オブジェクト指向プログラミング
期末試験

問題冊子

「はじめ」の合図があるまで
問題冊子を開いてはいけません。

この問題冊子は試験終了後、回収します。

・この問題冊子は、（表紙を除いて）17ページあります。

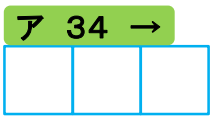
・この問題冊子は、11章に分かれており、各章に複数の問題があります。

・各章のプログラムは、他の章のプログラムには依存せず、
章ごとに独立して動作するものとします。

・問題冊子に記載されているプログラムは、
全てプログラミング言語「Java」のプログラムです。

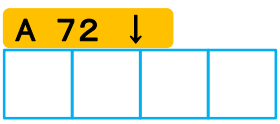
解答方法の例

例1）



カタカナの解答用紙に記入してください。
記入場所は、34番です。
記入方向は、横方向です。
記入する文字数は、3文字です。

例2）



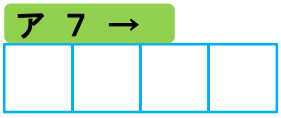
アルファベットの解答用紙に記入してください。
記入場所は、72番です。
記入方向は、縦方向です。
記入する文字数は、4文字です。

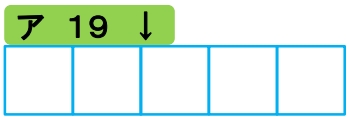
1 UML

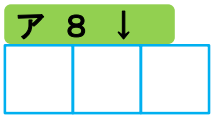
●UML

UMLの正式名称は？

日本語

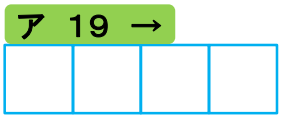






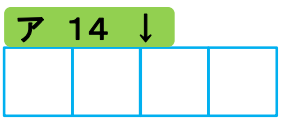
英語

●UMLの構成要素

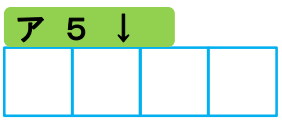
(1) ユースケース図は、システムの 

を表す。

(2) ストラクチャーダイアグラムは、

