平成１８年度　春期　ＦＥ　午後問題　Java

問８ 次の Java プログラムの説明及びプログラムを読んで，設問に答えよ。

〔プログラムの説明〕

PC のセキュリティに関する更新プログラム（以下，パッチという）の適用状況を管理するプログラムである。

クラス PCChecker は，PC に適用が要求されるパッチと各 PC に適用済みのパッチを管理し，その適用状況の一覧を出力する。PC のパッチ情報として次のものが登録される。

① PC の識別子
② PC の種別（デスクトップ PC 又はノート PC）
③ 適用済みのパッチの識別子（複数の場合は複数個）
④ ノート PC の場合，暗号化ツールがインストールされているか否か

PC に問題がない（secure）と判断する基準は，次のとおりである。

デスクトップ PC：適用が要求されるパッチがすべて適用されている。

ノート PC：　適用が要求されるパッチがすべて適用されている。さらに，暗号化ツールがインストールされている。

(1) クラス PCChecker は，PC のパッチ情報を管理するために，クラス DesktopPC とクラス NotebookPC のインスタンスを用いる。

(2) クラス PC は，デスクトップ PC とノート PC に共通の処理を実現する。メソッド installPatch は，指定されたパッチを適用済みにする。メソッド installed は，指定されたパッチが適用済みのとき true，未適用のとき false を返す。メソッド requiresEncryptionTool は，暗号化ツールのインストールが必要なとき true，不必要なとき false を返す。

(3) クラス DesktopPC は，デスクトップ PC を表し，デスクトップ PC 独自の処理を実現する。

(4) クラス NotebookPC は，ノート PC を表し，ノート PC 独自の処理を実現する。

(5) クラス PCChecker のメソッド main は，このプログラムの使用例である。その実行結果を，次に示す。

```
pc001: requires [sp1-A01, sp1-A03]
pc002: requires encryption tool
pc003: secure
```

(6) クラス java.util.ArrayList は，オブジェクトをリストで管理する手段を提供する。次のメソッドを利用している。

public boolean add(Object obj)
　オブジェクト obj をリストに追加し，true を返す。

public boolean contains(Object obj)
　オブジェクト obj がリストに含まれる場合に true，含まれない場合に false を返す。

public boolean isEmpty()
　リストが空の場合に true，空でない場合に false を返す。

〔プログラム１〕

```
import java.util.List;
import java.util.ArrayList;

public abstract class PC {
  private String id;
  private List installedPatches
                        = new ArrayList();
  public PC(String id) { this.id = id; }
  public String getID() { return id; }
  public void installPatch(String patch) {
    installedPatches.add(patch);
  }
  public boolean installed(String patch) {
    return [ a ];
  }
  public abstract Boolean
                requiresEncryptionTool();
}
```

〔プログラム２〕

```
public class DesktopPC [ b ] {
  public DesktopPC(String id) {
                    [ c ]; }
  public boolean requiresEncryptionTool() {
    return false;
  }
}
```

〔プログラム３〕

```
public class NotebookPC [ b ] {
  private boolean hasEncryptionTool;
  public NotebookPC(String id, boolean
                        hasEncryptionTool) {
    [ c ] ;
    this.hasEncryptionTool
                        = hasEncryptionTool;
  }
  public boolean requiresEncryptionTool() {
    return !hasEncryptionTool;
  }
}
```

