,2,3},

 {4,5,6},

 {7,8,9}};

print(array);

}

static void print(int[][] array){

for(int i = 0; i < array.length; i++){

int [] innerArray = array[i] ;

for(int j = 0; j < innerArray.length; j++){

System.out.print(innerArray[j] + ' ');

}

System.out.println();

}

}

int[][] array  = new int[2][3];

int [] firstInnerArray = array[0];

firstInnerArray [1] = 10;

array[1][2] = 250;

array[1] = new int[2];

array[0] = new int[]{1,2,3};

System.out.println(array.length);

**public static void main(String [] args){**

**int [] arrayOne = new int [5];**

**int [] array = {1,2,3,4,5};**

**print(array);**

**changeArray(array, 1, 300);**

**print(array);**

**}**

**static void changeArray(int[] array, int index, int value){**

**array[index] = value;**

**array = new int[]{5,4,3,2,1};**

**}**

**public static void main(String [] args){**

**int a = 5;**

**int b = 7;**

**swap(a, b);**

**System.out.println("a = " + a);**

**System.out.println("b = " + b);**

**}**

**static void swap (int a, int b){**

**// a = 5, b = 7**

**int temp = a;**

**a = b;**

**b = temp;**

**}**

**import java.util.\*;**

**class GameOne {**

**public static void main(String [] args){**

**int [] array = generateRandomArray(5);**

**print(array);**

**String str = "hello";**

**char c = str.charAt(4);**

**System.out.println(str.indexOf("ll"));**

**System.out.println(Math.abs(-10000));**

**}**

**static void print(int [] array) {**

**for(int i = 0; i < array.length; i++){**

**System.out.println(array[i]);**

**}**

**}**

**static int[] generateRandomArray(int size){**

**Random random = new Random();**

**int [] array = new int[size];**

**for(int i = 0; i < array.length; i++){**

**array[i] = 60 + random.nextInt(100);**

**}**

**return array;**

**}**

**}**

import java.util.\*;

class GameOne {

public static void main(String [] args){

int [] array = generateRandomArray(5);

print(array);

}

static void print(int [] array) {

for(int i = 0; i < array.length; i++){

System.out.println(array[i]);

}

}

static int[] generateRandomArray(int size){

Random random = new Random();

int [] array = new int[size];

for(int i = 0; i < array.length; i++){

array[i] = 60 + random.nextInt(100);

}

return array;

}

}

import java.util.\*;

class GameOne {

public static void main(String [] args){

//code

printHello();

String message = "java rulezzz!!";

printMessage(message);

printMessage(message, 2);

int aPlusB = sum(10, 5);

}

static int sum(int a, int b){

// a = 10, b = 5

return a + b;

}

static void printMessage(String msg, int count){

for(int i = 0; i < count; i++){

printMessage(msg);

}

}

static void printMessage(String message){

printHello();

System.out.print("<<<<");

System.out.print(message);

System.out.println(">>>>");

}

static void printHello(){

System.out.print("hello");

System.out.println("world");

}

}

import java.util.\*;

class GameOne {

public static void main(String [] args){

/\*

Random randomGenerator;

randomGenerator = new Random();

int randomValue = randomGenerator.nextInt(10);

System.out.println(randomValue);

\*/

Scanner scanner = new Scanner(System.in);

for(int i = 0; i < 5; i++){

String str = scanner.nextLine();

if("exit".equals(str)){

break;

}

int number = Integer.parseInt(str);

System.out.println("you just entered " + number);

}

}

}

import java.util.\*;

class GameOne {

public static void main(String [] args){

/\*

Random randomGenerator;

randomGenerator = new Random();

int randomValue = randomGenerator.nextInt(10);

System.out.println(randomValue);

\*/

Scanner scanner = new Scanner(System.in);

for(int i = 0; i < 5; i++){

String str = scanner.nextLine();

int number = Integer.parseInt(str);

System.out.println("you just entered " + number);

}

}

}

class PrimitiveTypes {

public static void main(String [] args){

int myCountValue = 5;

for(int i = 0; i < myCountValue; i++){

for(int j = 0; j < myCountValue; j++){

if(i <= j){

System.out.print('\*' + " ");

}

}

System.out.println();

}

}

}

public class HelloWorld {

public static void main(String [] args){

// for(init; condition; operation)

for(int i = 0; i < 10; i++){

if(i % 3 == 0){

System.out.print("here");

} else if (i % 4 == 0){

break;

}

}

}

}

public class HelloWorld {

public static void main(String [] args){

// for(init; condition; operation)

int i = 10;

int j = 3;

int result = i/j;

System.out.println(result); // 3

result = i % j;

System.out.println(result); //

}

}

for(int i = 0; i < 3; i++){ // i = 2

for(int j = 3; j > 0; j--){

if(i == j) {

System.out.println("Equal");

}else if(i < j) {

System.out.println("i less then j");

}else {

for(int n = 0; n < i; n++){

System.out.print('\*');

}

System.out.println();

}

}

}

public class HelloWorld {

public static void main(String [] args){

boolean bool = true; // true

int i = 0;

while(bool){

if(i == 3){

bool = false;

} else {

System.out.println(i);

}

i++; // i = i + 1;

}

}

}

public class HelloWorld {

public static void main(String [] args){

String value;

value = "Java";

System.out.println(value);

int i = 8;

String newValue = value + " rulezzz!!!!";

newValue = value + i;

System.out.println("i = " + i); // i = 8

}

}

public class HelloWorld {

public static void main(String [] args){

int i; // declare

i = 10; // initialize

int a = i; // declare & initialize

System.out.println(i); // 10

System.out.println(a); // 10

int sum = i + a;

System.out.println(sum); // 20

System.out.println(sum \* sum); // 20

}

}

class PrimitiveTypes {

public static void main(String [] args){

int a = 4;

int b = 7;

int sum = a + b;

int diff = a - b;

boolean bool = a > b;

boolean flag = !bool && (a \* 3 < 7 || 5 < b \* 3);

System.out.println(flag);

if (flag){

System.out.println(sum);

if (sum \* 2 <= 1000){

System.out.println("Less");

} else {

System.out.println("More");

}

} else{

System.out.println(diff);

}

}

}

class PrimitiveTypes {

public static void main(String [] args){

String message = "Primitive types: ";

System.out.println(message);

int i; // declare

i = 10; // initialize

byte b = 300; // declare & initialize

short s = 30000;

long l = 111111111;

System.out.println(i);

System.out.println(b);

System.out.println("s = " + s);

System.out.println("l = " + l);

// =================================

float f = 10.5f;

double d = 10;

System.out.println("f = " + f);

System.out.println("d = " + d);

boolean bool = false;

bool = true;

System.out.println("bool = " + bool);

char c = 'A';

System.out.println("char c = " + c);

}

}