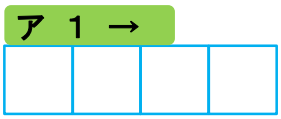


システムの

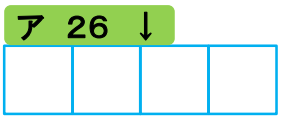


な を表す。

(3) ビヘイビアーダイアグラムは、



システムの



な を表す。

2 プログラミング

●継承と委譲の比較

	継承	委譲
ア 6 ↓	容易	やや複雑
ア 15 ↓	低い	高い

●デザインパターン

(1) デザインパターンは ア 8 ↓ に依存しない。

(2) どのデザインパターンにも ア 12 ↓ と ア 16 → がある。

(3) E.Gamma, Helm.R, R.Johnson, J.Vissides,の A 46 → 人が

著書「Design Patterns」で A 38 → 個のデザインパターンを提示した。

●プログラミングの原理

(1) オブジェクト指向プログラミングでは、

ア 21 → の ア 17 ↓ を ア 22 → に入れておく。

ア 21 → を管理するために、ア 3 → 的なプログラムになりやすい。

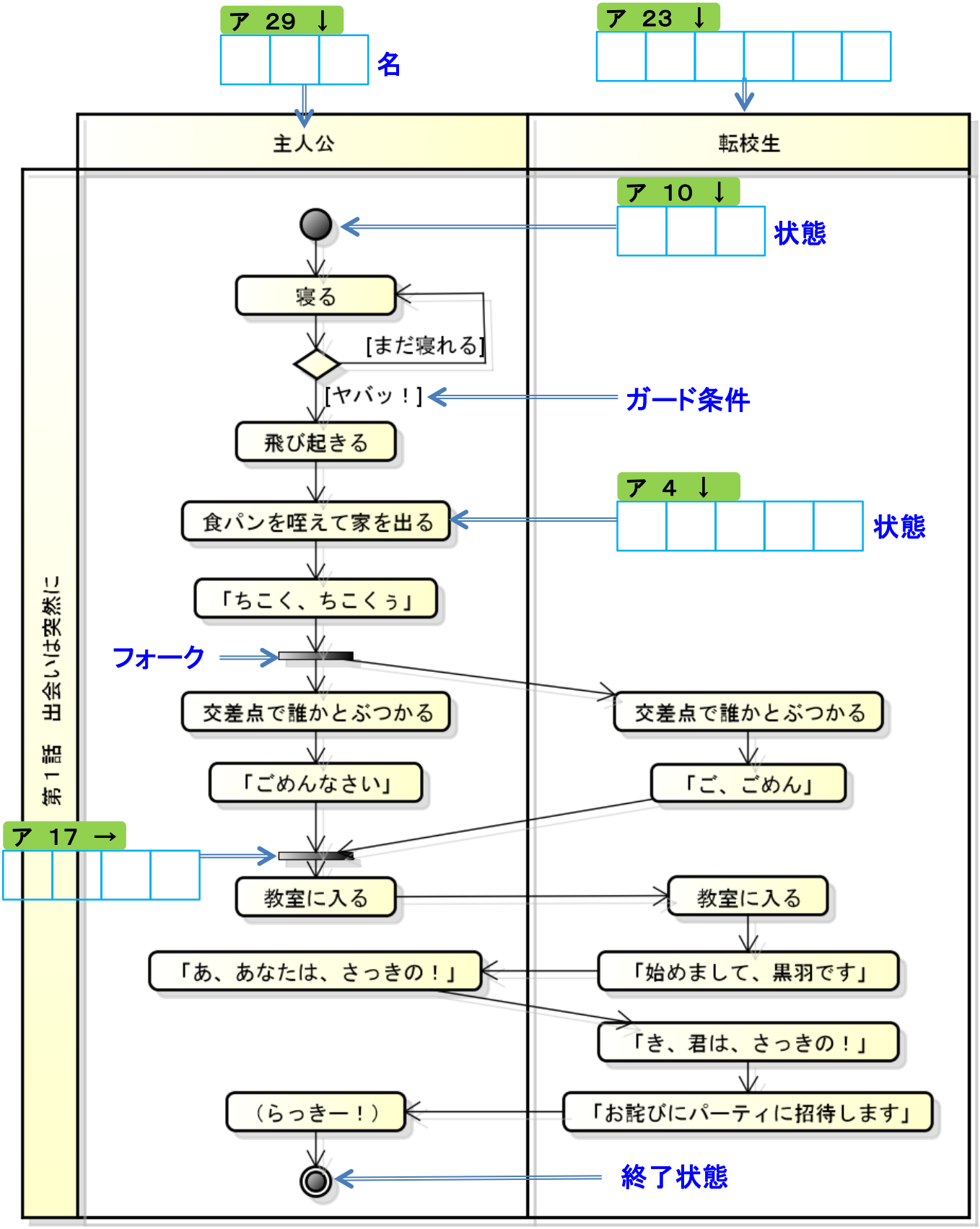
(2) 関数型プログラミングでは、

ア 24 → の ア 18 ↓ を ア 22 → に入れておく。

ア 24 → を重視するため、ア 11 → 的なプログラムになりやすい。

3 出会いは突然に

●アクティビティ図の構成要素の名称



4 事件発生

転校生に招待されて参加したパーティ会場で、
館に代々伝わる宝石の盗難事件が発生した！

経過

14:00　受付開始
14:30　----------(この時刻に監視カメラが切られていた)
14:50　受付完了
15:00　パーティ開始
15:30　----------館の主人が書斎に戻った
16:00　----------使用人が、館の主人を呼びに行ったところ
金庫が開けられ宝石が盗まれているのを発見