〔プログラム４〕

```
import java.util.List;
import java.util.ArrayList;

public class PCChecker {
   public static void main(String[] args) {
      // PC のパッチ情報を登録する。
      PC[] pc = new PC[3];
      pc[0] = new DesktopPC("pc001");
      pc[1] = new NotebookPC("pc002", false);
      pc[2] = new NotebookPC("pc003", true);
      pc[0].installPatch("sp1-A02");
      pc[1].installPatch("sp1-A01");
      pc[1].installPatch("sp1-A02");
      pc[1].installPatch("sp1-A03");
      pc[2].installPatch("sp1-A01");
      pc[2].installPatch("sp1-A02");
      pc[2].installPatch("sp1-A03");
      // PC の状態を出力する。curPatches は，適用が
         要求されるパッチを示す。
      String[] curPatches = { "sp1-A01",
                        "sp1-A02","sp1-A03"};
      for (int i = 0; i < pc.length; i++) {
         System.out.print(pc[i].getID() + ":");
         List notApplied = new ArrayList();
         for (int j = 0; j < curPatches.length;
                                        j++)
            if ( [ d ] )
               notApplied.add(curPatches[j]);
         if (!notApplied.isEmpty() ||
             pc[i].requiresEncryptionTool()) {
            if (!notApplied.isEmpty())
               System.out.println(" requires "
                              + notApplied);
            if (pc[i].
                    requiresEncryptionTool())
               System.out.println(" requires
                           encryption tool");
         } else
            System.out.println(" secure");
      }
   }
}
```

設問　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　installedPatches.isEmpty()

イ　!installedPatches.isEmpty()

ウ　installedPatches.contains(patch)

エ　!installedPatches.contains(patch)

オ　patch == null

カ　patch != null

b に関する解答群

ア　extends Object

イ　extends PC

ウ　extends PCChecker

エ　implements Object

オ　implements PC

カ　implements PCChecker

c に関する解答群

ア　PC()　　イ　PC(id)

ウ　super()　　エ　super(id)

オ　this.id = id

d に関する解答群

ア　curPatches[j] != null

イ　notApplied.isEmpty()

ウ　pc[i] != null

エ　pc[i].installed(curPatches[j])

オ　!pc[i].installed(curPatches[j])

問 12　次の Java プログラムの説明及びプログラムを読んで，設問に答えよ。

〔プログラムの説明〕

トランプを用いた一人遊びのゲームを実行するプログラムである。ゲームのルールは，次のとおりである。

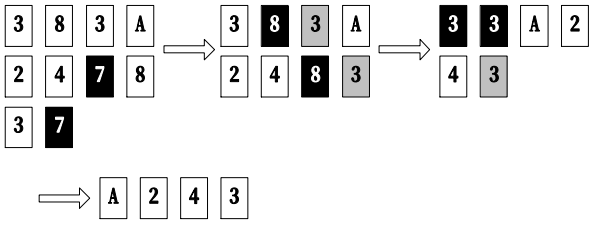
(1)　トランプは，スペード，ハート，ダイヤ及びクラブの 4 種類のスートがあり，各スートは A（エース），2 ～ 10，J（ジャック），Q（クイーン）及び K（キング）の 1～13 の位（ランク）を表す 13 枚のカードからなる。このゲームでは合計 52 枚のカードを使用し，スートは区別しない。

(2)　52 枚のカードを，1 枚ずつ表を向けて横に並べていく。4 枚並べたら，その下の列に移り，再び左端から横に並べる。カードを並べていくところを場と呼び，横の並びを列と呼ぶ。

(3)　図に示すとおり，(2) で 1 枚並べるごとに，縦，横，斜めに隣り合ったカードが同じ位であるかどうかを調べ，同じ位のカードの組み（ペア）ができた場合は，そのペア（図中の 7 ）を取り除き，空いたスペースを詰める。列の最後のカードと次の列の最初のカードは隣り合ったカードとはみなさない。ペアが複数できた場合（ 8 と 3 の各 2 枚）は，最も長く場にあるカードを含むペア

（8）を取り除く。最も長く場にあるカードを含むペアが複数ある場合は，最も長く場にあるカードとペアを作るカードのうちで，最も長く場にあるカードとのペア（3）を取り除く。

(4)　(3)の操作を隣り合うカードの位が全部異なるようになるまで繰り返す。



(5)　(2)～(4)の操作を手持ちのカードがなくなるまで繰り返す。

(6)　手持ちのカードがなくなったときに，すべてのカードが場から取り除かれると上がりである。

クラス Card は，トランプのカードを表す。クラスの初期化のとき，A（エース）～ K（キング）の位に相当する Card のインスタンスを 4 種類のスート分生成して Card の配列 cards に格納する。Card のインスタンスは不変であり，1 枚のカードに必ず同一のインスタンスが対応する。例えば，ハートのエースを表す Card のインスタンスは一つしか存在しない。クラスメソッド newDeck は，cards をランダムな順番に並べ替えた Card の配列をトランプの一山として返す。

