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1～4に答えよ。

〔プログラムの説明〕

A君は，先輩エンジニアであるBさんの指導の下で，表現式（以下，式という）を構築するためのライブラリの作成を進めている。ライブラリはAPIを提供する。

まず，次のインタフェース及びクラスをパッケージ com.example.expr に作成した。

(1) インタフェースExpressionは，式を表す。すなわち，Expressionを実装するクラスは，型としての式を表す。

① メソッドevaluateは，式を評価した結果を型intで返す。

(2) クラスConstantは，定数を式（型Expression）として表す。

① コンストラクタは，引数で与えられた型intの値を表す定数を生成する。

② メソッドevaluateは，この定数の値を返す。

③ メソッドtoStringは，この定数の値を文字列として返す。

(3) クラスAdditionは，加算を式（型Expression）として表す。

① コンストラクタは，加算式を生成する。加算式a＋bにおいて，演算子+の左側aの部分を左側の式，右側bの部分を右側の式とし，それぞれ引数left及び引数rightで指定する。いずれかの引数がnullのときは，NullPointerExceptionを投げる。

② メソッドevaluateは，左側の式を評価した値に右側の式を評価した値を加え，その値を返す。

③ メソッドtoStringは，この加算式を表す文字列を返す。

〔プログラム1〕

```
package com.example.expr;

public interface Expression {
   public int evaluate();
}
```

〔プログラム2〕

```
package com.example.expr;

public class Constant implements Expression {
   private final int value;

   public Constant(int value) {
      this.value = value;
   }

   public int evaluate() { return value; }

   public String toString() { return String.valueOf(value); }
}
```

〔プログラム3〕

```
package com.example.expr;

public class Addition implements Expression {
   private final Expression left, right;

   public Addition(Expression left, Expression right) {
      if (left == null || right == null) {
         throw new NullPointerException();
      }
      this.left = left;
      this.right = right;
   }

   public int evaluate() {
      return left.evaluate() + right.evaluate();
   }

   public String toString() {
      return String.format("(%s + %s)", left, right);
   }
}
```

設問1　A君は，クラス Constant 及び Addition を API として呼び出すテストをするために，クラス Test をパッケージ com.example.test に作成して実行したところ，図 1 の出力結果を得た。プログラム 4 中の［　　］に入れる正しい答えを，解答群の中から選べ。

```
(2 + 5) = 7
```

図1　メソッド main の出力結果

〔プログラム4〕

```
package com.example.test;

import [ a ].Addition;
import [ a ].Constant;
import [ a ].Expression;

public class Test {
   public static void main(String[] args) {
      Expression two = new Constant(2);
      Expression five = new Constant(5);
      Expression add = new Addition( [ b ] );
      System.out.println( [ c ] + " = " + [ d ] );
      /* α */
   }
}
```

aに関する解答群

ア　com.example.expr　　イ　com.example.test
ウ　static com.example.expr　　エ　static com.example.test

bに関する解答群

ア　2, 5　　イ　5, 2　　ウ　five
エ　five, two　　オ　two　　カ　two, five

c，dに関する解答群

ア　add　　イ　add.evaluate()　　ウ　five
エ　five.evaluate()　　オ　two　　カ　two + five
キ　two.evaluate()

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。

A君は，これまでに作成したプログラムをBさんに見てもらったところ，次のアドバイスを受けた。

“乗除算などの式を実装するクラスを追加する前に，上位クラスを導入して二項演算を表すクラスを簡単に実装できるようにすべきである。また，アプリケーションが必要とする全ての二項演算をライブラリで用意するのは一般に難しいので，パッケージ外でも二項演算がそのクラスの下位クラスとして簡単に実装できるようにすべきである。”

そこで，A君は，次の仕様の抽象クラスをライブラリに追加した。

(1) 抽象クラス BinaryOperatorExpression は，二項演算子を使用する式（以下，二項演算式という）を表す。

① コンストラクタは，二項演算式を生成する。二項演算式 a OP b において，二項演算子 OP の左側 a の部分を左側の式，右側 b の部分を右側の式とし，それぞれ引数 left 及び引数 right で指定する。いずれかの引数が null のときは，NullPointerException を投げる。

② メソッド getLeft 及び getRight は，それぞれ左側の式及び右側の式を返す。

③ 抽象メソッド getOperator は，この二項演算式の演算子を文字列で返す。

④ メソッド toString は，この二項演算式を表す文字列を返す。

このクラスをBさんに見てもらったところ，“ライブラリで用意されたAPIを呼び出すアプリケーションで二項演算式を表すクラスの実装に使えるようにすることが目的なので，［　　］ようにするために，コンストラクタとメソッド getLeft，getRight 及び getOperator は public ではなく protected にすべきである”という指摘を受け，その修正をしてプログラム5を作成した。

〔プログラム5〕

```
package com.example.expr;

public abstract class BinaryOperatorExpression implements Expression {
   private final Expression left, right;

   protected BinaryOperatorExpression(Expression left, Expression right) {
      if (left == null || right == null) {
         throw new NullPointerException();
      }
      this.left = left;
      this.right = right;
   }

   protected Expression getLeft() {
      return left;
   }

   protected Expression getRight() {
      return right;
   }

   protected abstract String getOperator();

   public String toString() {
      return String.format("(%s %s %s)",
            getLeft(), getOperator(), getRight());
   }
}
```

解答群

ア　下位クラスを実装するクラスだけで使える

イ　上位クラスの実装が下位クラスで改変されない

ウ　通常のアプリケーションがどのパッケージからでも呼び出せる

エ　パッケージ com.example.expr 内の下位クラスだけで使える

設問３　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。ここで，プログラム６中の［　a　］には，設問１の正しい答えが入っているものとする。

プログラム６は，ライブラリのパッケージ外で二項演算式を実装するテストとして，減算式を表すクラス Subtraction を実装したものである。

〔プログラム６〕

```
package com.example.test;

import [ a ].BinaryOperatorExpression;
import [ a ].Expression;

public class Subtraction extends BinaryOperatorExpression {
   public Subtraction(Expression left, Expression right) {
      super(left, right);
   }

   public int evaluate() {
      return getLeft().evaluate() - getRight().evaluate();
   }

   public String getOperator() {
      return "-";
   }
}
```

プログラム６をテストするために，プログラム４のメソッド main の /* α */ の行を図２に示す行に置き換えて実行したところ，図１に示した出力結果に加え，出力結果“［　　］”を得た。

```
System.out.println(new Subtraction(add, new Subtraction(two, five)));
```

図２　メソッド main の /* α */ の行を置き換える行

解答群

ア　((2 + 5) - (2 - 5))　　イ　((7) - (-3))　　ウ　(2 + 5 - 2 - 5)

エ　(2 + 5) - (2 - 5)　　オ　0　　カ　10

設問4　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。

Bさんから，更に次の指摘を受けた。

“オブジェクトの等価についての考慮も必要である。例えば，［　　］を評価すると，二つのインスタンスが表す値が等しいので true になるべきだが，今のクラス Constant の実装では false になる。”

解答群

ア　new Constant(9) == new Constant(9)

イ　new Constant(9).equals(new Constant(9))

ウ　new Constant(9).evaluate() == new Constant(9).evaluate()

エ　new Constant(9).evaluate().equals(new Constant(9).evaluate())