注意　問題は 3 問あります。問題中の Java プログラムの行頭の数は、説明の都合上付けた行番号であり、プログラムの一部ではありません。また、この科目のクラスライブラリを利用している場合があります。解答はすべて別紙の解答用紙に記入しなさい。

問題 I　次の Java プログラム Q01.java をコンパイルして起動すると、ウィンドウの中にトランプのデッキ (ジョーカーを含まない 52 枚の 1 揃えのカードの山) と空の山が現われ、デッキがシャッフルされた後、デッキから 1 枚ずつカードを引き、表向きにして、画面の下部に左から順に並べていく。ただし、2 枚目以降に引いたカードは、最後に画面下部に並べたカードと同じスートであるか、あるいは、同じランクのカードであるときのみ、画面下部に並べ、そうでなければデッキとは別の山に表向きのまま捨てる。

Q01.java

```
 1 import jp.ac.ryukoku.math.cards.*;
 2
 3 class Q01 {
 4     public static void main(String[] args) {
 5         GameFrame frame = new GameFrame();
 6         Pile p = [ (a) ];
 7         frame.add(p, 100, 100);
 8         Pile q = [ (b) ];
 9         frame.add(q, 300, 100);
10         q.shuffle();
11         Card last = null;
12         int num = 0;
13         while (!q.isEmpty()) {
14             Card c = q.pickUp();
15             [ (c) ];
16             if (last == [ (d) ] || [ (e) ]) {
17                 c.moveTo(100 + num * 25, 300);
18                 last = c;
19                 num++;
20             } else {
21                 [ (f) ];
22             }
23         }
24     }
25 }
```

(1) 空欄 (a) ～ (f) に補うべき式を解答用紙の解答欄に書きなさい。(27 点)

(2) このプログラムに現れている下表中の語が、次のいずれであるかを、解答用紙の解答欄に 1 ～ 8 の数字で示しなさい。(18 点)

1. クラス名　　2. インスタンスメソッド名　　3. クラスメソッド名　　4. クラス変数名
5. インスタンス変数名　　6. ローカル (局所) 変数名　　7. パッケージ名　　8. Java のキーワード

行番号	語
1	jp.ac.ryukoku.math.cards
3	Q01
4	static

行番号	語
4	main
9	add
17	c

(次ページに問題 II)

問題 Ⅱ　次の Java プログラムをコンパイルして実行したとする。このプログラムの 37 行目、41 行目、45 行目、49 行目、54 行目、59 行目で表向きになるカードを、例えば「スペードの A」のように、それぞれ解答用紙の解答欄に書きなさい。(24 点)

Q02.java

```
 1 import jp.ac.ryukoku.math.cards.*;
 2
 3 class C extends Card {
 4     Rank r = Rank.ACE;
 5
 6     C() {
 7         super(Suit.SPADES, Rank.DEUCE);
 8     }
 9
10     C(Rank r) {
11         super(Suit.HEARTS, r);
12     }
13
14     Rank get() {
15         return r;
16     }
17
18     void set(Rank rank) {
19     }
20 }
21
22 class D extends C {
23     D() {
24         super(Rank.THREE);
25     }
26
27     void set(Rank rank){
28         r = rank;
29     }
30 }
31
32 class Q02 {
33     public static void main(String[] args) {
34         GameFrame f = new GameFrame();
35         C c = new C();
36         f.add(c, 100, 100);
37         c.faceUp();
38
39         c = new C(Rank.FIVE);
40         f.add(c, 200, 100);
41         c.faceUp();
42
43         c = new C(c.get());
44         f.add(c, 300, 100);
45         c.faceUp();
46
47         c = new D();
48         f.add(c, 400, 100);
49         c.faceUp();
50
51         c.set(Rank.SIX);
52         c = new C(c.get());
53         f.add(c, 500, 100);
54         c.faceUp();
55
56         c.set(Rank.SEVEN);
57         c = new C(c.get());
58         f.add(c, 600, 100);
59         c.faceUp();
60     }
61 }
```

(次ページに問題 Ⅲ)

問題 III　RankStack クラスのインスタンスは、jp.ac.ryukoku.math.cards.Rank クラスのインスタンスを格納するためのスタックとして働くことができる。下表は、そのコンストラクタとインスタンスメソッドをまとめたものである。

RankStack クラス — Rank のスタックして働くオブジェクトのクラス

コンストラクタ	
RankStack(int size)	最大 size 個の Rank 型のデータを保持できるスタックとして働く (最初は空の) インスタンス
インスタンスメソッド	
void push(Rank r)	このスタックに r を積む (追加する)。ただし、r が null であったり、スタックが満杯であったりする場合は何もしない。
Rank pop()	最も近い過去に積まれた (push された) データをこのスタックから取り除き、そのデータを戻り値として返す。ただし、スタックが空であれば null を返す。
boolean isEmpty()	スタックが空であるかどうかを戻り値として返す。
boolean isFull()	スタックが満杯であるかどうかを戻り値として返す。

以下は、RankStack クラスのクラス宣言である。空欄の部分を解答用紙の解答欄に補いなさい。(31 点)

```
import jp.ac.ryukoku.math.cards.*;

class RankStack {
    private Rank[] stack;   // このスタックに積まれたデータを格納するための配列
    private int sp = 0;     // このスタックに積まれたデータの個数

}
```

(定期問題終り)

解 答 用 紙

問題 I (1) (a) `new Pile()`

(b) `new Deck()`

(c) `c.faceUp()`

(d) `null`

(e) `c.suit == last.suit || c.rank == last.rank`

(f) `c.moveTo(p)`

(2)

行番号	語	解答欄
1	`jp.ac.ryukoku.math.cards`	7
3	`Q01`	1
4	`static`	8

行番号	語	解答欄
4	`main`	3
9	`add`	2
17	`c`	6

問題 II 37 行目 スペードの 2

41 行目 ハートの 5

45 行目 ハートの A

49 行目 ハートの 3

54 行目 ハートの 6

59 行目 ハートの A

学籍番号　　氏名

(裏面に続く)

問題 **III**

```
import jp.ac.ryukoku.math.cards.*;

class RankStack {
    private Rank[] stack;
    private int sp = 0;
```

```
    RankStack(int size) {
        stack = new Rank[size];
    }

    void push(Rank r) {
        if (r != null && sp < stack.length) {
            stack[sp++] = r;
        }
    }

    Rank pop() {
        if (sp == 0) {
            return null;
        }
        return stack[--sp];
    }

    boolean isEmpty() {
        return sp == 0;
    }

    boolean isFull() {
        return sp >= stack.length;
    }
```

```
}
```

(解答用紙 終わり)