**Programming languages / Java (BSc, 18) lab4**

**Task 1**

Create a program which reads (from standard input) the coordinates of three Point objects, and instantiates three such objects, storing their references in an array.

Then the program should compute the *center of mass* of the points, and print out this point on the screen. (The coordinates of the center of mass is defined as the average of the corresponding coordinates of the original points.)

class Main {

public static void main(String[] args) {

Point[] arr = new Point[3];// declaration and instantiation

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

System.out.print("Enter x = " );

double x = Double.parseDouble(System.console().readLine());

System.out.print("Enter y = " );

double y = Double.parseDouble(System.console().readLine());

arr[i] = new Point(x, y); //

}

double center\_x = 0, center\_y = 0;

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

center\_x += arr[i].getX();

center\_y += arr[i].getY();

}

center\_x /= 3;

center\_y /= 3;

Point center = new Point(center\_x, center\_y);

System.out.println("center of mass: " + center);

}

}

public class Point {

private double x, y;

public Point(double x, double y) {

this.x = x;

this.y = y;

}

public Point(Point that) {

this.x = that.x;

this.y = that.y;

}

public double getX() {

return x;

}

public double getY() {

return y;

}

public void move(double dx, double dy) {

x += dx;

y += dy;

}

public void mirror(double cx, double cy) {

x = 2 \* cx - x;

y = 2 \* cy - y;

}

public void mirror(Point that) {

x = 2 \* that.x - x;

y = 2 \* that.y - y;

}

public double distance(Point that) {

double dx = x - that.x;

double dy = y - that.y;

return Math.sqrt(dx\*dx + dy\*dy);

}

public String toString() {

return "(" + x + "," + y + ")";

}

}

**Task 2**

Modify the previous program in such a way that the computation of the center of mass is performed by a static method of the Point class. This method takes the input points in an array, and provides the result as a return value.

**public class Point {**

**private double x, y;**

**public Point(double x, double y) {**

**this.x = x;**

**this.y = y;**

**}**

**public Point(Point that) {**

**this.x = that.x;**

**this.y = that.y;**

**}**

**public double getX() {**

**return x;**

**}**

**public double getY() {**

**return y;**

**}**

**public void move(double dx, double dy) {**

**x += dx;**

**y += dy;**

**}**

**public void mirror(double cx, double cy) {**

**x = 2 \* cx - x;**

**y = 2 \* cy - y;**

**}**

**public void mirror(Point that) {**

**x = 2 \* that.x - x;**

**y = 2 \* that.y - y;**

**}**

**public double distance(Point that) {**

**double dx = x - that.x;**

**double dy = y - that.y;**

**return Math.sqrt(dx\*dx + dy\*dy);**

**}**

**public String toString() {**

**return "(" + x + "," + y + ")";**

**}**

**public static Point centerOfMass(Point[] arr) {**

**double center\_x = 0, center\_y = 0;**

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

**center\_x += arr[i].getX();**

**center\_y += arr[i].getY();**

**}**

**center\_x /= arr.length;**

**center\_y /= arr.length;**

**return new Point(center\_x, center\_y);**

**}**

**}**

**class Main {**

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

**System.out.print("Enter number of Points: ");**

**int num = Integer.parseInt(System.console().readLine());**

**Point[] arr = new Point[num];**

**for (int i = 0; i < num; ++i) {**

**System.out.print("Enter x = " );**

**double x = Double.parseDouble(System.console().readLine());**

**System.out.print("Enter y = " );**

**double y = Double.parseDouble(System.console().readLine());**

**arr[i] = new Point(x, y);**

**}**

**Point center = Point.centerOfMass(arr);**

**System.out.println("center of mass: " + center);**

**}**

**}**

**Task 3**

Draw the memory map of the following Java programs. (Present the contents of the execution stack and the heap during the execution of the programs.)

Main.java:

class Foo {

private int x;

public Foo(int x) {

this.x = x;

}

}

public class Main {

public static void main(String[] args) {

int counter = 0;

Foo obj = new Foo(5);

counter = 10;

obj = new Foo(7);

// 1. Which objects can be removed by the garbage collector?

Foo obj2;

new Foo(20); //

obj2 = obj;

// 2. Which objects can be removed by the garbage collector?

obj2 = new Foo(30); // obj2 is a reference, Foo is an object

obj2 = new Foo(20);// another Foo object has been created, and obj2 now contain the address of the new Foo object.

}

}

Main2.java:

public class Main2 {

public static void main(String[] args) {

String s1;

System.out.println("len of s1 = " + s1.length());

String s2 = "";

System.out.println("len of s2 = " + s2.length());

s2 += "hello";

s2 += "world";

// Which objects can be removed by the garbage collector?

System.out.println("len of s2 = " + s2.length());

}

}

**Task 4**

Create the class IntegerMatrix with the following operations.

1. A constructor expecting three arguments:
   * int rowNum (number of rows in the matrix);
   * int colNum (number of columns in the matrix);
   * int[] linearData (an array storing the elements of the matrix in a row-major manner, i.e. {1,2,3,4,5,6) for the \(2 \times 3\) matrix below).

\[ \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \]

* A toString() method, which produces a character string from the matrix elements, separating rows with semicolon (;), and elements within a row with comma (,). The string representing the matrix above is: "1,2,3;4,5,6".

