BSZ WIESAU Informatik Campus	2. Schulaufgabe – AemP Nachtermin		
	Name:	Klasse: BFI12A	
	Datum: 29.11.2023	/ 29P.	

Gesamtsituation:
In der Cryptohochburg Wiesau wurde ein USB Stick gefunden, auf welchem Bitcoin-Adressen zusammen mit wichtigen Parametern und Klarnamen der Besitzer gespeichert sind. Sie versuchen so schnell wie möglich diese Daten auszuwerten um sich mit dem reichsten Bewohner anzufreunden.

Aufgabe 1: Dateieingabe

Hinweis: Verwenden Sie die Klasse BufferedReader für die Dateieingabe!
Schreiben Sie ein Java-Programm, das eine Datei mit angegebenen Namen einliest. Die Datei enthält alle relevanten Informationen mit Strichpunkten getrennt. Jede Zeile der Datei bezieht sich dabei auf ein Konto.
Erstellen Sie eine Methode namens *leseDatei(),* die die Datei einliest und zusammenhängende Informationen in einem Objekt der Klasse Bitcoinwallet (selbst zu erstellen) in einer geeigneten Datenstruktur speichert. Die Signatur der Methode lautet:

```
public static ArrayList<Bitcoinwallet> leseDatei(String dateiName)
                                                        throws Exception
```

Tipp: `DateTimeFormatter.ofPattern(????)`

Aufgabe 2: Dateiausgabe

Hinweis: Nutzen Sie die Klasse BufferedWriter für die Dateiausgabe!
Schreiben Sie eine weitere Methode namens *schreibeDatei(),* die eine ArrayList vom Typ Bitcoinwallet sowie einen Dateinamen als Parameter nimmt. Die Methode soll nur Bitcoin-Wale (Accounts mit 1000-5000 Bitcoin Vermögen) in einer neuen Datei speichern. Diese Liste soll in alphabetischer Reihenfolge nach Namen sortiert werden (Tipp: Lambda
Die Signatur der Methode lautet:

```
public static void schreibeDatei(String dateiName,
                                 ArrayList<Bitcoinwallet> wallets) throws Exception
```

Aufgabe 3: Durchschnittsnote

Implementieren Sie eine Methode namens *sucheMax ()*, die das größte Wallet mit allen Parametern zurückgiebt. Die Signatur der Methode lautet:

```
public static Bitcoinwallet sucheMax (ArrayList<Bitcoinwallet> wallets)
```

Aufgabe 4: Hauptprogramm

Schreiben Sie ein Hauptprogramm, das die oben genannten Methoden aufruft. Lesen Sie die Datei "Leak.csv" mit der Methode *leseDatei()* ein und speichern Sie die zurückgegebene ArrayList in einer Variable.
Lassen Sie die Datei "Whales.csv" erstellen. Geben Sie außerdem die reichste Person der Liste mit der Methode sucheMax() auf der Konsole aus.
Achtung: es muss anfangs überprüft werden, ob die Datei Leak.csv existiert und lesbar ist.

Geschafft!

MUSTERLÖSUNG:

```
//30 P insgesamt
public class Anwendung {

	// 5P insgesamt
	public static void main(String[] args) {
		try {
			// 2P
			File f = new File("src/Leak.csv");
			if (f.exists() && f.canRead()) {
				// 1P pro Aufruf
				ArrayList<Bitcoinwallet> liste = leseDatei("src/Leak.csv");
				schreibeDatei("src/Whales.csv", liste);
				System.out.println(sucheMax(liste));
			}
		} catch (Exception e) {
			e.printStackTrace();
		}
	}

	// 11P
	public static ArrayList<Bitcoinwallet> leseDatei(String dateiName) throws
Exception {
		// 2P
		BufferedReader rdr = new BufferedReader(new FileReader(dateiName));
		// 1P
		ArrayList<Bitcoinwallet> liste = new ArrayList<>();
		// 1P
		DateTimeFormatter formatter = DateTimeFormatter.ofPattern("dd.MM.yyyy");
		try {
			// 1P
			String line = rdr.readLine();
			line = rdr.readLine();
			// 1P
			while (line != null) {
				// 1P
				String parts[] = line.split(";");
				// 3P
				liste.add(new Bitcoinwallet(parts[0], parts[1],
Integer.parseInt(parts[2]),
					LocalDate.parse(parts[3], formatter),
Integer.parseInt(parts[4])));
				line = rdr.readLine();
			}
		} catch (Exception ex) {
			System.out.println(ex.getMessage());
		} finally {
			// 1P
			rdr.close();
		}
		return liste;
	}
```

```
	// 9P insgesamt
	public static void schreibeDatei(String dateiName, ArrayList<Bitcoinwallet>
wallets) throws Exception {
		// 4P
		List<Bitcoinwallet> sortiert = wallets.stream().filter(w -> w.getBalance()
>= 1000 && w.getBalance() <= 5000)
				.sorted((w1, w2) ->
w1.getName().compareTo(w2.getName())).collect(Collectors.toList());
		// 2P
		BufferedWriter wrt = new BufferedWriter(new FileWriter(dateiName));
		// 2P
		for (Bitcoinwallet item : sortiert) {
			wrt.write(item.toString());
		}
		// 1P
		wrt.close();

	}

	// 3P
	public static Bitcoinwallet sucheMax(ArrayList<Bitcoinwallet> liste) {

		Bitcoinwallet max =
liste.stream().max(Comparator.comparing(Bitcoinwallet::getBalance))
				.orElseThrow(NoSuchElementException::new);
		return max;
	}

}
```

Klasse Bitcoinwallet: 2 Punkte