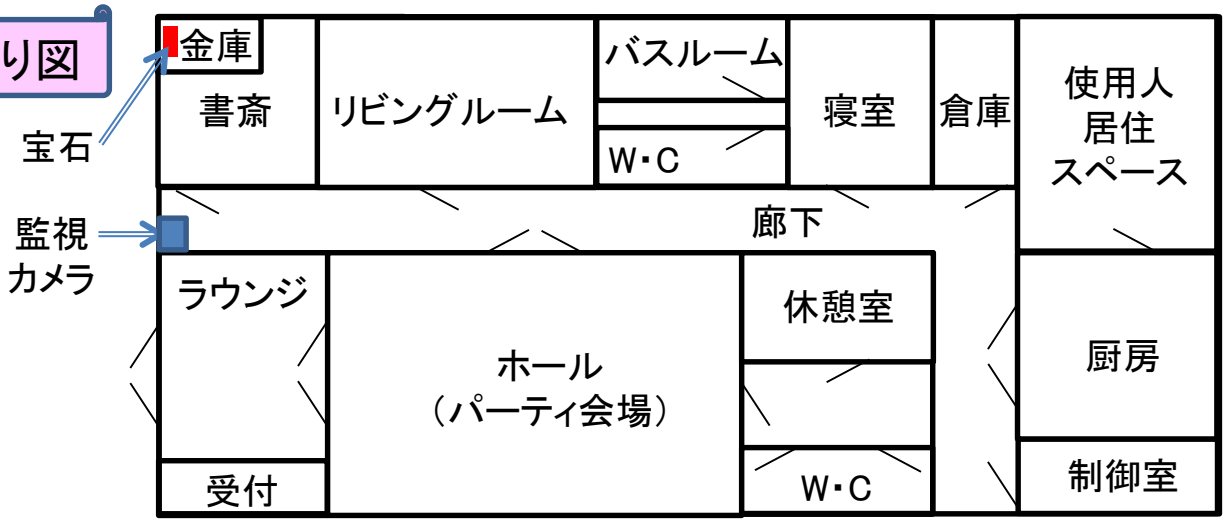
犯行の手口

①監視カメラは切り、
②館の主人を眠らせ、
③カギを使って金庫を開けた、
と考えられる。

状況

・監視カメラは、制御室でしかON/OFFできない。
・館の主人は、睡眠薬入りのワインで眠らされた。
・金庫のカギは、複製不可能で４つしかない。

見取り図



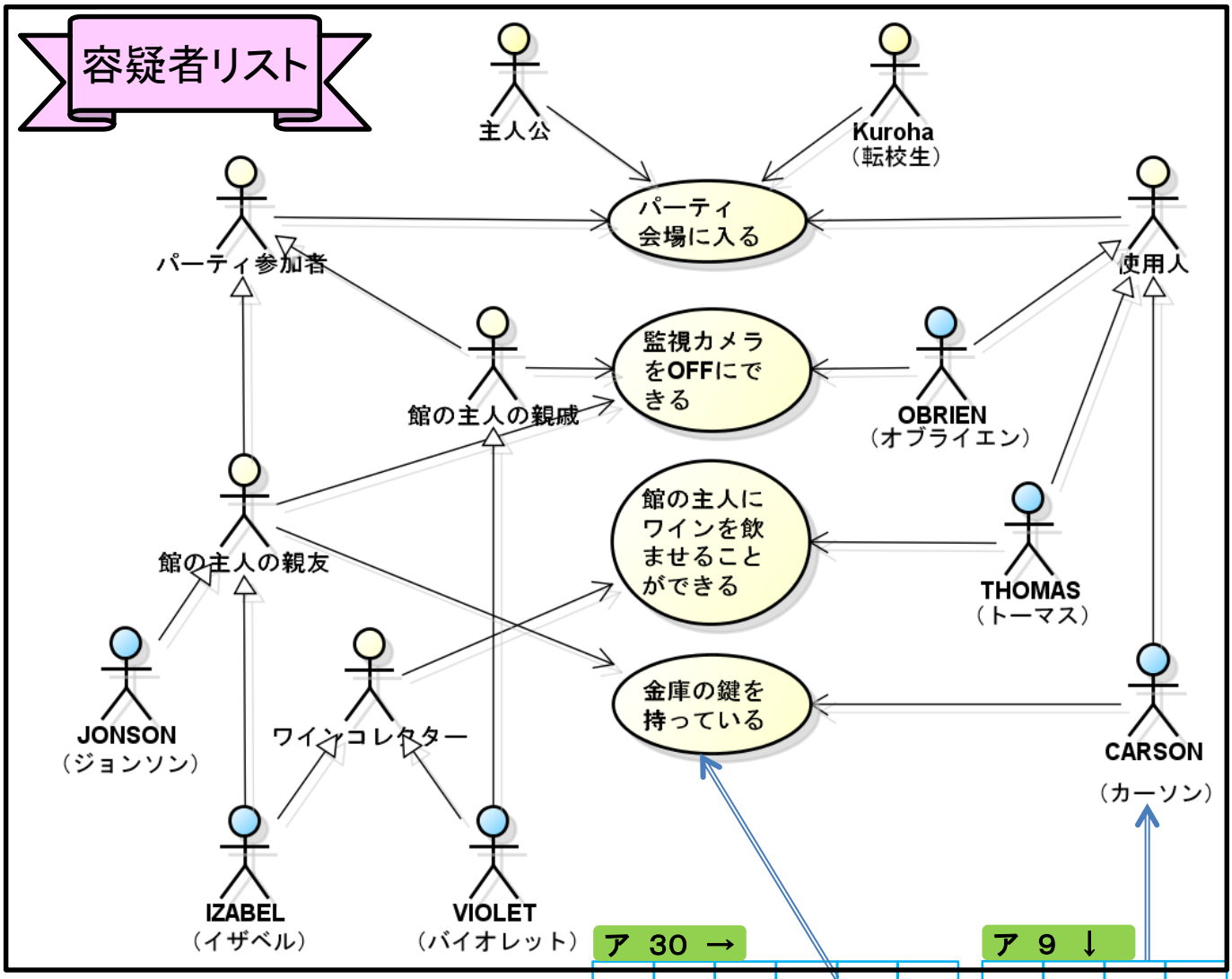
※上記の説明を覚えなくても、解答に支障はありません。

黒羽「さっ、さっそく、事件を解決しましょう。」
主人公「ちょっと待ってよ。それは警察の仕事でしょ。なんで私が？」
黒羽「あれ、できないんですか？」
主人公「しないんです。」
黒羽「招待してもらったのに、何か力になりたいとか思わないんですか？」
主人公「そういうことじゃないでしょ。」
黒羽「ふーん。」
主人公「だ、だったら、あなたがやればいいじゃない。」

5 聞き取り調査

黒羽「聞き取り調査の結果、以下の情報が得られました。」
主人公「黒羽君ってすごーい（棒）。」
黒羽「監視カメラ、ワイン、金庫の鍵の３つの条件を満たす人物は。。。」

