ZÁPADOČESKÁ
UNIVERZITA
V PLZNI

Samostatná práce

z předmětu

Počítače a programování 1

Jméno a příjmení: *Martin Sloup*
Osobní číslo: *A04372*
Studijní skupina: *13*
Obor: *INI*
E-mail: *sloup@students.zcu.cz*

Označení zadání: *a) řazení náhodných prvků*

Datum odevzdání: 8.12.2004

Zadání práce:

Napište program v jazyku Java pro realizaci třech následujících algoritmů řazení (vzestupně i sestupně) metodami Insert sort(vkládáním), Select sort (výběrem) a Bubble sort (bublinkové řazení). Pracujte s celými nebo reálnými hodnotami prvků generovanými generátorem náhodných čísel. Součástí metod bude zjištění počet výměn (přesunů) prvků a počet jejich porovnání. Pro všechny tři metody vytvořte tabulku závislosti buď počtu výměn (přesunů) nebo počtu porovnání na velikosti dat při řazení jednou z variant zadání dat (a, b, c, d). Tabulku uložte do výstupního textového souboru a exportujte do MS Wordu či MS Excelu, kde vytvoříte graf závislosti.Tabulka i graf budou rovněž součástí dokumentace.

Varianty zadání pro vytvoření tabulky a grafů:

a) řazení náhodných prvků
b) řazení částečně již seřazených prvků
c) řazení již seřazených prvků
d) řazení opačně seřazených prvků

Použité zadání:

a) řazení náhodných prvků

Uživatelská příručka:

Softwarové řešení problému jsem rozdělil pro přehlednost na dva programy: Hlavní a pomocný.

Oba programy jsou dodány ve zkomprimované podobě, tedy v JAR archívu. Oba programy lze spustit pod MS Windows jejich dávkovými soubory (soubory s příponou bat). Pro Hlavní program je to **TestTrideni.bat** a pro pomocný program zas **GenTabulku.bat**. Pod jiným operačním systémem (jako třeba Linux, Unix, …) program spustíme příkazem **„java -jar TestTrideni.jar“** a **„java -jar GenTabulku.jar“.**

Hlavní program

Tento program slouží k ilustraci použití třídících metod. Po spuštění programu se na obrazovku nejprve vypíše nabídka, zda chcete uchovat v souboru vše, co bylo provedeno. Pak následuje hlavní nabídka, kde si můžete vybrat, jaké pole chcete vytvořit („1“=pole s náhodně generovanými prvky; „2“=pole s ručně zadanými prvky; a volba „3“ pro ukončení programu). Po výběru volby 1 se vás zeptá, jak velké pole chcete vytvořit. Při volbě „2“ se program taktéž zeptá na velikost pole, ale i se zeptá na samotné prvky.

Dále se vás program zeptá, zda chcete vypsat obsah pole. To se náramně hodí, pokud jsme nechali vytvořit pole naplněné náhodnými čísly. Pokud zvolíte volbu „1“, program provede výpis pole na obrazovku.

V následujícím kroku si budeme moct vybrat metodu kterou bude pole setříděno. Na výběr jsou metody pro Insert, Select a Bubble sort oběma směry, tedy vzestupně i sestupně. Takže je k dispozici 6 voleb: 1=InsertSort vzestupně, 2=InsertSort sestupně, 3=SelectSort vzestupně, atd.

Po setřídění je možné zase vypsat obsah pole, tedy už setříděného pole.

Nakonec nám program vypíše lehkou statistiku: počet porovnaní a přehozů (přesunů).

Pokud jsme na začátku programu vybrali zápis do souboru, tak nyní se můžeme podívat, najdeme ho ve stejném adresáři jako je program. Jméno souboru bude **log.txt**.

Pomocný program

Pro přímé generování tabulky jsem vytvořil ještě další program: „Pomocný program.“ Ten se stará o vytvoření potřebných tabulkových dat (počet přehozů / přesunů a porovnání) pro následné vygenerování grafu. Výstup je uložen do speciálního souboru s příponou csv (Comma Separated Variable). To kvůli snadnému importu do tabulkového procesoru.

Po spuštění programu se vás program zeptá na počáteční a koncovou velikost generovaného pole. To se hodí, hlavně když potřebujeme vypsat počet přehozů / přesunů a porovnání třeba pro prvních deset polí.

V programu následuje volba: „Co všechno má obsahovat výstupní soubor?“ Potřeboval jsem totiž vypsat jen počet výměn (přesunů) a tato volba se právě na toto hodí.

Po zadání potřebných hodnot začne program generovat do souboru potřebná tabulková data. Výsledek bude uložen v souboru **bigtable.csv**. Formát je rozepsán dále v tomto dokumentu („Příklad výstupního souboru…“).

Programátorská příručka:

Program je napsán v programovacím jazyku Java a potřebuje pro své spuštění mít nainstalované minimálně JRE 1.4. Na starším JRE by to sice mohlo fungovat, ale potřeboval by znovu překompilovat pro starší JRE.

Program je psán tak, aby byl přenositelný i na jiné systémy než OS Microsoft Windows, kde byl původně vyvíjen.

V programu byli použity a následně implantovány následující třídící algoritmy:

InsertSort

Tento způsob třídění připomíná řazení karet. Postupně vybíráme jednu kartu za druhou a zařazujeme je na odpovídající místo podle barvy a hodnoty - klíče, ostatní ještě nezatříděné posuneme o jednu pozici. Pro jednoduché pochopení tedy:

1) První prvek pole ponecháme na svém místě.
2) Vezmeme druhý prvek a porovnáme jej s prvním. Je-li menší, zařadíme ho na první místo a první prvek posuneme, jinak je ponecháme na místě.
3) Vezmeme třetí prvek a porovnáme jej s prvními dvěma prvky. Je-li menší než některý z nich, zařadíme jej na odpovídající pozici a následující prvky podle potřeby posuneme. Jinak je ponecháme na původních místech.
4) Obdobně postupujeme i s ostatními prvky v poli.

Pokud by jsme to chtěli napsat tak, aby se pole řadilo sestupně, musíme prvky přesouvat způsobem takovým, že když aktuální prvek je větší než předchozí, provádíme potřebný přesun, jak je naznačeno ve 2 a 3 kroku.

