Rozdělení do více metod a přidání javadocu

```java
public static double[][] InputPrimky() {
    double ax, by, x, y;
    System.out.println("Zadej body primky:");
    // vezme body primky
    double[][] primka = new double[3][3];
    for (int i = 0; i < 2; i++) {
        primka[i][0] = sc.nextDouble();
        primka[i][1] = sc.nextDouble();
    }
    if (primka[0][0] > primka[1][0]) {
        x = primka[0][0];
        y = primka[0][1];
        ax = primka[0][0] - primka[1][0]; // ax
        by = primka[0][1] - primka[1][1]; // by
    } else {
        x = primka[1][0];
        y = primka[1][1];
        ax = primka[1][0] - primka[0][0]; // ax
        by = primka[1][1] - primka[0][1]; // by
    }
    double c = -(ax * x) - (by * y);
    primka[2][0] = ax;
    primka[2][1] = by;
    primka[2][2] = c;
    return primka;
}
```

Před úpravou

```java
public static double[][] inputPrimky() {
    /**
     * input of a line by 2 points
     */
    System.out.println("Zadej body primky:");
    double[][] primka = new double[2][2];
    for (int i = 0; i < 2; i++) {
        primka[i][0] = sc.nextDouble();
        primka[i][1] = sc.nextDouble();
    }
    return primka;
}

public static double[] vypocetRcePrimky(double[][] primka) {
    /**
     * calculation of an normal equation of line
     * "ax+by+c=0"
     */
    double[] rce = new double[3];
    double a, b, x, y;
    if (primka[0][0] > primka[1][0]) { // 1. point is more right on axis X then 2. point
        x = primka[0][0];
        y = primka[0][1];
        a = primka[0][0] - primka[1][0]; // ax
        b = primka[0][1] - primka[1][1]; // by
    } else {
        x = primka[1][0];
        y = primka[1][1];
        a = primka[1][0] - primka[0][0]; // ax
        b = primka[1][1] - primka[0][1]; // by
    }
    double c = -(a * x) - (b * y);
    rce[0] = a;
    rce[1] = b;
    rce[2] = c;
    return rce;
}
```

Po úpravě

Rozdělení do více metod, přidání bubblesortu, javadocu

```java
public static double[][] SortSouradnic(double[][] body, double[][] primka) {
    // provede sort v zadanych bodech
    double[][] sortVypis = new double[body.length][body[0].length - 1];
    double[] vypocet = new double[body.length];
    double a = primka[2][0];
    double b = primka[2][1];
    double c = primka[2][2];
    for (int i = 0; i < body.length; i++) {
        // vypocet vzdalenosti bodu od primky a importovani do tretiho sloupce
        double a1 = body[i][0];
        double a2 = body[i][1];
        double v;// promenne do vzorce
        if (a == 0 && a1 == 0) {
            v = 0;
        } else if (b == 0 && a2 == 0) {
            v = 0;
        } else {
            v = Math.abs(a * a1 + b * a2 + c) / Math.sqrt(Math.pow(a, 2) + Math.pow(b, 2));
        }
        vypocet[i] = v;
        body[i][2] = v;
    }

    Arrays.sort(vypocet);
    /*
     * double temp = 0;
     * for (int i = 0; i < vypocet.length; i++) {
     * if ((i + 1) != vypocet.length && vypocet[i] > vypocet[i + 1]) {
     * temp = vypocet[i];
     * vypocet[i] = vypocet[i + 1];
     * vypocet[i + 1] = temp;
     * i = 0;
     * }
     * }
     */

    // mozno udelat takto, ale rychlejsi pres "Arrays.sort(vypocet)"

    // serazeni vysledku
    for (int i = 0; i < vypocet.length; i++) {// projede celou puvodni matici
        double temp = vypocet[i];
        for (int ii = 0; ii < body.length; ii++) {
            if (temp == body[ii][2]) {
                sortVypis[i][0] = body[ii][0];
                sortVypis[i][1] = body[ii][1];
                body[ii][2] = 42069;
                break;
            }
        }
    }
    return sortVypis;
}
```

