**Задачи. Простейшие классы и объекты.**

1. Создайте класс Test1 двумя переменными. Добавьте метод вывода на экран и методы изменения этих переменных. Добавьте метод, который находит сумму значений этих переменных, и метод, который находит наибольшее значение из этих двух переменных.

package by.htp.les01.start;

public class Test1 {

int x;

int y;

public void printOnTheScreen(int x, int y) {

System.out.println(x + " " + y);

}

public int changingOfvariable(int a) {

return a + 1;

}

public int sumOfvariable(int x, int y) {

return x + y;

}

public int sravnenie(int x, int y) {

if (x >= y) {

return x;

} else {

return y;

}

}

}

1. Создйте класс Test2 двумя переменными. Добавьте конструктор с входными параметрами. Добавьте конструктор, инициализирующий члены класса по умолчанию. Добавьте set- и get- методы для полей экземпляра класса.

package by.htp.les01.start;

public class Test2 {

int x;

int y;

public Test2(int x, int y) {

this.x = x;

this.y = y;

System.out.println("asd");

}

public Test2() {

this.x = 1;

this.y = 2;

System.out.println(x + " " + y);

}

public int getX() {

return x;

}

public void setX(int x) {

this.x = x;

}

public int getY() {

return y;

}

public void setY(int y) {

this.y = y;

}

}

1. Создайте класс с именем Student, содержащий поля: фамилия и инициалы, номер группы, успеваемость (массив из пяти элементов). Создайте массив из десяти элементов такого типа. Добавьте возможность вывода фамилий и номеров групп студентов, имеющих оценки, равные только 9 или 10.

package by.htp.les01.start;

import java.util.Arrays;

public class Student {

private String familiya;

private String inicialy;

private int nomerGruppy;

private int[] uspevaemost = new int[5];

public Student(String familiya, String inicialy, int nomerGruppy) {

this.familiya = familiya;

this.inicialy = inicialy;

this.nomerGruppy = nomerGruppy;

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

uspevaemost[i] = (int) ((Math.random() \* 10) + 1);

if (uspevaemost[i] >= 9) {

System.out.println(familiya + " " + nomerGruppy);

}

}

}

public String getFamiliya() {

return familiya;

}

public void setFamiliya(String familiya) {

this.familiya = familiya;

}

public String getInicialy() {

return inicialy;

}

public void setInicialy(String inicialy) {

this.inicialy = inicialy;

}

public int getNomerGruppy() {

return nomerGruppy;

}

public void setNomerGruppy(int nomerGruppy) {

this.nomerGruppy = nomerGruppy;

}

public int[] getUspevaemost() {

return uspevaemost;

}

public void setUspevaemost(int[] uspevaemost) {

this.uspevaemost = uspevaemost;

}

@Override

public String toString() {

return "Student [familiya=" + familiya + ", inicialy=" + inicialy + ", nomerGruppy=" + nomerGruppy

+ ", uspevaemost=" + Arrays.toString(uspevaemost) + "]";

}

}

**package** by.htp.les01.start;

**import** java.util.ArrayList;

**public** **class** Main {

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

ArrayList<Student> group = **new** ArrayList<>();

group.add(**new** Student("qwe", "aq", 1));

group.add(**new** Student("rty", "aw", 2));

group.add(**new** Student("uio", "ae", 3));

group.add(**new** Student("asd", "ar", 4));

group.add(**new** Student("fgh", "at", 5));

group.add(**new** Student("jkl", "ay", 6));

group.add(**new** Student("zxc", "au", 7));

group.add(**new** Student("vbn", "ai", 8));

group.add(**new** Student("mmm", "ao", 9));

group.add(**new** Student("mmmmmm", "ap", 10));

**for** (**int** i = 0; i < group.size(); i++) {

System.***out***.println(group.get(i));

}

}

}

1. Создайте класс Train, содержащий поля: название пункта назначения, номер поезда, время отправления. Создайте данные в массив из пяти элементов типа Train, добавьте возможность сортировки элементов массива по номерам поездов. Добавьте возможность вывода информации о поезде, номер которого введен пользователем. Добавьте возможность сортировки массив по пункту назначения, причем поезда с одинаковыми пунктами назначения должны быть упорядочены по времени отправления.

public class Train {

protected String punktNaznacheniya;

protected Integer nomerPoezda;

protected Double vremiaOtpravleniya;

public Train(String punktNaznacheniya, Integer nomerPoezda, Double vremiaOtpravleniya) {

this.punktNaznacheniya = punktNaznacheniya;

this.nomerPoezda = nomerPoezda;

this.vremiaOtpravleniya = vremiaOtpravleniya;

}

public String getPunktNaznacheniya() {

return punktNaznacheniya;