SelectSort

Pro jednoduchost si představme posloupnost rozdělenou do dvou částí. V jedné části jsou již setříděné prvky, zatímco druhá část je nesetříděná. V nesetříděné části najdeme nejmenší prvek a přesuneme ho na konec setříděné části (vyměníme s posledním prvkem setříděné části). Což dá ve třech krocích programátorsky napsat:

1) V posloupnosti najdeme nejmenší prvek a vyměníme ho s prvkem na první pozici. Tím dojde k rozdělení posloupnosti na dvě části. Setříděná část obsahuje pouze jeden prvek, nesetříděná N-1.

2) V nesetříděné části najdeme nejmenší prvek a vyměníme ho s prvním prvkem v nesetříděné části, čímž dojde k zařazení tohoto prvku do setříděné části.
3) Obsahuje-li nesetříděná část více než jeden prvek, pokračujeme bodem 2, jinak je třídění ukončeno.

Pokud by jsme to chtěli napsat tak, aby se pole řadilo pozpátku, stačí hledat ve druhém kroku největší prvek.

BubbleSort

BubbleSort je třídění přímou výměnou sousedních prvků. To znamená, že procházíme polem a porovnáváme dva sousední prvky. V případě, že nejsou v požadovaném pořadí, prohodíme je. Takže algoritmus vypadá následovně:

1) Posloupnost rozdělíme na dvě části, setříděnou a nesetříděnou.Setříděná část je prázdná.
2) Postupně porovnáme všechny sousední prvky v nesetříděné části a pokud nejsou v požadovaném pořadí, prohodíme je.
3) Krok 2 opakujeme tak dlouho, dokud nesetříděná část obsahuje více než jeden prvek. Jinak algoritmus končí.

Algoritmus BubbleSortu jsem naprogramoval s modifikací 3 kroku. To znamená: Pokud nastala v kroku 2 alespoň jedna výměna, došlo ke zmenšení nesetříděné části o jeden prvek, pokračujeme bodem 2. Jinak je třídění ukončeno.

Pro seřazení pole sestupně lze postupovat následujícími dvěmi způsoby. Buď v obou cyklech typu for prohodíme **výraz_start** a **výraz_stop** a nebo v podmínce vyměníme relační operátor menší (znak „<“) za větší (znak „>“).

Popis použitých souborů, metod a tříd (abecedně):

FileLog.java (třída FileLog): – třída pro zápis do souboru
- constructor **FileLog**(String) – konstruktor obj.; vytvoří objekt se zadáním souboru (parametr)
- void **finalize**() – destruktor objektu; provede uzavření souboru, pokud je soubor otevřen
- void **print**(String) – zapsání textu (parametr) do souboru
- void **println**() – zapsaní nového řádku do souboru
- void **println**(String) – zapsání textu (parametr) do souboru a přidání nového řádku
- void **close**() – uzavření souboru

Main.java (třída Main): – třída s hlavním programem předvádějící třídění
- static void **Main**(String []) – spouštěcí metoda hlavního programu

Tabulka.java (třída Tabulka): – třída s pomocným programem generující potřebnou tabulku pro graf
- static void **Main**(String []) – spouštěcí metoda pomocného programu

Tools.java (třída Tools): – třída shromažďující všechny pomocné statické metody
- static int[] **RandomArray**(int) – vrací pole určité velikosti (parametr) naplněné náhodnými čísly
- static void **hlavickaProgramu**() – vypíše na obrazovku hlavičku programu

Trideni.java (třída Trideni): – třída zastřešující všechny třídící metody
- int **porovnani** – proměnná uchovávající počet porovnání poslední volané třídící metody
- int **prehozu** – proměnná uchovávající počet přesunů / přehozů poslední volané třídící metody
- void **InsertSortASC**(int[]) – metoda pro třídění pole (parametr) vkládáním vzestupně
- void **InsertSortDESC**(int[]) – metoda pro třídění pole (parametr) vkládáním sestupně

void **SelectSortASC**(int[]) – metoda pro třídění pole (parametr) výběrem vzestupně
void **SelectSortDESC**(int[]) – metoda pro třídění pole (parametr) výběrem sestupně
void **BubbleSortASC**(int[]) – metoda pro třídění pole (parametr) bublinkově vzestupně
void **BubbleSortDESC**(int[]) – metoda pro třídění pole (parametr) bublinkově sestupně

VstupData.java (třída VstupData): – pomocná, volně dostupná třída na čtení dat z klávesnice
static boolean **ctiBoolean**() – vrací hodnotu typu boolean načtenou z klávesnice
static byte **ctiByte**() – vrací hodnotu typu byte načtenou z klávesnice
static short **ctiShort**() – vrací hodnotu typu short načtenou z klávesnice
static int **ctiInt**() – vrací hodnotu typu int načtenou z klávesnice
static long **ctiLong**() – vrací hodnotu typu long načtenou z klávesnice
static float **ctiFloat**() – vrací hodnotu typu float načtenou z klávesnice
static double **ctiDouble**() – vrací hodnotu typu double načtenou z klávesnice
static char **ctiChar**() – vrací hodnotu typu char načtenou z klávesnice
static String **ctiString**() – vrací hodnotu typu String načtenou z klávesnice

Příklad výstupního souboru „bigtable.csv“ z pomocného programu

1;0;0;0;0;1;1;1;1;0;0;0;0;
2;1;1;1;0;3;2;3;2;1;1;0;0;
3;3;2;3;1;6;3;6;3;3;2;2;1;
4;4;2;6;4;10;4;10;4;6;2;5;3;
5;5;2;10;8;15;5;15;5;10;2;9;7;
6;9;5;13;10;21;6;21;6;15;5;14;10;
7;12;7;18;14;28;7;28;7;21;7;20;14;
8;19;14;19;14;36;8;36;8;28;14;27;14;
9;26;20;22;16;45;9;45;9;36;20;35;16;
10;31;24;28;21;55;10;55;10;45;24;44;21;
11;39;31;32;24;66;11;66;11;55;31;54;24;
12;44;35;40;31;78;12;78;12;66;35;65;31;
13;48;38;50;40;91;13;91;13;78;38;77;40;
14;60;49;53;42;105;14;105;14;91;49;90;42;
15;63;51;66;54;120;15;120;15;105;51;104;54;
16;71;58;75;62;136;16;136;16;120;58;119;62;
17;83;69;81;67;153;17;153;17;136;69;135;67;
18;96;81;87;72;171;18;171;18;153;81;152;72;
19;98;82;105;89;190;19;190;19;171;82;170;89;
20;117;100;107;90;210;20;210;20;190;100;189;90;

Řazení čísel z leva:

1. Délka pole

2. Počet porovnání při řazení vkládáním vzestupně (pro volbu 1 a 3)
3. Počet přesunů (přehozů) při řazení vkládání vzestupně (pro volbu 2 a 3)
4. Počet porovnání při řazení vkládáním sestupně (pro volbu 1 a 3)
5. Počet přesunů (přehozů) při řazení vkládáním sestupně (pro volbu 2 a 3)

6. Počet porovnání při řazení výběrem vzestupně (pro volbu 1 a 3)
7. Počet přesunů (přehozů) při řazení výběrem vzestupně (pro volbu 2 a 3)
8. Počet porovnání při řazení výběrem sestupně (pro volbu 1 a 3)

9. Počet přesunů (přehozů) při řazení výběrem sestupně (pro volbu 2 a 3)

10. Počet porovnání při bublinkovém řazení vzestupně (pro volbu 1 a 3)
11. Počet přesunů (přehozů) při bublinkovém řazení vzestupně (pro volbu 2 a 3)
12. Počet porovnání při bublinkovém řazení sestupně (pro volbu 1 a 3)
13. Počet přesunů (přehozů) při bublinkovém řazení sestupně (pro volbu 2 a 3)

Zdrojový kód:

soubor FileLog.java:

```
/**
 * Trida na vytvareni logovaciho souboru.
 *
 * FILE: FileLog.java
 * @author Martin Sloup (A04372), 13. krouzek, FAV - Informatika
 * @version 1.0, 12/04/04
 */
// naimportujeme potrebne balicky
import java.io.*;

public class FileLog {
	private boolean is_open = false;
	private PrintWriter fPrint;
	private File oFile;

	/**
	 * Konstruktor objektu; otevreni souboru, pripadne jeho zalozeni
	 */
	public FileLog(String filename) {
		// osetrime vyjjimky
		try {
			// vytvoreni objektu typu File
			this.oFile = new File(filename);
			// existuje soubor? pokud ne, zalozime ho
			if (!this.oFile.exists()) this.oFile.createNewFile();
			// vytvoreni objektu typu PrintWriter
			this.fPrint = new PrintWriter(new FileWriter(oFile));
			this.is_open = true;
		} catch(IOException e) {
			this.is_open = false;
			System.out.println("Nastala chyba pri praci se souborem!");
			e.printStackTrace();
		}
	}

	/**
	 * Destruktor objektu; uzavreni souboru, pokud jsme ho zapomneli zavrit
	 */
	protected void finalize() throws Throwable {
		// pokud je otevren soubor, uzavreme ho
		if (this.is_open) {
			this.fPrint.close();
			this.fPrint = null;
			this.oFile = null;
		}
		super.finalize();
	}

	/**
	 * Zapis do souboru
	 */
```

```
	public void print(String text) {
		// pokud je otevren soubor, zapisem do nej
		if (this.is_open) this.fPrint.print(text);
	}

	/**
	 * Zapis do souboru a pridani noveho radku
	 */
	public void println(String text) {
		// pokud je otevren soubor, zapisem do nej
		if (this.is_open) this.fPrint.println(text);
	}

	/**
	 * Zapis do souboru a pridani noveho radku
	 */
	public void println() {
		// pokud je otevren soubor, zapisem do nej
		if (this.is_open) this.fPrint.println();
	}

	/**
	 * Uzavreni souboru
	 */
	public void close() {
		// pokud je otevren soubor, uzavreme ho
		if (this.is_open) {
			this.fPrint.close();
			this.fPrint = null;
			this.oFile = null;
		}
	}
}
```

