第十回　演習問題（ファイル入出力）

諸注意

- 「FileUtil.java」，「Student.java」の２ファイルを Web から提出する．
- 指定した処理が正常に動くようであれば，適宜独自メソッド等を実装してくれて構わない．
- コピペ発覚時は見せた側も見せてもらった側も両方０点とする．
- 必ずコンパイルエラーのない状態で提出すること（自動採点したいのでコンパイルエラーがあると，全て０点になってしまう）．
- 課題の途中で提出することになった場合，コンパイルエラーさえ出なければ，課題の途中の状態で提出してくれて構わない．一部のメソッドだけが実現できていない場合，コンパイルエラー出ないならばそのままの状態で提出してくれてよい．
- 主にコンソール出力で評価しているため，デバッグに用いたようなコンソール出力が残っていないように気をつけること．基本的にコンソール出力を指定しない限りは，課題内でコンソール出力はないものとする．
- Package は使わないこと（デフォルトパッケージで実装する）．Package で実装すると，自動採点がうまくいきません．

課題 1

<table>
<tr><td rowspan="6">1</td><td>問題設定</td><td colspan="3">引数で与えられたパスのファイルを読み込んで，ファイル内に記述された行数を返すメソッドを実装せよ．テスト例は「workspace¥本日のプロジェクトディレクトリ」に，Web から DL した「kadai1.txt」を格納しておくことで実施できる．</td></tr>
<tr><td>クラス名</td><td>FileUtil</td><td>メソッド名</td><td>getLineNum()</td></tr>
<tr><td>引数</td><td>String path</td><td>戻り値</td><td>int 型</td></tr>
<tr><td>制限</td><td colspan="3">● BufferedReader を用いて読み込むこと．
（FileReader を経由することは許容する）
● try-cath で close()を適切に呼び出すこと．
（try-with-resource 文は使用しない）
● ファイルの存在確認等は if 文で実装しなくてよい．
（StackTrace を出力するだけで良い．）</td></tr>
<tr><td>諸注意</td><td colspan="3">● static メソッドとすること．
● 全クラス，全パッケージからアクセスできるようにすること．</td></tr>
<tr><td>テスト例</td><td colspan="3">（Main.javaのmainメソッドに以下をコピペする）
<code>int num = FileUtil.getLineNum("kadai1.txt");
System.out.println(num);</code></td></tr>
<tr><td>テスト
出力例</td><td colspan="3">4765</td></tr>
</table>

課題１：解答例（FileUtil.java）

```
public static int getLineNum(String path){
	// try-catchを使わなければならないという司令なので
	// finallyでclose()するためにtry-catchの外で宣言
	FileReader fr = null;
	BufferedReader br = null;
	int res = 0;  // 行数カウント用の変数
	try{
		fr = new FileReader(path);
		br = new BufferedReader(fr);
		String str;
		// strにbr.readLine()で1行読み込み,
		// null出ない間ループする（全行読める）.
		while((str = br.readLine()) != null){
			res++;
		}
	}catch(IOException e){
		e.printStackTrace();
	}finally{
		// finallyでclose()しないと，例外落ちした
		// 際にclose()されずメモリリークにつながる.
		if(br != null){
			try {
				br.close();
			} catch (IOException e) {
				e.printStackTrace();
			}
		}
	}
	return res;
}
```

課題２

<table>
<tr><td rowspan="6">2</td><td>問題設定</td><td colspan="3">九九の表を「kuku.txt」に出力するメソッドを実装せよ．ただし，行（縦方向の長さ）は可変長とする（詳細は後述する）．kuku.txt，kuku2.txt は「workspace¥本日のプロジェクトディレクトリ」に生成すれば良いものとする．</td></tr>
<tr><td>クラス名</td><td>FileUtil</td><td>メソッド名</td><td>kuku()</td></tr>
<tr><td>引数</td><td>int row</td><td>戻り値</td><td>なし</td></tr>
<tr><td>処理</td><td colspan="3">以下に，引数 row=3 の場合の出力例を示す．
1,2,3,4,5,6,7,8,9,
2,4,6,8,10,12,14,16,18,
3,6,9,12,15,18,21,24,27,
[EOF]
上記のように，row*9 の表をカンマ区切りで出力する．すなわち，row の段までの九九の表を出力する．なお縦の行数は９を超えることもありうる．</td></tr>
<tr><td>制限</td><td colspan="3">● 区切り文字はカンマとし半角スペース等は入れない．
● 改行コードは OS に依存せず適切なものを挿入せよ．
● try-with-resources 文を用い close() を省略せよ．
● 事前にファイル（kuku.txt）の存在確認を行い，存在する場合は「kuku2.txt」に出力せよ（kuku2.txt）も存在するケースは想定しなくて良いものとする．</td></tr>
<tr><td>諸注意</td><td colspan="3">● static メソッドとすること．
● 全クラス，全パッケージからアクセスできるようにすること．</td></tr>
<tr><td>テスト例</td><td colspan="3">上述の通りなので割愛する．
出力されたファイルをメモ帳等で確認すること．</td></tr>
</table>