Před úpravou

```java
public static double[][] vypocetVzdalenosti(double[][] body, double[] rce) {
    /**
     * method for calculation of a distance between points and line
     */
    double a = rce[0];
    double b = rce[1];
    double c = rce[2];
    for (int i = 0; i < body.length; i++) {
        double a1 = body[i][0];
        double a2 = body[i][1];
        double v; // variables for equation
        if (a == 0 && a1 == 0) {
            v = 0;
        } else if (b == 0 && a2 == 0) {
            v = 0;
        } else { // equation
            v = Math.abs(a * a1 + b * a2 + c) / Math.sqrt(Math.pow(a, 2) + Math.pow(b, 2));
        }
        body[i][2] = v;
    }
    return body;
}

public static double[][] sortVzdalenosti(double[][] body) {
    /**
     * method that sorts inputed 2D array via bubble sort
     */
    double[] tempPole = new double[2];
    for (int i = 0; i < body.length; i++) {
        for (int j = 1; j < (body.length - i); j++) {
            if (body[j - 1][2] > body[j][2]) {
                tempPole = body[j - 1];
                body[j - 1] = body[j];
                body[j] = tempPole;
            }
        }
    }
    return body;
}
```

Po úpravě

Přidání variability a javadocu

```java
public static void vypisSouradnic(double[][] matrix) {
    System.out.println("Setridene body:");
    for (int i = 0; i < matrix.length; i++) {
        String temp = "";
        for (int ii = 0; ii < matrix[i].length; ii++) {
            temp = temp + matrix[i][ii] + " ";
        }
        System.out.println(temp);
    }
}
```

Před úpravou

```java
public static void vypisSouradnic(double[][] matrix) {
    /**
     * method that prints out sorted 2D array
     */
    System.out.println("Setridene body:");
    for (int i = 0; i < matrix.length; i++) {
        String temp = "";
        for (int ii = 0; ii < matrix[i].length - 1; ii++) { /**
                                                             * can print out even the distance; change
                                                             * "matrix[i].length-1" to "matrix[i].length"
                                                             */
            temp = temp + matrix[i][ii] + " ";
        }
        System.out.println(temp);
    }
}
```

Po úpravě

Použití více metod v main switchi

```
switch (inRozhodnuti) { // switch for input
    case "a", "A" -> { // continue
        // main function
        double[][] primka = tools.InputPrimky(); // input for line
        double[][] body = tools.InputSouradnic(); // input for points
        double[][] vypis = tools.SortSouradnic(body, primka); // sorts points
        tools.vypisSouradnic(vypis); // print

    }
    case "n", "N" -> { // stop
```

Před úpravou

```
switch (inRozhodnuti) { // switch for input
    case "a", "A" -> { // continue
        // main function
        double[][] primka = tools.inputPrimky();
        double[] rce = tools.vypocetRcePrimky(primka);
        double[][] body = tools.inputSouradnic();
        double[][] bodySeVzdalenosti = tools.vypocetVzdalenosti(body, rce);
        double[][] vypis = tools.sortVzdalenosti(bodySeVzdalenosti);
        tools.vypisSouradnic(vypis);
    }
    case "n", "N" -> { // stop
        // program stops
```

Po úpravě

Lépe formulovaný text

Navrh reseni

1. Rozdělení UI a metod pro vstup a počítání s body do odlišných souborů
2. Vytvořit UI se vstupem pro pokračování programu, ze kterého se pak budou volat metody do dalšího souboru
3. Vytvoření metod pro každou operaci, která je třeba v tomto programu
   a) Vstup přímky
      i. Vložení zadaných bodů do 2D pole o rozměrech 2x2
      ii. Následné vypočítání vzorce přímky, který se vloží do jednorozměrného pole, se kterým dále program počítá
   b) Vstup bodů
      i. Zjistit kolik bodů se bude načítat (dále v tomto bodě referováno jako „**K**“)
      ii. Vložení zadaných bodů do 2D pole o rozměrech **K**x3
   c) Výpočet vzdálenosti bodů od přímky a setřídění bodů
      i. Výpočet vzdálenosti bodu od přímky na základě 2 předešlých polí (zde je aplikována podmínka pro speciální případy, aby program zjistil, jaký bod je na ose OXY více vpravo)
      ii. Tyto hodnoty se pak vloží do dvou 2D polí, do prvního pro setřídění a do druhého pole, kde jsou zadané body, a vloží se do 3. sloupce (dále využíváno při třídění a výpisu do setříděného pole)
      iii. Pole pouze s vypočítanými hodnotami se setřídí
      iv. Dále následuje setřídění a přepis proměnných do nového pole, které se z metody vrací.
   d) Vypsání požadovaných hodnot do konzole
      i. Vezme si pole z předešlé metody a setříděné hodnoty vypíše do konzole, tak jak jsou zapsána v poli
4. Revize kódu, jestli je někde místo na zkrácení, či vylepšení

Před úpravou

Zadání semestrální práce

Zapište program, který nejprve načte přímku zadanou dvěma body. Potom má program načíst libovolnou sadu bodů a setřídit je na základě vzdálenosti bodu od přímky.