soubor Main.java:

```
/**
 * Hlavni program predvadejici tridu Trideni.
 *
 * FILE: Main.java
 * @author Martin Sloup (A04372), 13. krouzek, FAV - Informatika
 * @version 1.0, 12/04/04
 */
// naimportujeme potrebne balicky
import java.io.*;

public class Main {
	public static void main(String[] args) {
		// v cele ukazce budeme pouzivat pole, tak ho tady definujeme
		int[] pole;
		// promena pro ulozeni klavesy z volby
		int klavesa = 0;

		// vytvoreni objektu pro zapis do souboru
		FileLog log = null;
		// vytvoreni objektu pro trideni
		Trideni sort = new Trideni();

		// zobrazime hlavicku programu
		Tools.hlavickaProgramu();

		// kez by existoval prikaz na vymaz obrzovky. Tak to tedy bude
neprehledne
```

```
            //zeptame se, zda se bude logovat do souboru
            do {
                  System.out.println("* Chcete ukladat nejdulezitejsi informace do
souboru?");
                  System.out.println("*   1. Ano.");
                  System.out.println("*   2. Ne.");
                  System.out.println();
                  System.out.print("* Vase volba?: ");
                  klavesa = VstupData.ctiInt();
                  System.out.println();
                  System.out.println();
            // uvazujeme jen klavesy 1 a 2, jinak opakujeme zadani otazky
            } while ((klavesa > 2) || (klavesa < 1));

            // pokud byla zadana klavesa 1, zacneme logovat
            if (klavesa==1) log = new FileLog("log.txt");

            // zeptame se, jake pole mame vytvorit
            do {
                  System.out.println("* Jake pole chcete vytvorit?");
                  System.out.println("*   1. Pole s nahodnymi hodnotamy.");
                  System.out.println("*   2. Pole s rucne zadanymi hodnotamy.");
                  System.out.println("*   3. Ukoncit program.");
                  System.out.println();
                  System.out.print("* Vase volba?: ");
                  klavesa = VstupData.ctiInt();
                  System.out.println();
                  System.out.println();
            // uvazujeme jen klavesy 1 - 3, jinak opakujeme zadani otazky
            } while ((klavesa > 3) || (klavesa < 1));

            // pokud byla zadana klavesa 3, tak nebudeme pokracovat
            if (klavesa!=3) {
                  if (klavesa==1) {
                        /*
                         * pokud byla zmacknuta klavesa 1, tak vytvorime pometovy
                         * prostor pro pole a naplnime ho nahodnymi cisly + zapiseme
                         * nejaky text do souboru
                         */
                        if (log!=null) log.println("V programu bylo pracovano s
polem naplnenym nahodnymi cisly.");
                        System.out.print("* Velikost pole?: ");
                        pole = Tools.RandomArray(VstupData.ctiInt());
                        if (log!=null) log.println("Velikost pole byla:
"+pole.length);
                  } else {
                        /*
                         * pokud byla zadana tedy klavesa 2, tak budeme zadvat
hodnoty
                         * rucne, vytvorime pole a for cyklem se budeme ptat na
potrebne
                         * hodnoty + zapiseme nejaky text do souboru
                         */
                        if (log!=null) log.println("V programu bylo pracovano s
polem naplnenym rucne zadanymi hodnotamy.");
                        System.out.print("* Velikost pole?: ");
                        pole = new int[VstupData.ctiInt()];
                        if (log!=null) log.println("Velikost pole byla:
"+pole.length);
                        for(int i=0;i<pole.length;i++) {
                              System.out.print("* "+i+". prvek?: ");
                              pole[i] = VstupData.ctiInt();
                        }
```

```
                }
                System.out.println();
                System.out.println();

                if (log!=null) {
                      log.println();
                      log.println();
                }

                // zapis textu do souboru
                if (log!=null) log.println("Vypisuji pole, ktere bylo pred
tridenim:");

                // zeptame se, zda chceme vypsat zadane pole
                do {
                      System.out.print("* Chcete vypsat pole? (1=ANO, 0=NE):");
                      klavesa = VstupData.ctiInt();
                      System.out.println();
                      System.out.println();
                // uvazujeme jen klavesy 0 a 1, jinak opakujeme zadani otazky
                } while ((klavesa > 1) || (klavesa < 0));
                if (klavesa==1) {
                      //pokud je zmacknuta klavesa 1, vypiseme pole
                      System.out.println("---{ Vypis pole }----------------------
----------------");
                      for(int i=0;i<pole.length;i++) {
                            System.out.println(pole[i]+" ");
                            if (log!=null) log.println(pole[i]+" ");
                      }
                      System.out.println("----------------------------------------
----------------");
                      System.out.println();
                      System.out.println();
                } else {
                      // pokud je zapnut zapis do souboru, tak tam zapiseme pole
                      if (log!=null) for(int i=0;i<pole.length;i++)
log.print(pole[i]+" ");
                }

                if (log!=null) {
                      log.println();
                      log.println();
                }

                // zeptame se jakou tridici metodou se ma seradit pole
                do {
                      System.out.println("* Jakou metodou chcete setridit pole?");
                      System.out.println("*   1. Setridit InsertSortem vzestupne
[A-Z].");
                      System.out.println("*   2. Setridit InsertSortem sestupne
[Z-A].");
                      System.out.println("*   3. Setridit SelectSortem vzestupne
[A-Z].");
                      System.out.println("*   4. Setridit SelectSortem sestupne
[Z-A].");
                      System.out.println("*   5. Setridit BubbleSortem vzestupne
[A-Z].");
                      System.out.println("*   6. Setridit BubbleSortem sestupne
[Z-A].");
                      System.out.println();
                      System.out.print("* Vase volba?: ");
                      klavesa = VstupData.ctiInt();
```

```
                        System.out.println();
                        System.out.println();
                  // uvazujeme jen klavesy 1 - 6, jinak opakujeme zadani otazky
                  } while ((klavesa > 6) || (klavesa < 1));
                  // vypiseme na obrazovku text o tom, ze tridime, aby se uzivatel
nestrachoval
                  System.out.print("* Tridim... ");

                  // zapiseme text do souboru
                  if (log!=null) log.print("Vyse vypsane pole bylo setrideno
metodou");
                  //vybereme switchem, jakou metodou budeme tridit
                  switch(klavesa) {
                        case 1:
                              if (log!=null) log.println(" InsertSort
(vzestupne).");
                              sort.InsertSortASC(pole);
                              break;
                        case 2:
                              if (log!=null) log.println(" InsertSort (sestupne).");
                              sort.InsertSortDESC(pole);
                              break;
                        case 3:
                              if (log!=null) log.println(" SelectSort
(vzestupne).");
                              sort.SelectSortASC(pole);
                              break;
                        case 4:
                              if (log!=null) log.println(" SelectSort (sestupne).");
                              sort.SelectSortDESC(pole);
                              break;
                        case 5:
                              if (log!=null) log.println(" BubbleSort
(vzestupne).");
                              sort.BubbleSortASC(pole);
                              break;
                        case 6:
                              if (log!=null) log.println(" BubbleSort (sestupne).");
                              sort.BubbleSortDESC(pole);
                              break;
                  }
                  // taaaak a ted vypiseme na obrazovku text, ze je trideni do
konceno
                  System.out.println("HOTOVO!");
                  System.out.println();
                  System.out.println();
                  // zapiseme text do souboru
                  if (log!=null) log.println("Vypisuji setridene pole:");

                  // zeptame se uzivatele, zda chce vypsat pole
                  do {
                        System.out.print("* Chcete vypsat pole? (1=ANO, 0=NE):");
                        klavesa = VstupData.ctiInt();
                        System.out.println();
                        System.out.println();
                  // uvazujeme jen klavesy 0 a 1, jinak opakujeme zadani otazky
                  } while ((klavesa > 1) || (klavesa < 0));
                  if (klavesa==1) {
                        // pokud byla zmacknuta klavesa 1, vypiseme pole a to i do
souboru
                        System.out.println("---{ Vypis pole }-----------------------
----------------");
                        for(int i=0;i<pole.length;i++) {
```

```
                              System.out.println(pole[i]);
                              if (log!=null) log.println(pole[i]+" ");
                        }
                        System.out.println("----------------------------------------
----------------");
                        System.out.println();
                        System.out.println();
                  } else {
                        /*
                         * pokud to nebyla klavesa 1, tak to stejne zapiseme do
souboru,
                         * pokud tedy je zapnute logovani
                         */
                        if (log!=null) for(int i=0;i<pole.length;i++)
log.println(pole[i]+" ");
                  }

                  // taaaak a je cas na mensi statistiku, aby se nereklo.
                  // takze nejdriv na obrazovku
                  System.out.println("* Statistika:");
                  System.out.println("*   Pocet porovnani: "+sort.porovnani);
                  System.out.println("*   Pocet vymen (presunu): "+sort.prehozu);

                  if (log!=null) {
                        log.println();
                        log.println();
                  }

                  // taaak a ted tez i do souboru, pokud je zaple logovani
                  if (log!=null) {
                        log.println("Statistika:");
                        log.println("   Pocet porovnani: "+sort.porovnani);
                        log.println("   Pocet vymen (presunu): "+sort.prehozu);
                  }
            }

            // a pokud je tedy zaple logovani, tak uzavreme soubor
            if (log!=null) log.close();
            log = null;
      }
}
```