課題２：解答例（FileUtil.java）

```
public static void kuku(int row){
	// 作業ディレクトリに出力で良いので相対パス表記をする.
	File file = new File("kuku.txt");
	// kuku.txtがあるばあい, kuku2.txtを対象とする.
	if(file.exists()) file = new File("kuku2.txt");

	// try-with-resources文でBufferedWriterを生成する.
	try(FileWriter fw = new FileWriter(file);
		BufferedWriter bw = new BufferedWriter(fw)){
		// 九九を出力するfor文
		for(int i=1; i<=row; i++){
			for(int j=1; j<=9; j++){
				bw.write(i*j + ",");
			}
			bw.newLine();
		}
	}catch(IOException e){
		e.printStackTrace();
	}
	// finallyでclose()しなくても勝手にclose()される.
	//  そのタイミングでBufferedWriterからファイルに書き込まれる.
}
```

課題３

3-1	問題設定	学生を管理するクラスを開発し，シリアライズしてファイルに出力できるようにせよ．
	クラス名	Student
	フィールド	全てのフィールドは private とする． name: 学生の名前(文字列型) id: 学生を一意に識別するための ID（数値型） birth: 学生の生年月日(Date 型)
	static フィールド	全ての static フィールドは private とする． nextId: 次に発番されるべき ID（数値型） ＞初期値 1 で，使用される毎にインクリメントされる．
	コンストラクタ	全てのフィールドを以下の引数で初期化する． name: 学生の名前 birth: 学生の生年月日（int 型：20180101 の書式） 1．name はそのままフィールドを初期化する． 2．id は static フィールド nextId の現在値を使用し，使用時に次の人のためにインクリメントしておく． 3．birth は生年月日の int 値なので，然るべき変換を行い Date 型としてフィールド birth を初期化する．なお，時刻は何時何分何秒でも良い．
	メソッド	String toString(): インスタンスの概要を返すメソッド 引数　：なし 戻り値：次の書式でインスタンスの概要を文字列で返す 長谷川達人(ID:1)_1988/12/02 （**アンダーバー_の部分は半角スペースとすること**） int getId(): Id を取得するための getter メソッド 引数　：なし 戻り値：id フィールド static void setNextId(): nextId を設定する． 引数　：設定する ID（数値型） 戻り値：なし 処理　：static フィールドの nextId を引数で更新する．
	諸注意	シリアライズしてファイル出力できるように，適切な記述を加えておくこと．

	テスト例	（Main.javaのmainメソッドに以下をコピペする）
		`Student s1 = new Student("長谷川達人", 19881202);`
		`System.out.println(s1.toString());`

