選択した問題は，選択欄の(選)をマークしてください。マークがない場合は，採点されません。

問11　次のJavaプログラムの説明及びプログラムを読んで，設問1，2に答えよ。

（Javaプログラムで使用するAPIの説明は，この冊子の末尾を参照してください。）

〔プログラムの説明〕

図1のように，文書の書式を表すひな形に置換表を適用して，出力文書を得るプログラムである。

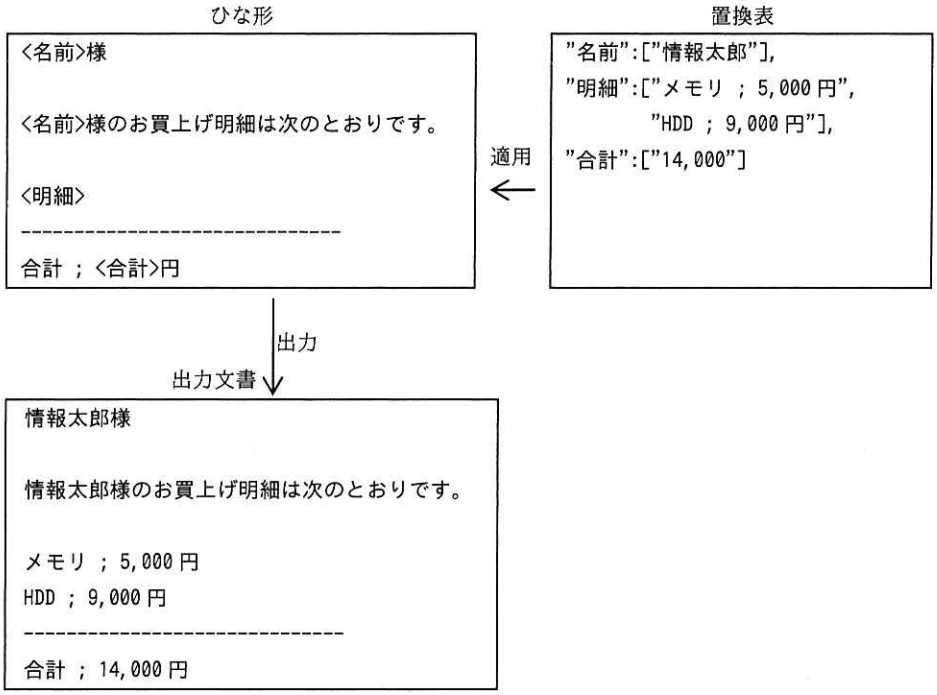

図1　ひな形に置換表を適用して出力文書を得る例

(1)　ひな形は，0個以上の置換指示と，0個以上のそのまま出力される置換指示以外の部分が連なるテキストである。

置換指示は，キー名称をくと＞で囲ったものである。ひな形中で，くと＞は，置換指示としてキー名称を囲う用途にだけ使用できる。

(2) 置換表は，キー名称とこれに対応する文字列の並びとの組みを一つ以上記述したテキストであり，それぞれの組みは，次の形式で記述する。

"キー名称":[文字列の並び]

組みを二つ以上記述するときは，コンマで区切る。

文字列の並びには，置換に用いる文字列（以下，置換用文字列という）を二重引用符で囲んだものを一つ以上記述する。置換用文字列を二つ以上記述するときは，コンマで区切る。例えば，置換用文字列が二つである場合は，次の形式で記述する。

"キー名称":["置換用文字列","置換用文字列"]

(3) ひな形に置換表を適用すると，ひな形中の〈キー名称〉は，次の規則に従って置換される。

① 置換表のキー名称に対応する文字列の並びに含まれる置換用文字列が一つだけのときは，その置換用文字列で置換される。

② 置換表のキー名称に対応する文字列の並びに含まれる置換用文字列が二つ以上あるときは，各置換用文字列の間に改行を挟んだ上で，並び順に連結してできる文字列で置換される。

図 1 の例では，置換表中のキー名称 名前 に対応する文字列の並びに含まれる置換用文字列は 情報太郎 だけなので，ひな形中の 〈名前〉 は，情報太郎 に置換される。

置換表中のキー名称 明細 に対応する文字列の並びには，置換用文字列として メモリ ； 5,000 円 と HDD ； 9,000 円 とが含まれるので，ひな形中の〈明細〉は，メモリ ； 5,000 円⏎HDD ； 9,000 円（⏎は改行）に置換される。

このプログラムでは，ひな形を，0 個以上の置換指示と 0 個以上の置換指示以外の部分が連なる文字ストリームとして扱う。個々の置換指示及び個々の置換指示以外の部分をフラグメントと呼ぶ。