soubor Tabulka.java:

```
/**
 * Hlavni program na generovani tabulky pro graf.
 *
 * FILE: Tabulka.java
 * @author Martin Sloup (A04372), 13. krouzek, FAV - Informatika
 * @version 1.0, 12/04/04
 */
// naimportujeme potrebne balicky
import java.io.*;
import java.util.Random;

public class Tabulka {
      public static void main(String[] args) {
            /**
             * vytvorime objekt pro zapis z nasi suprtrupr tridy a hned priradime
             * novou instanci
             */
            FileLog bigtable = new FileLog("bigtable.csv");
            boolean is_ok = false;
```

```
            // objekt pro trideni, pritom hned vytvorime novou instanci
            Trideni sort = new Trideni();
            // objekt pro praci s nahodnyma cislama, mnohem komplexnejsi, nez
Math.random();
            Random nc = new Random();

            // nadeklarujeme potrebna pole...
            int[] origArr;
            int[] arr;
            int[] tmpArr;

            // zobrazime hlavicku programu
            Tools.hlavickaProgramu();

            // zeptame se na pocatecni a koncovou velikost pole
            System.out.print("Zadejte pocatecni velikost pole (napr. 1): ");
            int odVelikosti = VstupData.ctiInt();
            System.out.print("Zadejte koncovou velikost pole (napr 1000): ");
            int doVelikosti = VstupData.ctiInt();

            // zeptame se uzivatele co vsechno ma obsahovat vystupni soubnor
            int typZapisu = 0;
            do {
                  System.out.println("Co vsechno se ma obsahovat vystupni soubor?");
                  System.out.println("  1 ... pocet porovnani dvou prvku");
                  System.out.println("  2 ... pocet vymen (presunu) prvku");
                  System.out.println("  3 ... obe vise uvedene volby");
                  System.out.println();
                  System.out.print("Vase volba: ");
                  typZapisu = VstupData.ctiInt();
            } while (typZapisu < 1 || typZapisu > 3);

            //priradime do promene, ze je vse ok
            is_ok = true;

            /**
             * otestujem zda se nesnazi uzivatel oblbnout program, pokud jo,
vyhodime
             * hlasku a udelame vsechno pro to, aby se nepokracovalo dal
             */
            if (odVelikosti>doVelikosti) {
                  System.out.println("Koncova velikost pole nesmi byt mensi nez
pocatecni velikost pole!!!");
                  // nyni neni vse ok, takze priradime do promene false
                  is_ok = false;
            }
            if (odVelikosti<=0) {
                  System.out.println("Pocatecni velikost pole musi byt vetsi nez
cislo 0!!!");
                  // nyni neni vse ok, takze priradime do promene false
                  is_ok = false;
            }

            // pokud tedy je vse ok, tak muzeme pokracovat
            if (is_ok) {

                  // pridelime pametovy prostor pro pole
                  origArr = new int[1];

                  // tak zacneme pocitat hodnoty pro pole od odVelikosti do
doVelikosti
                  for(int i=odVelikosti;i<=doVelikosti;i++) {
                        /*
```

```
                        * vytvorime docasne pole, aby jsem mohli zvetsit velikost
pole
                        * a prigenerovat tam dalsi prvek
                        */
                       arr = new int[i];
                       // zkopirujeme pole
                       System.arraycopy(origArr,0,arr,0,origArr.length);

                       // prigenerujeme dalsi prvek
                       arr[i-1]=nc.nextInt();
                       // priradime nove pole do orig pole
                       origArr=arr;
                       arr = null;

                       // nyni pridelime pametovy prostor pro originalni pole,
ktere se bude pouzivat pro trideni
                       tmpArr = new int[i];

                       // do souboru pridame velikost souboru
                       bigtable.print(i+";");

                       // zkopirujeme z originalniho pole do tridiciho pole
                       System.arraycopy(origArr,0,tmpArr,0,origArr.length);
                       // nyni setridime pole
                       sort.InsertSortASC(tmpArr);
                       // zapiseme do souboru potrebne udaje pro vysledny graf /
tabulku
                       if (typZapisu!=2) bigtable.print(sort.porovnani+";");
                       if (typZapisu!=1) bigtable.print(sort.prehozu+";");

                       // to same provedem i pro ostatnich 5 tridicich metod
                       System.arraycopy(origArr,0,tmpArr,0,origArr.length);
                       sort.InsertSortDESC(tmpArr);
                       if (typZapisu!=2) bigtable.print(sort.porovnani+";");
                       if (typZapisu!=1) bigtable.print(sort.prehozu+";");

                       System.arraycopy(origArr,0,tmpArr,0,origArr.length);
                       sort.SelectSortASC(tmpArr);
                       if (typZapisu!=2) bigtable.print(sort.porovnani+";");
                       if (typZapisu!=1) bigtable.print(sort.prehozu+";");

                       System.arraycopy(origArr,0,tmpArr,0,origArr.length);
                       sort.SelectSortDESC(tmpArr);
                       if (typZapisu!=2) bigtable.print(sort.porovnani+";");
                       if (typZapisu!=1) bigtable.print(sort.prehozu+";");

                       System.arraycopy(origArr,0,tmpArr,0,origArr.length);
                       sort.BubbleSortASC(tmpArr);
                       if (typZapisu!=2) bigtable.print(sort.porovnani+";");
                       if (typZapisu!=1) bigtable.print(sort.prehozu+";");

                       System.arraycopy(origArr,0,tmpArr,0,origArr.length);
                       sort.BubbleSortDESC(tmpArr);
                       if (typZapisu!=2) bigtable.print(sort.porovnani+";");
                       if (typZapisu!=1) bigtable.print(sort.prehozu+";");

                       // ukoncime radek
                       bigtable.println();
                   }
                   // uzavreme soubor
                   bigtable.close();
               }
           }
```

```
}
```

soubor Tools.java:

```
/**
 * Trida s pomocnymi statickymi metodamy, ktere se jinam nevesly
 *
 * FILE: Tools.java
 * @author Martin Sloup (A04372), 13. krouzek, FAV - Informatika
 * @version 1.0, 12/04/04
 */
// naimportujeme potrebne balicky
import java.io.*;
import java.util.Random; // pro tridu Random

public class Tools {
	/**
	 * Metoda na generovani pole s naplnenymi nahodnymi hodnotami v rozsahu int.
	 */
	public static int[] RandomArray(int items) {
		// inicializace pole s poctem prvku [items]
		int[] arr = new int[items];
		// inicializace objectu Random (pro nas komplexnejsi nez Math.random())
		Random rnd = new Random();

		// pro jedeme pro kazdy prvek pole
		for (int i =0;i<arr.length;i++) {
			/* do kazdeho prvku pole vygenerujeme nahodnou hodnotu v rozsahu
			 * INTEGERu. Proto je trida Random komplexnejsi, protoze nemusime
			 * nasobit Math.random() konstantou urcujici rozsah INTEGERu
			 */
			arr[i] = rnd.nextInt();
		}
		// snazime se psat neprasacky, proto zrusime objekt
		rnd = null;
		return arr;
	}
	/**
	 * Metoda na vypsani hlavicky: autor, ID studenta, krouzek, e-mail.
	 */
	public static void hlavickaProgramu() {

	System.out.println("************************************************************")
;
		System.out.println(" Semestralni prace z predmetu Pocitace a
programovani 1 ");
		System.out.println(" Autor: Martin Sloup (A04372), 13. krouzek,
");
		System.out.println("        msloup@students.zu.cz
");

	System.out.println("************************************************************")
;
		System.out.println();
	}
}
```