●ユースケース図の構成要素の名称



●上のユースケース図からわかること

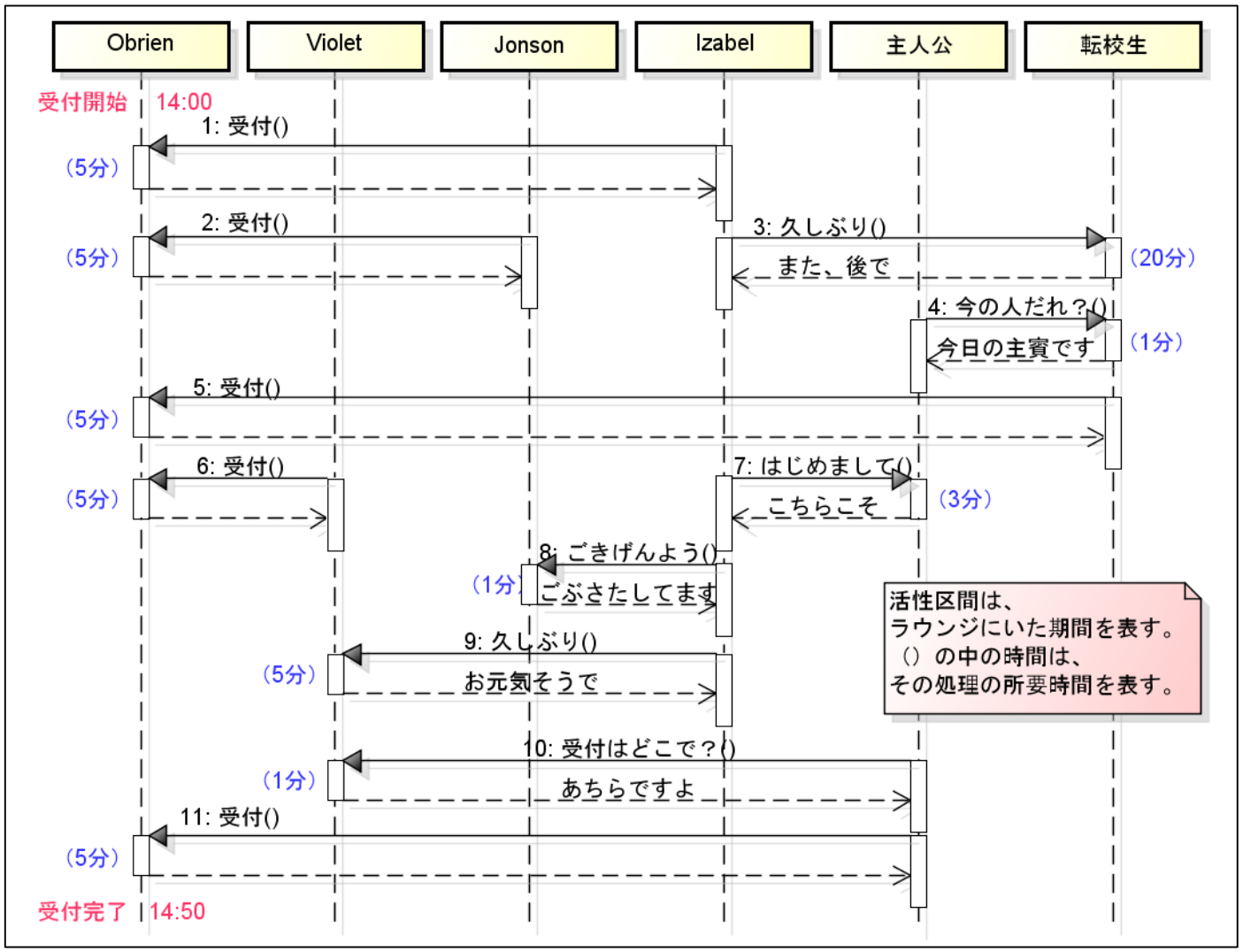
（1） A 16 ↓ は、単独で犯行が可能である。

（2） Thomas と A 1 ↓ は、どちらも単独では犯行は不可能だが、お互いに協力すれば犯行が可能である。

（3） Carson と A 10 → は、どちらも単独では犯行は不可能だが、お互いに協力すれば犯行が可能である。

6 ラウンジ

主人公「監視カメラが切られた14:30って、ラウンジで受付してた頃ね。」
黒羽「その時ラウンジにいた人物は、監視カメラをOFFにできませんね。」



●上のシーケンス図からわかること

(1) A 12 → [ ][ ][ ][ ][ ][ ] の受付開始時刻は、最も早くて14:05である。

(2) 主人公が「今の人だれ？」と言ったのは、最も早くて14: A 43 → [ ][ ] である。

(3) A 26 ↓ [ ][ ][ ][ ][ ][ ] の受付終了時刻は、最も遅くて14:39である。

(4) 転校生の受付開始時刻は、最も早くて14: A 42 → [ ][ ]、最も遅くて14: A 43 ↓ [ ][ ] である。

主人公「ということは A 37 ↓ [ ][ ][ ][ ][ ][ ] は、カメラをOFFにできないわね。」

7 監視カメラ（1／2）

黒羽「大変です。監視カメラに写ってましたよ！」
主人公「監視カメラは切られていたはずでは？」
黒羽「はい、切られてました。」
主人公「じゃ、写ってないでしょ。」
黒羽「バカですねー。」
主人公「ムッカー！」
黒羽「監視カメラが切られる直前の映像ですってば！」
主人公「切られる前なんだから、そりゃ何か写ってるでしょ？」
黒羽「バカですねー。」
主人公「ムッカー！！！！！！！！！！！！」
黒羽「監視カメラの設置場所と、制御室は離れてます。
したがって監視カメラを切る直前に監視カメラに映っていた人物には、
監視カメラを切ることはできないという事です。」
主人公「。。。で、誰が写ってたの。」

