SJC-P ストリーム解析 (Scanner)

・デフォルトのデリミタは空白文字、変更には useDelimiter() を使用する (1)
・hasXXX() は次のトークンが、XXX として有効かどうかを判定する hasNext(Pattern) も同様。なので、hasXXX() と nextXXX() を、以下のような使い方で、全トークンを処理することはできない。(2)

```
while (s.hasNextInt()) { // 予想に反して全件処理されない
  s.nextInt();
}
```

使用例

```
import java.io.StringReader;
import java.util.Scanner;
import java.util.regex.Pattern;

public class RegexTest2 {
    public static void main(String[] args) {
        RegexTest2 me = new RegexTest2();
        me.testScanner();
    }
    public void testScanner(){
        StringBuilder buf = new StringBuilder();
        buf.append("山田　太郎,052-123-4567,taro@xxx.co.jp,1\n");
        buf.append("鈴木　一郎,090-3211-2345,ichiro@xxx-yyy.ne.jp,2\n");
        buf.append("佐藤　花子,080-1111-2222,,3\n");

        AbstractFieldScanner[] fs = new AbstractFieldScanner[2];
        fs[0] = this.new TelNumScanner();
        fs[1] = this.new MailAddrScanner();
        for (AbstractFieldScanner sc : fs) {
            sc.scan(new StringReader(buf.toString()));
        }
    }
    private abstract class AbstractFieldScanner {
        public abstract Pattern getFiledPattern();
        public void scan(Readable source) {
            Scanner scan = new Scanner(source);
            scan.useDelimiter(",");      // デフォルトのデリミタを変更 (1)
            Pattern ptn = getFiledPattern();

            while (scan.hasNext()) {     // 全トークンを処理 (2)
                if (scan.hasNext(ptn)) { // 次のトークンが、パターンに一致するか判定
                    System.out.println(scan.next(ptn));
                } else {
                    scan.next();
                }
            }
        }
    }
    private class TelNumScanner extends AbstractFieldScanner {
        public Pattern getFiledPattern() {
            return Pattern.compile("\\d+[-]\\d+[-]\\d+");
        }
    }
    private class MailAddrScanner extends AbstractFieldScanner {
        public Pattern getFiledPattern() {
            return Pattern.compile(".+[@].+[.].+");
        }
    }
}
```