クラス Game は，ゲームを実行するプログラムである。List のインスタンス list がトランプを並べていく場を表し，トランプの山を表す deck から 1 枚ずつ list に追加し，その都度メソッド checkAndRemove を呼び出して同じ位の隣り合うカードのペアを取り除く。

クラス java.util.Random は，乱数を生成するためのクラスである。メソッド nextInt(int n) は，範囲 0 ～ n－1 の乱数を int 型で返す。

インタフェース java.util.List は，リスト構造を表し，各要素はインデックスで指定される。リストの最初の要素は，インデックスの値 0 で指定される。メソッド add(Object obj) は，リストの最後にオブジェクト obj を追加する。メソッド get(int index) は，index で指定された要素のオブジェクトを返す。メソッド remove(int index) は，index で指定された要素のオブジェクトを削除し， index + 1 以降にオブジェクトがあれば，それらをシフトして空きを詰める。メソッド size() は，リストにあるオブジェクトの個数を int 型で返す。

クラス java.util.ArrayList は，配列を用いてインタフェース List を実装する。

〔プログラム１〕

```
import java.util.Random;

public class Card {
  public static final int SPADES = 0;
  public static final int HEARTS = 1;
  public static final int DIAMONDS = 2;
  public static final int CLUBS = 3;

  private static final Random rand
                                  = new Random();
  private static final Card[] cards
                              = new Card[13 * 4];
  private final int suit;
  private final int rank;

  static {
    for (int i = SPADES; i <= CLUBS; i++) {
      for (int j = 1; j <= 13; j++) {
        cards[                b                ]
                              = new Card(i, j);
      }
    }
  }

  private Card(int suit, int rank) {
    this.suit = suit;
    this.rank = rank;
  }

  public int getSuit() { return suit; }
  public int getRank() { return rank; }

  public static Card[] newDeck() {
    Card[] deck = new Card[cards.length];
    System.arraycopy(cards, 0, deck, 0,
                             cards.length);
    for (int i = cards.length - 1;
                              i > 0; i--) {
      int index = rand.nextInt(i + 1);
      Card tmp = deck[i];
      deck[i] = deck[index];
      deck[index] = tmp;
    }
    return deck;
  }
}
```

〔プログラム２〕

```
import java.util.ArrayList;
import java.util.List;

public class Game {
  // 現在注目しているカードからの相対インデックス
  private static final int[][] indexDiff = {
    { 1, 4, 5 },
    { 1, 3, 4, 5 },
    { 1, 3, 4, 5 },
```

```
    { 3, 4 }
  };

  private static void checkAndRemove
                                (List list) {
    // list の最初からその隣り合うカードを調べる。
    // currentIndex は現在注目しているカードの
                                    インデックス値
    for (int currentIndex = 0; currentIndex
          < list.size(); currentIndex++) {
       // currentRank は現在注目しているカードの位
      int currentRank = ((Card) list.get
              (currentIndex)).getRank();
       // 現在注目しているカードと隣り合うカードへの
      // インデックスを求める。
      int[] diffList = indexDiff
                           [currentIndex % 4];
      for (int i = 0; i < diffList.length
                                      ; i++) {
        // adjacentIndex は隣り合うカードへのイン
        デックス値
        int adjacentIndex =
                    [          c          ];
        if ([          d          ]
            && ((Card) list.get
                (adjacentIndex)).getRank()
                 == currentRank) {
          list.remove(adjacentIndex);
          list.remove
                  ([          e          ]);
          checkAndRemove(list);
          return;
        }
      }
    }
  }

  public static void main(String[] args) {
    Card[] deck = Card.newDeck();
    List list = new ArrayList();
    for (int i = 0; i < deck.length; i++) {
      list.add(deck[i]);
      checkAndRemove(list);
    }
    if (list.size() == 0) {
      System.out.println("上がり！");
    } else {
      System.out.println
              ("残り" + list.size() + "枚");
    }
  }
}
```