プログラム　プログラムが正しく動作するようにすること

```
public class SecurityCamera {
   public static void main(String[] args) {
      CameraPerson person;

      person = new CameraViolet();
      ask(person);

      person = new CameraIzabel();
      ask(person);

      person = new CameraJonson();
      ask(person);
   }

   public static void ask(  CameraPerson  person  ){
      System.out.print( "[“ + person.getName() + "]“ );
      System.out.print( person.areYouInPictures() );
      System.out.print( person.areYouCrime() );
      System.out.println( person.getComment() );
   }
}
```

7 監視カメラ（2／2）

プログラム（つづき） プログラムが正しく動作するようにすること

```
public class CameraPerson {
   public String getName( ){ return "一般人"; )
   public String areYouInPictures(){ return “写ってまーす。"; }
   public String areYouCrime( ){ return “私は無罪でーす。“ ; }
   public String getComment(){ return "いいかげんにして。"; }
}
```

A 22 →

```
public class CameraViolet  extends  Camera [      ] {
   public String getName( ){ return “Violet"; )
   public String getComment( ){ return “何か問題でも？"; }
}
```

A 5 ↓

```
public class CameraIzabel  extends  Camera [      ] {
   public String getName( ){ return “Izabel"; )
   public String areYouCrime( ){ return “何かの間違いです。”; }
}
```

A 8 ↓

```
public class CameraJonson extends Camera [      ] {
   public String getName( ){ return “Jonson"; )
   public String areYouInPictures( ){ return “写ってなーい。"; }
}
```

実行結果

ア 27 ↓

[Violet]写って[　　]。何かの間違いです。何か問題でも？

ア 28 ↓

[Izabel]写って[　　]。何かの間違いです。いいかげんにして。

ア 2 ↓

[Jonson]写って[　　]。何かの間違いです。何か問題でも？

A 36 ↓

主人公「ということは[　　]が、監視カメラを切ったのね。」

8 館の主人の証言（1／2）

黒羽「館の主人が意識を取り戻しました！」
主人公「それは良かったわね。」
黒羽「ええ、でも残念ながら何も覚えていないそうです。」
主人公「もしかして、私のことも、すべて忘れてしまったというの！」
黒羽「。。。。。」
主人公「言ってみただけ。」
黒羽「眠らされていたので、宝石が盗まれたことも知りませんでした。」
主人公「そう。しかたないわね。
でも誰にワインを飲まされたかがわかれば、共犯者がわかるはず！」
黒羽「だと良いんですが。。。」
主人公「そして、犯人を見つけて、宝石を取り戻して、
莫大な謝礼をもらって、さらに名探偵として取材が殺到して。。。」
黒羽「。。。。。」
主人公「言ってみただけ。」
黒羽「とりあえず、話を聞いてみましょう。」

プログラム　プログラムが正しく動作するようにすること

```
public class TestimonyOfMaster {
   public static void main(String[] args) {
      IWinePerson person;
                                  A 21 →
      person = new  Wine[      ]();

      System.out.print("パーティ会場で");
      System.out.print(person.getName());
      System.out.println("に勧められてワインを飲みました。");
                                  A 4 ↓
      person = new Wine[      ]();

      System.out.print("書斎で");
      System.out.print(person.getName());
      System.out.println("に勧められてワインを飲みました。");

      System.out.println(“今日は、その２人に勧められた時以外には、
                              ワインを飲んでいません。");
   }
}
```

8 館の主人の証言（2／2）

プログラム（つづき）　プログラムが正しく動作するようにすること

A 32 ↓

```
public [          ] IWinePerson {
   public   String  getName( );
}
```

A 29 ↓

```
public class WineViolet [          ] IWinePerson{
   public String getName( ){ return "Violet"; }
}
```

A 29 ↓

```
public class WineThomas [          ] IWinePerson{
   public String getName( ){ return "Thomas"; }
}
```

A 29 ↓

```
public class WineIzabel [          ] IWinePerson{
   public String getName( ){ return “Izabel"; )
}
```

実行結果

パーティ会場で、[A 21 →] に勧められてワインを飲みました。

書斎で、[A 4 ↓] に勧められてワインを飲みました。

今日は、その２人に勧められた時以外には、ワインを飲んでいません。

A 33 ↓

主人公「ということは、[　　] は、共犯者ではない？」

黒羽「たしかにワインは飲ませてませんが、別の方法で協力しているかも。」
主人公「もう少し、館の主人に話を聞けないかしら。」
黒羽「残念ながら、宝石の盗難を知って、また意識を失ってしまいました。」
主人公「そう。仕方ないかもね。宝石の盗難だけでも大変なのに、
身内が関わっているかもしれないとなるとショックだものね。」

