```java
package skolni_threads2b;

import java.io.BufferedReader;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;
import java.util.Iterator;
import java.util.TreeSet;

/* z tehle tridy udelame vlakno
 * farmar zpracovava delniky
 */


public class Farmer extends Thread{ // potrebujeme definovat run()


	private TreeSet<WordRecord> wordRecords;

	private BufferedReader in;

	private int jokeID;

	private static final int WORKERS = 3;

	private Worker worker[] = new Worker[WORKERS];

	Farmer (String file) {

		System.out.println("Farmar - zacinam.");

		// nacteni vstupniho souboru
		System.out.println("Farmar - oteviram soubor...");



		try {
			FileReader fr = new FileReader(file);
			in = new BufferedReader(fr);


		} catch (FileNotFoundException e) {
			// TODO Auto-generated catch block
			e.printStackTrace();
		}


		// vytvoreni globalniho seznamu vysledku
		wordRecords = new TreeSet<WordRecord>(new WRComparer());
	}



	public void run() {

		System.out.println("Farmar: vytvarim a spoustim delniky.\n");
		// piseme uplne nove telo, protoze jeho praci vykona worker
		int i;
```

```
		for(i = 0; i < WORKERS; i++){

			worker[i] = new Worker("Makac " + i, this, (i+1)*100);


			worker[i].start();
		}
		System.out.println("Farmar: cekam na ukonceni delniku.\n");



		// ceka se na workery, az domakaj
		for(i = 0; i < WORKERS; i++){
			try {


				worker[i].join();
			} catch (InterruptedException e) {
				// TODO Auto-generated catch block
				e.printStackTrace();
			}
		}

		// farmarova prace - uzavreni souboru
		try {
			in.close();
		} catch (IOException e) {
			// TODO Auto-generated catch block
			e.printStackTrace();
		}
		System.out.println("Farmar: tisknu vysledek.\n");

		printResult(); // vypsani vysledku
	}



	// prideleni prace workerovi
	synchronized public String getJoke(Worker w) {

		System.out.println(w.getWorkerName() + ": Zadam vtip.");

		String jokeText = "";

		String line;

		try {
			while ((line = in.readLine()) != null) {

				if (line.trim().equals("%")) {

					System.out.println("Vtip nacten ("+jokeID + ").");

					jokeID++;

					return jokeText.trim();
				}

				jokeText += line + "\n";
			}
		} catch (IOException e) {
			// TODO Auto-generated catch block
```

```
                e.printStackTrace();
            }
            System.out.println("Neni co cist.");

            return "$$skonci$$";

        }


        // ukladani vysledku z jednoho vtipu do globalniho seznamu
        synchronized public void reportResult(Worker w) {

            System.out.println(w.getWorkerName() + ": Pridavam vysledky.");

            Iterator<WordRecord> it = w.results.iterator();

            while (it.hasNext()) {

                WordRecord listItem = it.next();

                WordRecord wr = getWordRecord(listItem.word);

                if (wr == null)

                    wordRecords.add(listItem);
                else

                    wr.frequency += listItem.frequency;
            }
        }


        // ziskani zaznamu slova z globalniho seznamu
        private WordRecord getWordRecord (String item) {

            Iterator<WordRecord> it = wordRecords.iterator();

            while (it.hasNext()) {

                WordRecord listItem = it.next();

                if (listItem.word.equals(item))

                    return listItem;
            }
            return null;
        }

        // tisk vysledneho seznamu
        public void printResult() {

            Iterator<WordRecord> it = wordRecords.iterator();

            System.out.println("Vysledek:");
            System.out.println("=========");

            while (it.hasNext()) {

                WordRecord listItem = it.next();
                System.out.println(listItem.word +" - " + listItem.frequency);
            }
        }
}
```

```java
public class FarmerWorkers {

	private final static String INPUT_FILE_NAME = "vtipy.txt";

	public static void main(String[] args) throws InterruptedException {
		// TODO Auto-generated method stub

		System.out.println("Main - spoustim farmare.");
		Farmer farmer = new Farmer(INPUT_FILE_NAME);

		farmer.start();

		// cekame na farmare :D
		farmer.join();

		System.out.println("Farmar - tisknu vysledek.");
		farmer.printResult();

		System.out.println("Main - konci.");
	}
}
```

```java
import java.util.Iterator;

import java.util.TreeSet;


public class Worker extends Thread {


// kam se budou ukladat slova s cetnostma
	public TreeSet<WordRecord> results;


	private String name;

	private Farmer farmer; // pro koho delnik pracuje

	private int jobCount;

	private String joke;

	private int speed;


	Worker(String name, Farmer farmer, int speed){
		this.name = name;
		this.farmer = farmer;
		this.speed = speed;

		results = new TreeSet<WordRecord> (new WRComparer());
	}

	public String getWorkerName(){
		return name;
	}
```

```
public void run() {
		int i;
		int jobCount;
		String joke;
		//TreeSet<WordRecord> results;
		System.out.println(name + ": Zacinam makat.");
		jobCount = 0;
		while ((joke = farmer.getJoke(this)).equals("$$skonci$$") == false)
{
			System.out.println("Dostal jsem praci.");
			results = new TreeSet<WordRecord>(new WRComparer());
			String[] words = joke.split("[ \".'-',;!?()*+&:=@#
                                        %^`□\'_$/|'['']'{}\t\n]");
			long prevTime = System.currentTimeMillis();
			for (i = 0; i < words.length; i++) {
				// aktivni cekani
				long currTime = System.currentTimeMillis();
				while(currTime < prevTime + speed){
					currTime = System.currentTimeMillis();
				}
				// prevedeni slova na mala pismena
				words[i] = words[i].toLowerCase();

				// zpracovani slova
				// nechci globalni vysledek, ale lokalni => upravuju
				WordRecord wr = getWordRecord(words[i]);
				if (wr == null) { // nove slovo ve vtipu
					wr = new WordRecord();
					wr.word = words[i];
					wr.frequency = 1;
					results.add(wr);
				}else { // slovo se jiz ve vtipu vyskytlo
					wr.frequency++;
				}
				System.out.println(name + ": Zpracovano slovo: " + " / "
                                    + wr.word + " /");
```

```
                        prevTime = currTime;
                }

                // ulozeni vysledku do globalniho seznamu u farmera
                farmer.reportResult(this);

                jobCount++;
            }
            System.out.println(name + ": Koncim, zpracoval jsem " + jobCount + "
                               vtipu.");
        }

        //ziskani zaznamu slova z lokalniho seznamu
        private WordRecord getWordRecord (String item) {

            Iterator<WordRecord> it = results.iterator();

            while (it.hasNext()) {

                WordRecord listItem = it.next();

                if (listItem.word.equals(item))

                    return listItem;
            }
            return null;
        }
}
```

```
public class WordRecord {
        String word;
        int frequency;
}
```

```
import java.util.Comparator;

public class WRComparer implements Comparator<WordRecord> {

        public int compare(WordRecord arg0, WordRecord arg1) {

            if (arg0.frequency < arg1.frequency)

                return 1;
            else

                return -1;

        }
}
```