設問　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　private Card() {
イ　private static void init() {
ウ　private void init() {
エ　static {
オ　synchronized {
カ　{

b に関する解答群

ア　i * 13 + j
イ　i * 13 + j - 1
ウ　i * 4 + j − 1
エ　j * 13 + i
オ　j * 13 + i − 1
カ　j * 4 + i - 1

c に関する解答群

ア　currentIndex * diffList[i]
イ　currentIndex + diffList[i]
ウ　currentIndex - diffList[i]
エ　i * diffList[currentIndex]
オ　i + diffList[currentIndex]
カ　i - diffList[currentIndex]

d に関する解答群

ア　adjacentIndex != list.size()
イ　adjacentIndex < list.size()
ウ　adjacentIndex <= list.size()
エ　adjacentIndex == list.size()
オ　adjacentIndex > list.size()
カ　adjacentIndex >= list.size()

e に関する解答群

ア　0
イ　adjacentIndex
ウ　adjacentIndex - 1
エ　currentIndex
オ　currentIndex - 1
カ　list.size() - 1

平成１８年度　秋期　ＦＥ　午後問題　Java

問８ 次の Java プログラムの説明及びプログラムを読んで，設問に答えよ。

（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〕

クラス UniqueMap は，キーと値を１対１に対応付けて保持する。

例えば，乗物の乗客（キー）と座席（値）との間に着席という関連があるとき，一つの座席には多くても１人の乗客だけが座り，１人の乗客は一つの席にしか座らない。クラス UniqueMap はこのような２種類のオブジェクトを，一方をキー，他方を値として１対１に対応させるクラスであり，次のメソッドをもつ。

public V put(K key, V value)
　キー（key）と値（value）を１対１に対応付けて登録する。key 又は value が null ならば，NullPointerException を投げる。また，value が既にほかのキーと対応付けられていれば，IllegalArgumentException を投げる。
　key が既にほかの値と対応付けられていれば，その値を value で置き換え，置き換えられる前の値を返す。key に値が対応付けられていなければ，null を返す。

public V get(K key)
　キー（key）に対応付けられた値を返す。key と値の対応付けがなければ，null を返す。

public V remove(K key)
　キー（key）と値の対応付けを削除し，対応付けられていた値を返す。key と値の対応付けがなければ，null を返す。

public void putForce(K key, V value)
　キー（key）と値（value）を１対１に対応付けて登録する。value が既にほかのキーと対応付けられていれば，その対応を削除したうえで登録する。戻り値はない。その他の仕様は，メソッド put と同一である。

〔プログラム〕

```
import java.util.Map;
import java.util.HashMap;

public class UniqueMap<K, V> {   // K はキーのクラス，V は値のクラス。
  // キーと値を対応させて記憶する。
  private Map<K, V> map = new HashMap<K, V>();
  // 値からキーを取り出せるようにする。
  private Map<V, K> reverse = new HashMap
                                    <V, K>();

  public V put(K key, V value) {
    if ( [ a ] )
      throw new NullPointerException();
    if (reverse.containsKey(value))
      throw new IllegalArgumentException();
    reverse.remove(map.get(key));
    reverse.put( [ b ] );
    return map.put(key, value);
  }

  public V get(K key) {
    return map.get(key);
  }

  public V remove(K key) {
    V value = map.remove(key);
    reverse.remove(value);
    return value;
  }

  public void putForce(K key, V value) {
    // value がほかのキーと対応付けられていれば，その対応付けを削除する。
    // value に対応するキーの存在の有無をチェックする必要はない。
    map.remove( [ c ] );
    [ d ] (key, value);
  }
}
```