9 鑑定結果（1／2）

黒羽「ワインに混入された睡眠薬が特定できました！」
主人公「あなた何者？」
黒羽「謎の転校生なんだから、それくらいできて当たり前です。」
主人公「。。。（自分で[謎の]とか言っちゃうかー）」
黒羽「結果は、以下の通りです。」

プログラム　プログラムが正しく動作するようにすること

```
public class SleepingPills {
   public  static  void  main( String[] args ) {
      Doctor doctor=new Doctor( );

      A 13 →
      [      ]  .setSleeper(  new   Wine( )  );
      A 13 →
      [      ]  .setSleeper(  new   Pill( )  );


      int  time =   doctor.howMuchTimeUntilSleeping( );

      System.out.print(“被害者の体質を考慮すると、");
      System.out.print("睡眠薬の入りのワインを飲んでから");
      System.out.print(“眠りにつくまで、");
      System.out.print(“最低でも”+time+“分はかかるでしょう。");
   }
}
```

実行結果

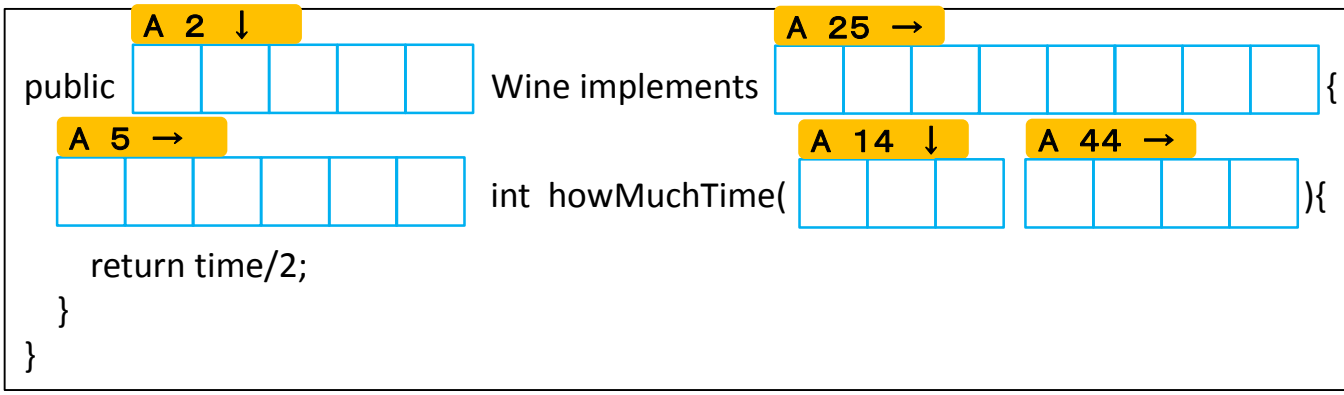
被害者の体質を考慮すると、睡眠薬の入りのワインを飲んでから眠りにつくまで、
最低でも A 39 ↓ [ ] 分はかかるでしょう。

主人公「パーティーは15:00に始まった。」
黒羽「突然、どうしました？」
主人公「館の主人は、パーティ開始後３０分で書斎に戻った。」
黒羽「もしかして、壊れちゃったんですか？」
主人公「そして、その３０分後、事件が発覚した。」
黒羽「おーい、戻ってこーい。」
主人公「ということはワインに睡眠薬を入れたのは A 34 ↓ [   ] !」
黒羽「！（なるほど）」

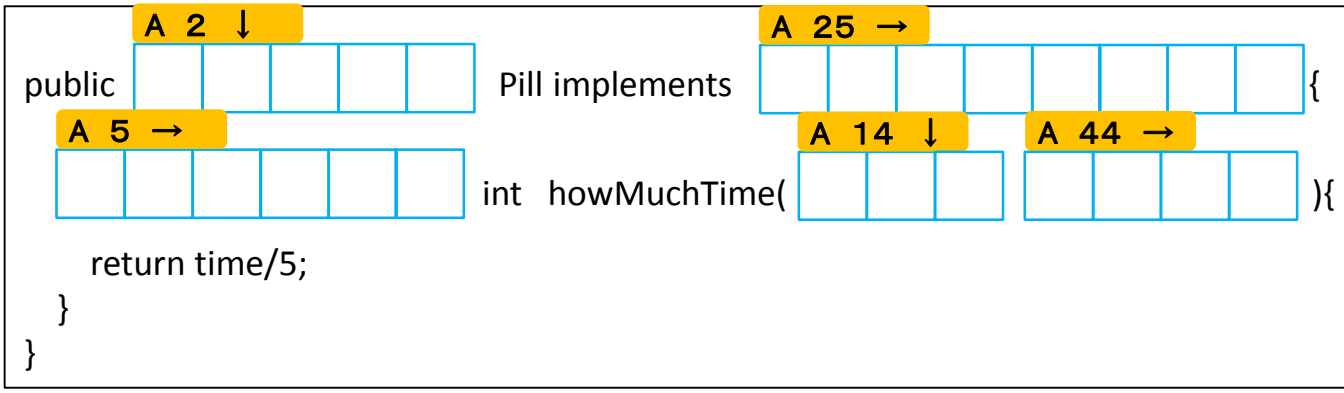
9 鑑定結果（2／2）

プログラム（つづき） プログラムが正しく動作するようにすること

```
public  interface Isleeper   {
   public  int  howMuchTime(  int  time );
}
```




```
      return time/2;
   }
}
```