	テスト結果例	長谷川達人(ID:1) 1988/12/02

課題 3-1：解答例（Student.java）

```
import java.io.Serializable;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.Date;

// implements Serializableは課題3-2用
public class Student implements Serializable{
	private static int nextId = 1;

	private String name = "";
	private int id = 0;
	private Date birth = null;

	public Student(String name, int birth){
		this.name = name;
		// idはnextIdから付番し，次の人のためにnextIdを更新する．
		this.id = Student.nextId;
		nextId ++;

		// Calendarを用いたパターンでint birthをDate birthに
		Calendar cal = Calendar.getInstance();
		cal.setTimeInMillis(0);  // 00:00:00にするため
		// 年月日に分割してCalendarにセットする．
		int year = (int)(birth/10000.0d);
		birth -= year*10000;
		int month = (int)(birth/100.0d);
		birth -= month*100;
		int day = birth;
		cal.set(Calendar.YEAR, year);
		cal.set(Calendar.MONTH, month-1);
		cal.set(Calendar.DAY_OF_MONTH, day);
		this.birth = cal.getTime();
		// もちろんint birthをStringにして，SimpleDateFormat
		// を用いた変換を行っても良い．
	}
	@Override
	public String toString(){
		SimpleDateFormat sdf = new SimpleDateFormat("yyyy/MM/dd");
		return this.name+"(ID:" + this.id + ") "+sdf.format(this.birth);
	}
	public int getId(){
		return this.id;
	}
	public static void setNextId(int id){
		Student.nextId = id;
	}
}
```

3-2	問題設定	Student クラスをせっかくシリアライズできるようにしたので，ファイルに読み書きできるようにしよう．
	クラス名	FileUtil
	メソッド	writeStudents()：学生をファイルに出力する 引数１：学生の一覧（ArrayList<Student>型） 引数２：出力ファイルパス（文字列型） 戻り値：なし 処理　：学生一覧のオブジェクトを全てファイルに出力する． readStudents()：学生をファイルから読み込む 引数　：入力ファイルパス（文字列型） 戻り値：学生の一覧（ArrayList<Student>型） 処理　：ファイルから学生一覧のオブジェクトを全て読み込む．~~学生一覧から一番新しい ID を検出し，Student クラスの static フィールドを忘れずに更新すること．~~
	諸注意	● 書き込み先フォルダが存在しない場合，フォルダを作成し書き込みを行う． ● 書き込み時既にファイルが存在していた場合，ファイルを上書きする（書き込み制限は行われていないものとしてよい）． ● 読み込み元ファイルが存在しないか同名のディレクトリが存在する場合，「ファイルがありません」と println で表示して終了する．終了は System.exit(1);とする．

	テスト例	（Main.javaのmainメソッドに以下をコピペする）
		`ArrayList<Student> list = new ArrayList<Student>();`
		`list.add(new Student("ブラウン加山", 19881202));`
		`list.add(new Student("ジェントル小岩", 19300101));`
		`list.add(new Student("トニー大岩", 19600101));`
		`FileUtil.writeStudents(list, "C:¥¥test¥¥student.ser");`
		
		`ArrayList<Student> read = FileUtil.readStudents("C:¥¥test¥¥student.ser");`
		`read.add(new Student("メンタル小池", 20000101));`
		`for(Student s : read)`
		`　　System.out.println(s.toString());`

	テスト結果例	ブラウン加山(ID:1) 1988/12/02 ジェントル小岩(ID:2) 1930/01/01 トニー大岩(ID:3) 1960/01/01 メンタル小池(ID:4) 2000/01/01

課題 3-2：解答例（FileUtil.java）

```
public static void writeStudents(ArrayList<Student> list, String
path){
	File file = new File(path);
	// 出力先の親ディレクトリの存在確認とディレクトリ生成
	if(!file.getParentFile().exists()){
		file.getParentFile().mkdirs();
	}

	// 実はArrayList<>自体がSerializableなので，Listの要素が
	// Serializableならばそのままるっとシリアライズできる．
	try(FileOutputStream fs = new FileOutputStream(file);
		ObjectOutputStream os = new ObjectOutputStream(fs)){
		// for文をまわしても良いがせっかくなので恩恵にあやかる．
		os.writeObject(list);
	}catch(IOException e){
		e.printStackTrace();
	}
}

public static ArrayList<Student> readStudents(String path){
	ArrayList<Student> list = null;
	File file = new File(path);
	// ファイルの存在確認とディレクトリでないことの確認
	if(!file.exists() ||
	   (file.exists() && file.isDirectory())){
		System.out.println("ファイルがありません");
		System.exit(1);
	}

	// ArrayList<>をまるっとファイルから読み込む
	try(FileInputStream fs = new FileInputStream(file);
		ObjectInputStream os = new ObjectInputStream(fs)){
		list = (ArrayList<Student>)os.readObject();

		// Student全員から最大のidを検索しnextIdを更新する．
		int max = -1;
		for(Student s : list){
			if(s.getId() > max) max = s.getId();
		}
		Student.setNextId(max + 1);
		// という処理を実装したものの，シリアライズした情報の中に
		// nextIdもしっかりと記録されていたため，この処理は不要
		// だった．．．失礼しました．
	} catch (IOException e) {
		e.printStackTrace();
	} catch (ClassNotFoundException e) {
		e.printStackTrace();
	}
	return list;
}
```