soubor Trideni.java:

```
/**
 * Trida na trideni posloupnosti cisel v poli urcitymi metodamy.
 *
 * FILE: Trideni.java
 * @author Martin Sloup (A04372), 13. krouzek, FAV - Informatika
 * @version 1.0, 12/04/04
```

```
 */

public class Trideni {
	// promene pro uchovani poctu prehozu (vymen) a porovnani
	public int porovnani = 0;
	public int prehozu = 0;
	/**
	 * Trideni primym vkladanim (vzestupne)
	 */
	public void InsertSortASC(int[] pole) {
		int x;
		int j = 0;
		this.porovnani=0;
		this.prehozu=0;
		// projedem prvek po prvku
		for(int i=1;i<pole.length;i++) {
			// zapamatujeme prvek, ktery mame zaradit na spravne misto
			x = pole[i];
			j=i-1;
			// posuvame prvek zpatky, pokud je kam posouvat
			for (j=i;j>0;j--) {
				this.porovnani++;
				/* pokud je predchozi prvek mensi, tak jsme ho uz zaradili
				 * spravne, takze muzeme vyzkocit z posouvani a provest to
same
				 * u dalsiho prvku
				 */
				if(x>pole[j-1]) break;
				// prohodime prvky (bereme to jako prehoz)
				pole[j]=pole[j-1];
				this.prehozu++;
			}
			// nyni priradime hodnotu na spravne misto (bereme to jako prehoz)
			pole[j] = x;
		}
	}

	/**
	 * Trideni primym vkladanim (sestupne)
	 */
	public void InsertSortDESC(int[] pole) {
		int x;
		int j=0;
		this.porovnani=0;
		this.prehozu=0;
		// projedem prvek po prvku
		for(int i=1;i<pole.length;i++) {
			// zapamatujeme prvek, ktery mame zaradit na spravne misto
			x = pole[i];
			j=i-1;
			// posuvame prvek zpatky, pokud je kam posouvat
			for (j=i;j>0;j--) {
				this.porovnani++;
				/* pokud je predchozi prvek vetsi, tak jsme ho uz zaradili
				 * spravne, takze muzeme vyzkocit z posouvani a provest to
same
				 * u dalsiho prvku
				 */
				if(x<pole[j-1]) break;
				// prohodime prvky
				pole[j]=pole[j-1];
				this.prehozu++;
			}
```

```
			// nyni priradime hodnotu na spravne misto
			pole[j] = x;
		}
	}

	/**
	 * Trideni primym vyberem (vzestupne)
	 */
	public void SelectSortASC(int[] pole) {
		int x;
		int nejmensi;
		this.porovnani=0;
		this.prehozu=0;

		// prvky budeme davat na i-te pozice
		for (int i=0;i<pole.length;i++) {
			// hledame nejmensi prvek
			nejmensi = i;

			for (int j=i+1;j<pole.length;j++) {
				if (pole[j]<pole[nejmensi]) nejmensi = j;
				this.porovnani++;
			}

			// zamenime prvky
			x = pole[i];
			pole[i]=pole[nejmensi];
			pole[nejmensi] = x;
			this.prehozu++;
		}
	}

	/**
	 * Trideni primym vyberem (sestupne)
	 */
	public void SelectSortDESC(int[] pole) {
		int x;
		int nejvetsi;
		this.porovnani=0;
		this.prehozu=0;

		// prvky budeme davat na i-te pozice
		for (int i=0;i<pole.length;i++) {
			// hledame nejvetsi prvek
			nejvetsi = i;

			for (int j=i+1;j<pole.length;j++) {
				if (pole[j]<pole[nejvetsi]) nejvetsi = j;
				this.porovnani++;
			}

			// zamenime prvky
			x = pole[i];
			pole[i]=pole[nejvetsi];
			pole[nejvetsi] = x;
			this.prehozu++;
		}
	}

	/**
	 * Trideni primou vymenou (vzestupne)
	 */
	public void BubbleSortASC(int[] pole) {
```

```
            int x;
            this.porovnani=0;
            this.prehozu=0;

            for(int i=1;i<pole.length;i++) {
                  for(int j=pole.length-1;j>=i;j--) {
                        // je prvek zarazen spravne?
                        this.porovnani++;
                        if (pole[j-1]>pole[j]) {
                              // zamenime prvky
                              x = pole[j-1];
                              pole[j-1] = pole[j];
                              pole[j] = x;
                              this.prehozu++;
                        }
                  }
            }
      }

      /**
       * Trideni primou vymenou (sestupne)
       */
      public void BubbleSortDESC(int[] pole) {
            int x;
            this.porovnani=0;
            this.prehozu=0;

            for(int i=pole.length-2;i>0;i--) {
                  for(int j=i;j<pole.length;j++) {
                        this.porovnani++;
                        // je prvek zarazen spravne?
                        if (pole[j-1]<pole[j]) {
                              // zamenime prvky
                              x = pole[j-1];
                              pole[j-1] = pole[j];
                              pole[j] = x;
                              this.prehozu++;
                        }
                  }
            }
      }

}
```