インタフェース Fragment は，フラグメントを表す。

クラス Replacer は，置換指示を表す。

クラス PassThrough は，置換指示以外の部分を表す。

クラス TemplateParser のメソッド parse は，ひな形を表す文字ストリームからフラグメントのリストを構築し，クラス Template のインスタンスを生成して返す。ひな形に誤りはないものとする。

クラス Template は，ひな形をフラグメントのリストとして保持する。メソッド apply は，ひな形に置換表を適用して，出力文書を文字列で返す。置換表には，このひな形に含まれるキー名称とそれに対応する文字列の並びが含まれているものとする。

クラス ReplacementTableParser のメソッド parse は，置換表を表す文字ストリームから，キー名称とそれに対応する文字列の並びのマップを構築して，返す。置換表に誤りはないものとする（プログラムは省略）。

クラス TemplateTester はテスト用のプログラムである。テキストファイル template.txt が図 1 のひな形と同じ内容であって，テキストファイル replacementTable.txt が図 1 の置換表と同じ内容であるとき，実行結果は図 1 の出力文書と同じになる。

〔プログラム 1〕

```
import java.util.List;
import java.util.Map;

public interface Fragment {
   String replace(Map<String, List<String>> table);
}
```

〔プログラム 2〕

```
import java.util.List;
import java.util.Map;

public class Replacer [ a ] Fragment {
   final String key;

   Replacer(CharSequence cs) { key = cs.toString(); }

   public String replace(Map<String, List<String>> table) {
      return String.join("\n", table.get(key));
   }
}
```

〔プログラム 3〕

```
import java.util.List;
import java.util.Map;

public class PassThrough [ a ] Fragment {
   final String str;

   PassThrough(CharSequence cs) { str = cs.toString(); }

   public String replace(Map<String, List<String>> table) {
      return str;
   }
}
```

〔プログラム 4〕

```
import java.io.IOException;
import java.io.Reader;
import java.util.ArrayList;
import java.util.List;

public class TemplateParser {
   static public Template parse(Reader reader) throws IOException {
      StringBuilder buf = new StringBuilder();
      List<Fragment> fragmentList = new ArrayList<>();
      int c;
      while ((c = reader.read()) >= 0) {
         switch (c) {
            case '<' :
               fragmentList.add(new [ b ]);
               buf = new StringBuilder();
               break;
            case '>' :
               fragmentList.add(new [ c ]);
               buf = new StringBuilder();
               break;
            default :                                    <-- α
               buf.append((char) c);
         }
      }
      fragmentList.add(new PassThrough(buf));
      return [ d ];
```

```
    }
}
```

〔プログラム5〕

```
import java.util.List;
import java.util.Map;

public class Template {
    List<Fragment> fragmentList;

    Template(List<Fragment> fragmentList) {
        this.fragmentList = fragmentList;
    }

    public String apply(Map<String, List<String>> table) {
        StringBuilder sb = new StringBuilder();
        for (Fragment fragment : fragmentList) {
            sb.append(fragment.replace(    e    ));
        }
        return sb.toString();
    }
}
```

〔プログラム6〕

```
import java.io.FileReader;
import java.io.IOException;
import java.io.Reader;
import java.util.List;
import java.util.Map;

public class TemplateTester {
    public static void main(String... args) throws IOException {
        try (
            Reader tReader = new FileReader("template.txt");
            Reader rReader = new FileReader("replacementTable.txt")
        ) {
            Template template = TemplateParser.parse(tReader);
            Map<String, List<String>> table =
                ReplacementTableParser.parse(rReader);
            System.out.print(template.apply(table));
        }
    }
}
```

設問1　プログラム中の［　　］に入れる正しい答えを，解答群の中から選べ。

aに関する解答群

ア　extends　　イ　interface　　ウ　implements　　エ　throws

b，cに関する解答群

ア　Fragment(buf)　　イ　Fragment(c)　　ウ　PassThrough(buf)

エ　PassThrough(c)　　オ　Replacer(buf)　　カ　Replacer(c)

dに関する解答群

ア　fragmentList　　イ　new Template(buf)

ウ　new Template(fragmentList)　　エ　new Template(reader)

eに関する解答群

ア　fragment　　イ　fragmentList

ウ　sb　　エ　table

設問2　次の記述中の［　　］に入れる正しい答えを，解答群の中から選べ。

<と>は，置換指示としてキー名称を囲う用途以外では，ひな形中で使用することができない。そこで，これらの文字を他の用途でも使用できるように，次の2行をプログラム4のクラスTemplateParserのαの位置に挿入した。これによって，\に続く1文字（\が複数個連続するときは奇数個目に続く1文字）は，置換指示以外の部分やキー名称の一部として扱われる。ここで，続く1文字は必ず読めるものとする。

```
case '\\' :
    [ f ];
```

fに関する解答群

ア　break

イ　buf.append((char) c)

ウ　buf.append((char) c); break

エ　buf.append((char) reader.read())

オ　buf.append((char) reader.read()); break