<table>
<tr><td rowspan="7">3-3</td><td>サービス問題</td><td>本問題は必須ではないが採点対象とする．3-2 までで 100 点だが，100 点を超えない範囲で本問題分加点する．</td></tr>
<tr><td>問題設定</td><td>Web から「kadai3.csv」を DL し，kadai3.csv を読み込んで，Student クラスを復元する処理を実装せよ．ついでに最高齢の学生を取得するメソッドも実装せよ．</td></tr>
<tr><td>メソッド</td><td>（以下は FileUtil クラスに static メソッドで実装せよ．）
readStudentsByCsv():学生をcsvファイルから読み込む
引数　：なし
戻り値：学生の一覧（ArrayList<Student>型）
処理　：ファイルからテキストを読み込み，Student 型に変換する．ID は新たに付番する．

（以下は Student クラスに static メソッドで実装せよ．）
getOldestStudent(): 最高齢の学生を取得する
引数　：学生の一覧（ArrayList<Student 型）
戻り値：最高齢の学生（Student 型）</td></tr>
<tr><td>諸注意</td><td>● 読み込み元ファイルが存在しないことはないものとして良い．
● csv の書式は以下の通りとする．
“名前１”，“2018/01/01”
“名前２”，” 2018/01/02”
※各要素はカンマ区切りで，ダブルクォーテーションで囲まれている．</td></tr>
<tr><td>ヒント</td><td>ダブルクォーテーションを文字列で表現する際にはエスケープシーケンス（¥）をつけること．</td></tr>
<tr><td>テスト例</td><td>（Main.javaのmainメソッドに以下をコピペする）
<pre>ArrayList<Student> list =
    FileUtil.readStudentsByCsv();
for(Student s : list){
    System.out.println(s.toString());
}
System.out.println(
    Student.getOldestStudent(list).toString());</pre></td></tr>
<tr><td>テスト結果例</td><td>Aaron(ID:1) 1949/03/21
Abel(ID:2) 1955/04/24
Abraham(ID:3) 2006/12/07
Abram(ID:4) 1993/04/06
Adam(ID:5) 1953/07/25
Adolph(ID:6) 1973/05/28
Adrian(ID:7) 1921/03/22

---- 中略 ----

Curtis(ID:99) 1993/02/04
Cyril(ID:100) 1977/01/26
Ambrose(ID:24) 1921/02/08</td></tr>
</table>

課題 3-3：解答例（FileUtil.java）

```
// CSVファイルからStudent配列に変換する.
public static ArrayList<Student> getStudentsByCsv(){
	ArrayList<Student> list = new ArrayList<Student>();
	// try-with-resources文でBufferedReaderを生成.
	try(FileReader fr = new FileReader("kadai3.csv");
		BufferedReader br = new BufferedReader(fr)){
		// すべての行を読み込む
		String str;
		while((str = br.readLine()) != null){
			// 今回のケースは各セルにカンマが含まれていないため
			// ダブルクォーテーションは削除して良さそうである.
			str = str.replace("¥"", "");
			// カンマ区切りにして各セルの値に分割する.
			String[] temp = str.split(",");
			Student s = new Student(temp[0],
					Integer.parseInt(temp[1]));
			list.add(s);
		}
	}catch(IOException e){
		e.printStackTrace();
	}
	return list;
}
```

課題 3-3：解答例（Student.java）

```
// 課題3-3の解答の一部
public static Student getOldestStudent(ArrayList<Student> list){
	// Date birthの最小値検索を行う.
	Student s = list.get(0);
	for(int i=0;i<list.size();i++){
		// getTime()のlong値を比較して最小の生年月日を検索する.
		if(s.birth.getTime() > list.get(i).birth.getTime()){
			s = list.get(i);
		}
	}
	return s;
}
```