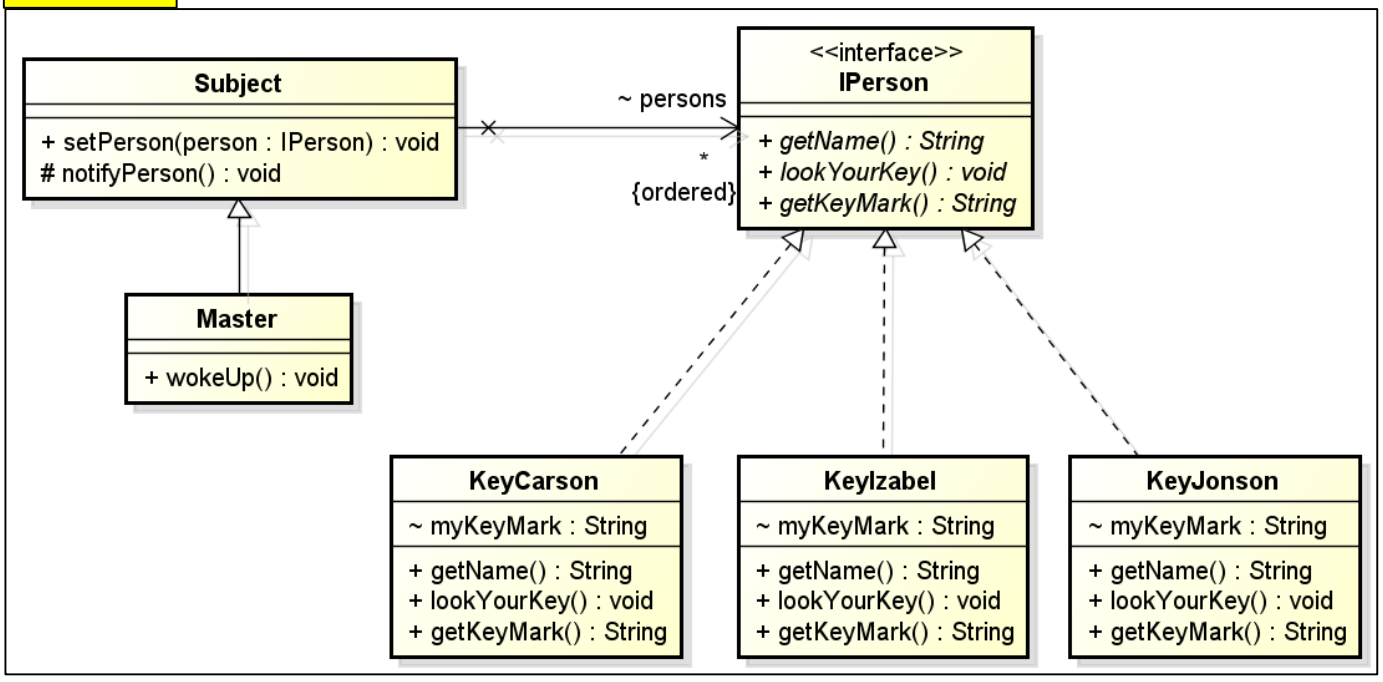

```
      return time/5;
   }
}
```

```
public class Doctor {
   LinkedList<ISleeper>  mySleepers = new LinkedList<ISleeper>( );

   public void setSleeper(ISleeper sleeper){
      mySleepers.add(sleeper);
   }

   public int  howMuchTimeUntilSleeping() {
      int  timeUntilSleeping = 360;
      for (ISleeper sleeper : mySleepers) {
         timeUntilSleeping = sleeper.howMuchTime(timeUntilSleeping);
      }
      return timeUntilSleeping;
   }
}
```

10 犯人の証（1／4）

黒羽「大変です！館の主人が着ていた
　　ガウンのポケットから金庫のカギが見つかりました。」
主人公「館の主人は金庫のカギを持っているんだから当然でしょ？」
黒羽「実は、館の主人は、自分が持っていた金庫のカギに
　　目印をつけていたのですが、
　　見つかったカギにはその目印がありませんでした！」
主人公「どゆこと？」
黒羽「犯人は、館の主人のカギと自分のカギを
　　間違えて持ち帰ってしまったのではないでしょうか？」
主人公「そんなことってあるのかな？」
黒羽「ま、推理小説なんかだと王道パターンだから有りです。」
主人公「だったら、金庫のカギを持っている３人に
　　カギを提出してもらえばいいじゃない！３人は今どこ？」
黒羽「容疑者は全員、休憩室にいます。」
主人公「じゃ、私が病院に行って館の主人が意識を取り戻したら連絡するから、
　　あなたは、館の主人からの指示だと言って、カギの目印を調べて！」