**public class IntegerMatrix {**

**private int rowNum;**

**private int colNum;**

**private int[][] data;**

**public IntegerMatrix(int rowNum, int colNum, int[] linearData){**

**if(rowNum\*colNum != linearData.length)**

**throw new IllegalArgumentException("rowNum\*colNum != linearData.length");**

**this.rowNum = rowNum;**

**this.colNum = colNum;**

**data = new int[rowNum][colNum];**

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

**int row = (int) Math.floor(i / colNum); // tĂ­puskĂŠnyszerĂ­tĂŠs**

**int col = i % colNum;**

**System.out.println(i + " " + row + " " +col );**

**data[row][col] = linearData[i];**

**}**

**}**

**public String toString(){**

**// return toString1(); // BAJ: It is hard to read, and the terms and conditions are ineffective, and the value is faster.**

**// return toString2(); // BAJ: The String immutable, every time a concatenation is a new instance, and must always be logged in.O(n^3)**

**return toString3();**

**}**

**public String toString1(){**

**String s = "";**

**// first line**

**if(data.length < 1) return s;**

**// first line first item**

**s += data[0].length < 1 ? "" : data[0][0];**

**// first line more items**

**for(int j=1; j<data[0].length; j++){**

**s += "," + data[0][j];**

**}**

**// other lines**

**for(int i=1; i<data.length; i++){**

**s += ";";**

**// i-th row first element**

**s += data[i].length < 1 ? "" : data[i][0];**

**// i-th row of other elements**

**for(int j=1; j<data[i].length; j++){**

**s += "," + data[i][j];**

**}**

**}**

**return s;**

**}**

**public String toString2(){**

**String s = "";**

**String rowDelim = "";**

**String colDelim = "";**

**for(int[] row : data){**

**s += rowDelim;**

**rowDelim = ";";**

**for(int elem :row){**

**s += colDelim;**

**colDelim = ",";**

**s += elem;**

**}**

**colDelim = "";**

**}**

**return s;**

**}**

**public String toString3(){**

**StringBuilder sb = new StringBuilder();**

**String rowDelim = "";**

**String colDelim = "";**

**for(int[] row : data){**

**sb.append(rowDelim);**

**rowDelim = ";";**

**for(int elem : row){**

**sb.append(colDelim);**

**colDelim = ",";**

**sb.append(elem);**

**}**

**colDelim = "";**

**}**

**return sb.toString();**

**}**

**}**

**public class Main {**

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

**int[] linearData = {0, 0, 1, 2, 3, 4};**

**System.out.println(new IntegerMatrix(2,3, linearData));**

**}**

**}**

**Task5:**

Create the IntVector class in package util, which represents a sequence of integer numbers! The class should have a field of type integer array, this will store the elements of the sequence. The class should have a constructor, which takes an array of integer numbers and initializes the IntVector with these numbers. (Make sure that the inner representation of the class does not escape from the class definition!)

Add a method named add(int n), which adds n to each element of the sequence! Prepare a toString() method producing a character string. This method should enumerate the elements of the sequence in brackets, separating consecutive elements with a space (for example: [1 2 3]).

**import util.IntVector;**

**class IntVectorDemo {**

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

**int[] ns = new int[] {1,2,3};**

**IntVector v = new IntVector(ns);**

**IntVector v2 = new IntVector(ns);**

**System.out.println(new int[]{1,2,3});**

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

**System.out.println(v2);**

**System.out.println("v.add(1);");**

**v.add(1);**

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

**System.out.println(v2);**

**}**

**}**

**package util;**

**import java.util.Arrays;**

**public class IntVector {**

**private int[] ns;**

**public IntVector(int[] numbers) {**

**// 1.**

**ns = new int[numbers.length];**

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

**ns[i] = numbers[i];**

**// 2.**

**ns = Arrays.copyOf(numbers, numbers.length);**

**}**

**public void add(int n) {**

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

**ns[i] += n;**

**}**

**public String toString() {**

**// return Arrays.toString(ns);**

**StringBuilder sb = new StringBuilder();**

**String delim = "";**

**for(int n : ns){**

**sb.append(delim);**

**sb.append(n);**

**delim = " ";**

**}**

**return sb.toString();**

**}**

**}**

Some related links:

For Task1:

* + 1. Difference between static method and non-static method:

<https://beginnersbook.com/2013/05/static-vs-non-static-methods/>

* + 1. Java array

<https://www.javatpoint.com/array-in-java>

* + 1. How to calculate center of Mass

<https://sciencing.com/calculate-frictional-torque-8621540.html>

4- “this” keyword

<https://www.javatpoint.com/this-keyword>

for task 4:

1. toString() method

<https://www.javatpoint.com/understanding-toString()-method>

1. Java for-each loop

<https://www.geeksforgeeks.org/for-each-loop-in-java/>

for task 5:

1. copyOf() method

<https://www.tutorialspoint.com/java/util/arrays_copyof_int.htm#:~:text=copyOf(int%5B%5D%20original%2Cint,arrays%20will%20contain%20identical%20values>.

1. Append () method

<https://www.geeksforgeeks.org/stringbuilder-append-method-in-java-with-examples/>