soubor VstupData.java:

```
/**
 * Trida na nacitani hodnot s klavesnice.
 *
 * FILE: VstupData.java
 * @author Neznamy
 * @version 1.0, 01/01/00
 */

public class VstupData {

public static boolean ctiBoolean() {
   byte[] pole = new byte[200];
   try {
      System.in.read(pole);
      System.in.skip(System.in.available());
      return Boolean.valueOf(new String(pole).trim()).booleanValue();
   }
   catch (Exception e) {
```

```
      System.out.println("Chyba pri nacitani !");
      return false;
   }

}


public static byte ctiByte() {
   byte[] pole = new byte[200];
   try {
      System.in.read(pole);
      System.in.skip(System.in.available());
      return Byte.parseByte(new String(pole).trim());
   }
   catch (Exception e) {
      System.out.println("Chybne nactene cislo !");
      return 0;
   }

}


public static short ctiShort() {
   byte[] pole = new byte[200];
   try {
      System.in.read(pole);
      System.in.skip(System.in.available());
      return Short.parseShort(new String(pole).trim());
   }
   catch (Exception e) {
      System.out.println("Chybne nactene cislo !");
      return 0;
   }

}


public static int ctiInt() {
   byte[] pole = new byte[200];
   try {
      System.in.read(pole);
      System.in.skip(System.in.available());
      return Integer.parseInt(new String(pole).trim());
   }
   catch (Exception e) {
      System.out.println("Chybne nactene cislo !");
   return 0;
 }
}


public static long ctiLong() {
   byte[] pole = new byte[200];
   try {
      System.in.read(pole);
      System.in.skip(System.in.available());
      return Long.parseLong(new String(pole).trim());
   }
   catch (Exception e) {
      System.out.println("Chybne nactene cislo !");
   return 0;
 }
}
```

```
public static float ctiFloat() {
   byte[] pole = new byte[200];
   try {
      System.in.read(pole);
      System.in.skip(System.in.available());
      return Float.parseFloat(new String(pole).trim());
   }
   catch (Exception e) {
      System.out.println("Chybne nactene cislo !");
      return 0.0F;
   }

}


public static double ctiDouble() {
   byte[] pole = new byte[200];
   try {
      System.in.read(pole);
      System.in.skip(System.in.available());
      return Double.parseDouble(new String(pole).trim());
   }
   catch (Exception e) {
      System.out.println("Chybne nactene cislo !");
      return 0.0;
   }

}

public static char ctiChar() {
  try {
      char c = (char) System.in.read();
      System.in.skip(System.in.available());
      return(c);
  }
  catch (Exception e) {
      System.out.println("Chyba pri nacitani znaku !");
      return '\u0000';
  }
}



public static String ctiString() {
   byte[] pole = new byte[2000];
   try {
      System.in.read(pole);
      System.in.skip(System.in.available());
      return new String(pole).trim();
   }
   catch (Exception e) {
      System.out.println("Chyba pri nacitani retezce");
      return null;
   }

}

} //end class
```

Ukázkový výsledek:

Výstupní hodnoty z programu (vstupní hodnoty pro tabulku a graf):

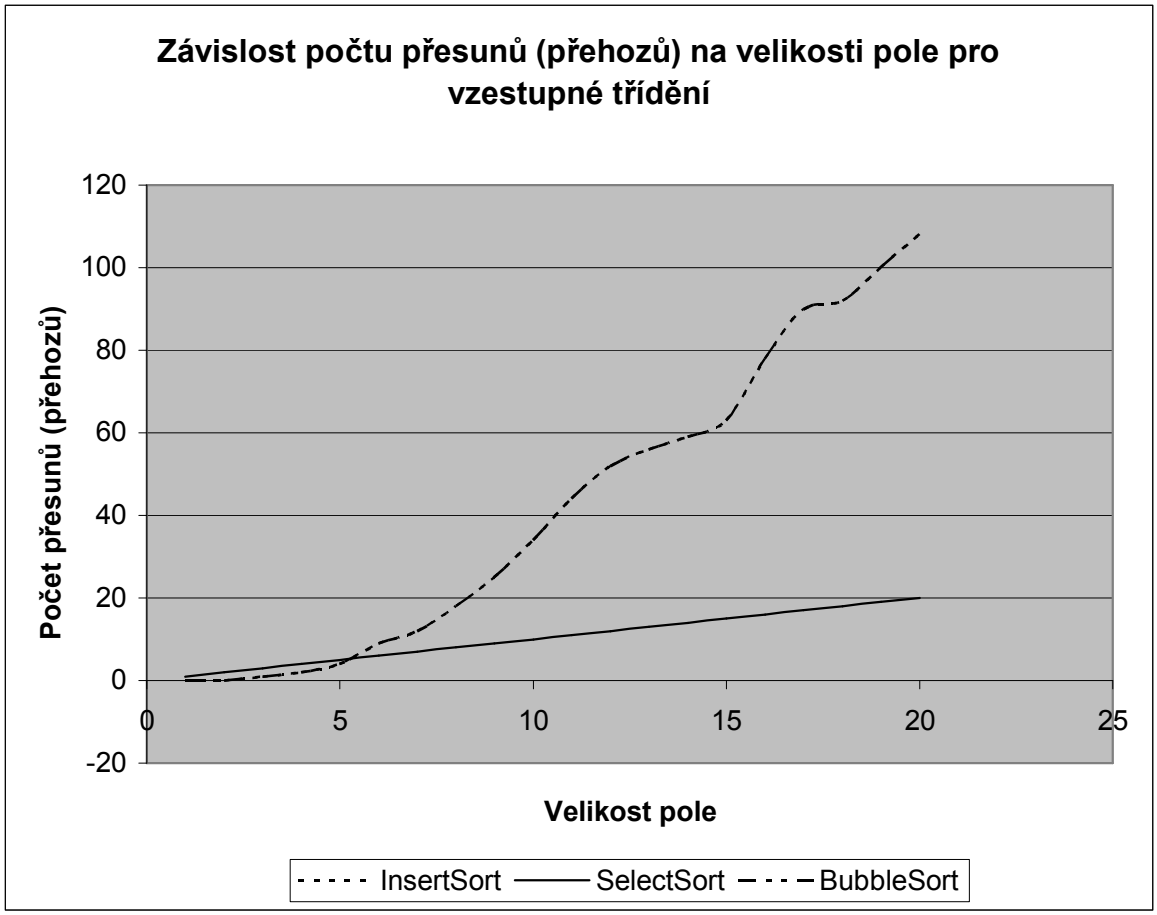
Jako vstupní hodnoty pro vytvoření tabulky a následné vygenerování grafu jsem použil výstupní soubor pomocného programu s použitím volby 2 při otázce: Co všechno má obsahovat výstupní soubor?“ Výstupní soubor tedy obsahuje počet počet výměn (přehozů) jak vzestupně, tak i sestupně pro všechny tři třídící metody. Tato potřebná čísla byla vypočítána postupně pro pole o počtu prvků 1 až 20 s tím, že vždy následující pole obsahovalo samé hodnoty jako předchozí velikost pole a vždy jeden prvek s vygenerovaným číslem. Toto pole bylo prohnáno všemi třemi třídícími metodami a to vzestupně a i sestupně. Stejnými hodnotami jsem se snažil výpočet provést co možná nejpřesněji.