設問　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　key == null || value == null
イ　key == null && value == null
ウ　key != null || value != null
エ　key != null && value != null

b に関する解答群

ア　key, key　　イ　key, value
ウ　value, key　　エ　value, value

c に関する解答群

ア　key
イ　reverse.get(value)
ウ　reverse.put(value,key)
エ　reverse.remove(value)

d に関する解答群

ア　map.put　　イ　put
ウ　putForce　　エ　reverse.put

問 12　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〕

図書の貸出しと返却の処理をするプログラムである。図書には書籍と雑誌があり，利用者が借りることのできる図書の冊数には上限がある。

(1) 抽象クラス Book は，図書に共通の属性と処理を定義する。属性 name は図書の名前であり，属性 id は図書に一意に付けられた識別子である。

(2) クラス RegularBook は書籍を表す。

(3) クラス Magazine は雑誌を表す。属性 issueNo は，雑誌の創刊号からの号数で，古い号から昇順の値をもつ。各雑誌について，issueNo が最大のものが最新号である。

(4) クラス User は図書の利用者を表す。属性 name は利用者の名前であり，属性 id は利用者に一意に付けられた識別子である。

(5) クラス Library は，貸出しと返却の処理をする。属性 availables は貸出し可能な図書の集合を表し，属性 checkedOut は貸出し中の図書の集合を表す。属性 limit は利用者が借りることのできる図書の冊数の上限を与える。メソッド checkoutBook は，貸出し処理を実行する。指定された図書が既に貸し出されている場合と利用者が既に貸出し冊数の上限まで図書を借りている場合，例外を投げる。メソッド returnBook は返却処理を実行する。

メソッド main はテスト用のメインプログラムである。実行結果を図に示す。

```
Taro is checking out Magazine:
                  JITEC News, No.3912 (M001)
Taro is checking out Magazine:
                  JITEC News, No.4001 (M002)
Taro is checking out Book:
                  Java Programming (P001)
Taro is checking out Book:
                  Ruby Programming (P003)
   => failed: exceeding checkout limit
Taro returned Magazine:
                  JITEC News, No.3912 (M001)
Hana is checking out Magazine:
                  JITEC News, No.3912 (M001)
Hana is checking out Book:
                  Java Programming (P001)
   => failed: unavailable
```

図　実行結果

〔プログラム〕

（行番号）

```
 1  import java.util.*;
 2
 3  abstract class Book {
 4    String name;
 5    String id;
 6    Book(String name, String id) {
 7      this.name = name;
 8      this.id = id;
 9    }
10    public boolean equals(Object object) {
11      return (object instanceof Book) &&
                  id.equals(((Book) object).id);
12    }
13    public int hashCode() {
14      return id.hashCode();
15    }
16  }
17
18  class RegularBook extends Book {
19    RegularBook(String name, String id) {
20        [          a          ];
21    }
22    public String toString() {
23      return "Book: " + name + " ("
                                    + id + ")";
24    }
25  }
26
27  class Magazine extends Book {
28    int issueNo;
29    Magazine(String name, int issueNo,
                                  String id) {
30        [          a          ];
31      this.issueNo = issueNo;
32    }
33    public String toString() {
34      return "Magazine: " + name + ", No."
                + issueNo + " (" + id + ")";
35    }
36  }
37
38  class User {
39    String name;
40    String id;
41    User(String name, String id) {
42      this.name = name;
43      this.id = id;
44    }
45    public String toString() {
46      return "User: " + name + " ("
                                    + id +")";
47    }
48  }
49  class Library {
50    Set<Book> availables
                        = new HashSet<Book>();
51    Map<Book, User> checkedOut
                    = new HashMap<Book, User>();
52    int limit;
53    Library(Book[] books, int limit) {
54      for (Book book : books) register
                                      (book);
55      this.limit = limit;
```

```
56    }
57    void register(Book book) {
58      availables.add(book);
59    }
60    void checkoutBook(User user, Book book)
                          throws Exception {
61      if (! availables.contains(book))
62          [          b          ]
                              ("unavailable");
63      int count = 0;
64      for (Map.Entry<Book, User> entry :
                      checkedOut.entrySet())
65        if (entry.getValue().
                       equals(user))count++;
66      if (count >= limit)
67          [          b          ]
                ("exceeding checkout limit");
68      availables.remove(book);
69      checkedOut.put(book, user);
70    }
71    void returnBook(Book book) {
72      for (Map.Entry<Book, User> entry :
                      checkedOut.entrySet())
73        if (entry.getKey().equals(book)) {
74          [          c          ](book);
75          availables.add(book);
76          return;
77        }
78    }
79    public static void main(String[] args){
80      Book
81        java  = new RegularBook
                ("Java Programming", "P001"),
82        perl  = new RegularBook
                ("Perl Programming", "P002"),
83        ruby  = new RegularBook
                ("Ruby Programming", "P003"),
84        jit39_12 = new Magazine
                ("JITEC News", 3912, "M001"),
85        jit40_01 = new Magazine
                ("JITEC News", 4001, "M002");
86      Book[] books = {java, perl, ruby,
                        jit39_12, jit40_01};
87      Library library = new Library
                                  (books,3);
88      User taro = new User("Taro", "ID-01");
89      User hana = new User("Hana", "ID-02");
90      Book[] books1 = {jit39_12, jit40_01,
                               java, ruby};
91      for (Book book : books1)
92        try {
93          System.out.print("Taro is
                      checking out " + book);
94           library.checkoutBook
                              (taro, book);
95          System.out.println();
96        }
97        catch (Exception e) {
98          System.out.println("\n" + " =>
              failed: " + e.getMessage());
99        }
100     library.returnBook(jit39_12);
101     System.out.println("Taro returned "
                                + jit39_12);
102     Book[] books2 = {jit39_12, java};
103     for (Book book : books2)
104       try {
105         System.out.print("Hana is
                      checking out " + book);
106         library.checkoutBook
                              (hana, book);
107         System.out.println();
108       }
109       catch (Exception e) {
110           System.out.println("\n" + " =>
                failed: " + e.getMessage());
111       }
112   }
113 }
```