}

public void setPunktNaznacheniya(String punktNaznacheniya) {

this.punktNaznacheniya = punktNaznacheniya;

}

public int getNomerPoezda() {

return nomerPoezda;

}

public void setNomerPoezda(int nomerPoezda) {

this.nomerPoezda = nomerPoezda;

}

public double getVremiaOtpravleniya() {

return vremiaOtpravleniya;

}

public void setVremiaOtpravleniya(double vremiaOtpravleniya) {

this.vremiaOtpravleniya = vremiaOtpravleniya;

}

@Override

public String toString() {

return "Train [punktNaznacheniya=" + punktNaznacheniya + ", nomerPoezda=" + nomerPoezda

+ ", vremiaOtpravleniya=" + vremiaOtpravleniya + "]";

}

}

**package** by.htp.les01.start;

**import** java.util.Arrays;

**public** **class** Main {

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

Train[] train = **new** Train[5];

train[0] = **new** Train("qwe", 1, 8.45);

train[1] = **new** Train("qwe", 2, 20.45);

train[2] = **new** Train("qwe", 6, 10.45);

train[3] = **new** Train("asd", 4, 11.45);

train[4] = **new** Train("fgh", 9, 12.45);

Arrays.*sort*(train, (a, b) -> a.punktNaznacheniya.compareTo(b.punktNaznacheniya));

System.***out***.println(Arrays.*asList*(train));

Arrays.*asList*(train).stream().filter(a -> a.punktNaznacheniya.equals("qwe"))

.sorted((a, b) -> a.vremiaOtpravleniya.compareTo(b.vremiaOtpravleniya)).forEach(System.***out***::print);

}

}

5. Опишите класс, реализующий десятичный счетчик, который может увеличивать или уменьшать свое значение на единицу в заданном диапазоне. Предусмотрите инициализацию счетчика значениями по умолчанию и произвольными значениями. Счетчик имеет методы увеличения и уменьшения состояния, и метод позволяющее получить его текущее состояние. Написать код, демонстрирующий все возможности класса.

package by.htp.les01.start;

public class Count {

private int count;

public Count(int count) {

this.count = count;

}

public int getCount() {

return count;

}

public void setCount(int count) {

this.count = count;

}

public void plusCount() {

count++;

}

public void minusCount() {

count--;

if (this.count < 0) {

count = 0;

}

}

}

package by.htp.les01.start;

public class Main {

public static void main(String[] args) {

Count c = new Count(0);

System.out.println(c.getCount());

System.out.println("setCount:");

c.setCount(5);

System.out.println(c.getCount());

System.out.println("plusCount:");

c.plusCount();

System.out.println(c.getCount());

System.out.println("minusCount:");

c.minusCount();

System.out.println(c.getCount());

System.out.println("The End");

}

}

6. Составьте описание класса для представления времени. Предусмотрте возможности установки времени и изменения его отдельных полей (час, минута, секунда) с проверкой допустимости вводимых значений. В случае недопустимых значений полей поле устанавливается в значение 0. Создать методы изменения времени на заданное количество часов, минут и секунд.

7. Описать класс, представляющий треугольник. Предусмотреть методы для создания объектов, вычисления площади, периметра и точки пересечения медиан.

8. Создать класс Customer, спецификация которого приведена ниже. Определить конструкторы, set- и get- методы и метод toString(). Создать второй класс, агрегирующий массив типа Customer, с подходящими конструкторами и методами. Задать критерии выбора данных и вывести эти данные на консоль.

Класс Customer: id, фамилия, имя, отчество, адрес, номер кредитной карточки, номер банковского счета.

Найти и вывести:

a) список покупателей в алфавитном порядке;

b) список покупателей, у которых номер кредитной карточки находится в заданном интервале

9. Создать класс Book, спецификация которого приведена ниже. Определить конструкторы, set- и get- методы и метод toString(). Создать второй класс, агрегирующий массив типа Book, с подходящими конструкторами и методами. Задать критерии выбора данных и вывести эти данные на консоль.

Book: id, название, автор(ы), издательство, год издания, количество страниц, цена, тип переплета.

Найти и вывести:

a) список книг заданного автора;

b) список книг, выпущенных заданным издательством;

c) список книг, выпущенных после заданного года.

10. Создать класс Airline, спецификация которого приведена ниже. Определить конструкторы, set- и get- методы и метод toString(). Создать второй класс, агрегирующий массив типа Airline, с подходящими конструкторами и методами. Задать критерии выбора данных и вывести эти данные на консоль.

Airline: пункт назначения, номер рейса, тип самолета, время вылета, дни недели.

Найти и вывести:

a) список рейсов для заданного пункта назначения;

b) список рейсов для заданного дня недели;

c) список рейсов для заданного дня недели, время вылета для которых больше заданного.