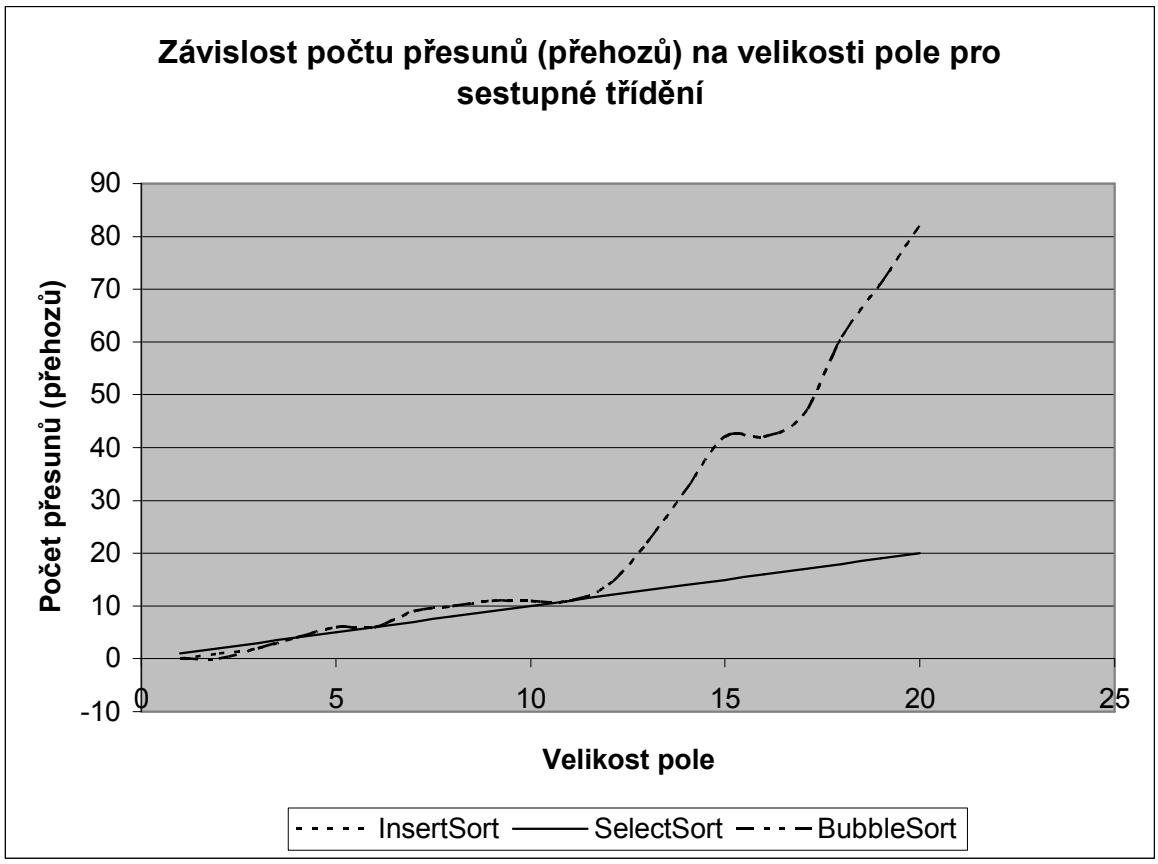
Tabulka počtu porovnání:

Závislost počtu přesunů (přehozů) na velikosti pole

Velikost pole	Počet výměn (přesunů) InsertSortu vzestupně	Počet výměn (přesunů) InsertSortu sestupně	Počet výměn (přesunů) SelectSortu vzestupně	Počet výměn (přesunů) SelectSortu sestupně	Počet výměn (přesunů) BubbleSortu vzestupně	Počet výměn (přesunů) BubbleSortu sestupně
1	0	0	1	1	0	0
2	0	1	2	2	0	0
3	1	2	3	3	1	2
4	2	4	4	4	2	4
5	4	6	5	5	4	6
6	9	6	6	6	9	6
7	12	9	7	7	12	9
8	18	10	8	8	18	10
9	25	11	9	9	25	11
10	34	11	10	10	34	11
11	44	11	11	11	44	11
12	52	14	12	12	52	14
13	56	22	13	13	56	22
14	59	32	14	14	59	32
15	63	42	15	15	63	42
16	78	42	16	16	78	42
17	90	46	17	17	90	46
18	92	61	18	18	92	61
19	100	71	19	19	100	71
20	108	82	20	20	108	82

Graf z tabulky počtu porovnání:





Z grafu lze vyčíst že, nejlepší v počtu přesunů (přehozů) pro náhodně generované pole je, podle mého názoru, třídící algoritmus SelectSort. Zbylí dva třídící algoritmy se chovaly zcela stejně.

Poznámka ke grafu a k hodnotám přesunů (přehozů) u InsertSortu:

Mezi mými spolužáky (studenty) se vedla vášnivá diskuze o tom, co se vše bere u InsertSortu jako přesun (přehoz). Proto je možné, že výsledky u InsertSortu nejsou zcela správné.

Použitá literatura:

Přednášky z předmětu Počítače a programování 1 – Doc. Dr. Ing. Jana Klečková
Algoritmy a štruktúry údajov – Nikolaus Wirth
Učebnice jazyka JAVA – Pavel Herout
Dokumentace jazyka JAVA
Ústní komunikace se studenty:
Jan Syrovátka (A04375), Martin Žibrický (A04396), Karel Langmaier (A04326), Milan Pixa (A04049), Miroslav Panský (A04343), Jiří Hradský (A04291), Jiří Kučera (A04321)

Přílohy:

Zdrojové kódy (adresář ./src/):
FileLog.java, Main.java, Tabulka.java, Tools.java, Trideni.java, VstupData.java
Samotný program (adresář ./):
TestTrideni.jar, TestTrideni.bat, GenTable.jar, GenTable.bat
Tabulka a graf (adresář ./tabulka/):
table.xls
Výstupní soubor z pomocného programu použitý na tabulku a graf (adresář ./tabulka/):
table.csv
Tento dokument v digitální podobě (adresář ./dokumentace/):
dokumentace.doc, dokumentace.pdf
Ostatní soubory (adresář ./)
Trideni.zip (obsahuje cely projekt JCreatoru + komprimaci do JAR archívů + ostatní soubory)