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　new Book()

イ　new Book(name, id)

ウ　super()

エ　super(name, id)

オ　this.name = name; this.id = id

b に関する解答群

ア　return Exception

イ　return new Exception

ウ　System.err.println

エ　System.out.println

オ　throw Exception

カ　throw new Exception

c に関する解答群

ア　checkedOut.add

イ　checkedOut.remove

ウ　entry.add

エ　entry.remove

設問２　雑誌について，最新号の貸出しを禁止する処理を追加したい。このため，次のようにプログラムを変更する。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。ただし，各雑誌の各号は１冊ずつしかなく，雑誌の登録は，必ずしも古い号から行われるとは限らない。また，[b] には設問１の正しい答えが入っているものとする。

処置	追加内容
行番号 50 の直後に追加	`Set<Magazine> latestMagazines = new HashSet<Magazine>();`
行番号 58 の直後に追加	`if (book instanceof Magazine) updateLatestMagazineList ((Magazine) book);`
行番号 62 の直後に追加	`if (book instanceof Magazine) { Magazine magazine = (Magazine) book; if (latestMagazines.contains (magazine)) [ b ] ("latest issue"); }`
行番号 78 の直後に追加	`void updateLatestMagazineList (Magazine magazine) { for (Magazine magazine2 : latestMagazines) if(magazine2.name.equals (magazine.name)){ if (magazine.issueNo [ d ] magazine2. issueNo) { return; } else { latestMagazines.remove (magazine2); break; } } [ e ] ; }`

d に関する解答群

ア　<　　イ　>　　ウ　!=　　エ　==

e に関する解答群

ア　latestMagazines.add(magazine)

イ　register((Magazine) book)

ウ　return

エ　returnBook(magazine)

平成１８年度　春期　ＦＥ　午後解答　Java

問８

設問
a－ウ　　b－イ　　c－エ　　d－オ

問 12

設問
a－エ　　b－イ　　c－イ　　d－イ　　e－エ

平成１８年度　秋期　ＦＥ　午後解答　Java

問８

設問
a－ア　　b－ウ　　c－エ　　d－イ

問 12

設問１
a－エ　　b－カ　　c－イ

設問２
d－ア　　e－ア