Body mohou být zadávány jakou desetinné číslo, aby tento program byl více variabilní pro případné budoucí využití. Také tento program umí operovat, pokud zadaná přímka leží na osách **X** a **Y**, či jestli je s těmito osami rovnoběžná. V tomto případě, pokud budou zadané body pro třídění ležet na zadané přímce setřídí se podle toho, jaký byl zadán dřív.

Návrh řešení

1. Rozdělení UI a metod pro vstup a počítání s body a polemi do odlišných souborů
2. Vytvořit UI se vstupem pro pokračování programu, ze kterého se pak budou volat metody do dalšího souboru
3. Vytvoření metod pro každou operaci, která je třeba v tomto programu
   a) Vstup přímky
      i. Vložení 2 bodů, kterými prochází přímka do 2D pole o rozměrech 2x2
      ii. Následné vypočítání vzorce přímky (*ax+by+c=0)*, který se vloží na poslední řádek pole se zadanými body, se kterým dále program počítá
   b) Vstup bodů
      i. Zjistit kolik bodů se bude načítat (dále v tomto bodě referováno jako „***k***“)
      ii. Vložení zadaných bodů do 2D pole o rozměrech ***k***x3 - 1. sloupec pro X souřadnici bodu, 2. sloupec pro Y souřadnici bodu a 3. sloupec pro následně vypočítanou vzdálenost
   c) Výpočet vzdálenosti bodů od přímky a setřídění bodů
      i. Výpočet vzdálenosti bodu od přímky na základě 2 předešlých polí (zde je aplikována podmínka pro speciální případy, aby program zjistil, jaký bod je na ose OXY více vpravo a když je přímka rovnoběžná s osami OXY)
      ii. vzorec:

$$v(A,p) = \frac{|a \cdot a_1 + b \cdot a_2 + c|}{\sqrt{a^2 + b^2}}$$

a,b,c – jsou proměnné ze vzorce přímky
a_1, a_2 – X,Y souřadnice zadaného bodu pro setřídění

      iii. Vypočítané vzdálenosti se vloží do pole s body do 3. sloupce na řádek k danému bodu se kterým se počítalo (a_1, a_2)
      iv. 2D pole se setřídí pomocí *Bubblesortu* dle vypočítaných vzdáleností
   d) Vypsání požadovaných hodnot do konzole
      i. Vezme si pole z předešlé metody a setříděné body vypíše do konzole, tak jak jsou zapsána v poli (po menší změně kódu může vypsat i vzdálenost bodů)
4. Revize kódu, jestli je někde místo na zkrácení, či vylepšení

Po úpravě

Přidání screenshotů z testů

Screenshoty z testů

```
Spoustim semestralni praci
Pokracovat ve zpracovani (a/n):
a
Zadej body primky:
1 0
-1 0
Zadejte pocet bodu:
4
Zadejte souradnice bodu:
10,32 0
-3 -8
15 20
1 0
Setridene body:
10.32 0.0
1.0 0.0
-3.0 -8.0
15.0 20.0
```

Screenshot k testu 1

```
Pokracovat ve zpracovani (a/n):
a
Zadej body primky:
0 1
0 -1
Zadejte pocet bodu:
4
Zadejte souradnice bodu:
0 10,32
-8 -3
20 15
0 1
Setridene body:
0.0 10.32
0.0 1.0
-8.0 -3.0
20.0 15.0
```

Screenshot k testu 2

```
Pokracovat ve zpracovani (a/n):
a
Zadej body primky:
5 2
3 1
Zadejte pocet bodu:
4
Zadejte souradnice bodu:
2 1
3 6
15 20
42 69
Setridene body:
3.0 6.0
2.0 1.0
15.0 20.0
42.0 69.0
```

Screenshot k testu 3

```
Pokracovat ve zpracovani (a/n):
a
Zadej body primky:
0 0
0 0
Zadejte pocet bodu:
4
Zadejte souradnice bodu:
12 0
1 0
12 13
42 69
Setridene body:
12.0 0.0
1.0 0.0
12.0 13.0
42.0 69.0
```

Screenshot k testu 4