クラス図



●このプログラムの説明

A 19 ↓

このプログラムでは [　][　][　][　][　][　][　][　] パターンを利用している。

10 犯人の証（2／4）

プログラム プログラムが正しく動作するようにすること

```
public class MarkOfCriminal {
    public static void main(String[] args) {
        [A 17 ↓ □□□□□□] [A 17 ↓ □□□□□□]
                      = new [A 17 ↓ □□□□□□] ();
        [A 47 → □□□□□□□] person1=new Key[A 40 → □□□□□□] ();
        [A 47 → □□□□□□□] person2=new Key[A 7 → □□□□□□] ();
        [A 47 → □□□□□□□] person3=new Key[A 1 → □□□□□□] ();
        master.setPerson( person1 );
        master.setPerson( person2 );
        master.setPerson( person3 );

        [A 17 ↓ □□□□□□] . [A 11 ↓ □□□□□□] ();
        ask(person1);
        ask(person2);
        ask(person3);
    }
    private static void ask([A 47 → □□□□□□□] person ){
        System.out.print( "["+ person.getName( ) + "]“ );
        System.out.println( person.getKeyMark( ) );
    }
}
```

実行結果

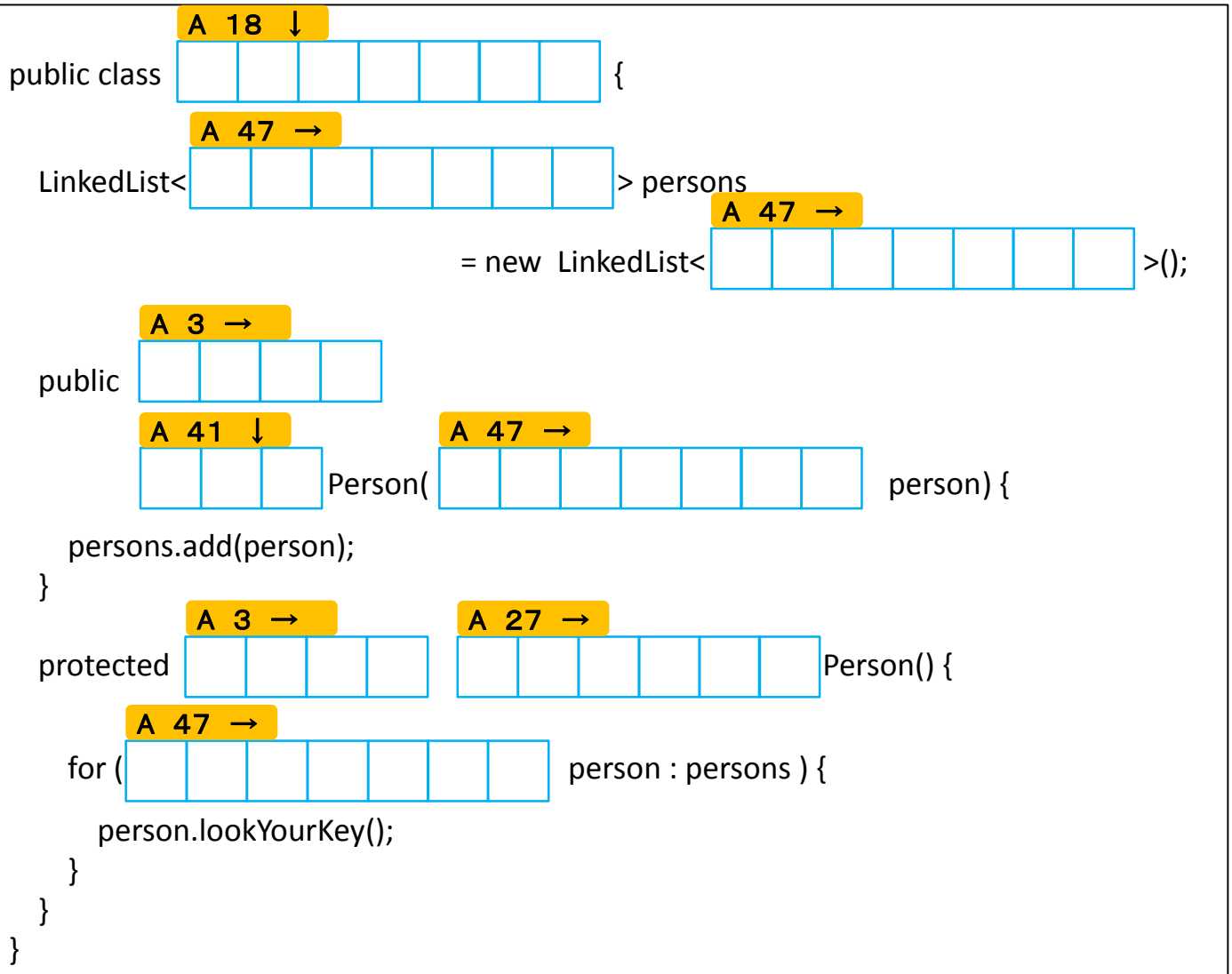
```
[Carson][A 28 → □□□□□□□□]
[Izabel][A 33 → □□□□□□□□]
[Jonson][A 45 → □□□□□□□□]
```

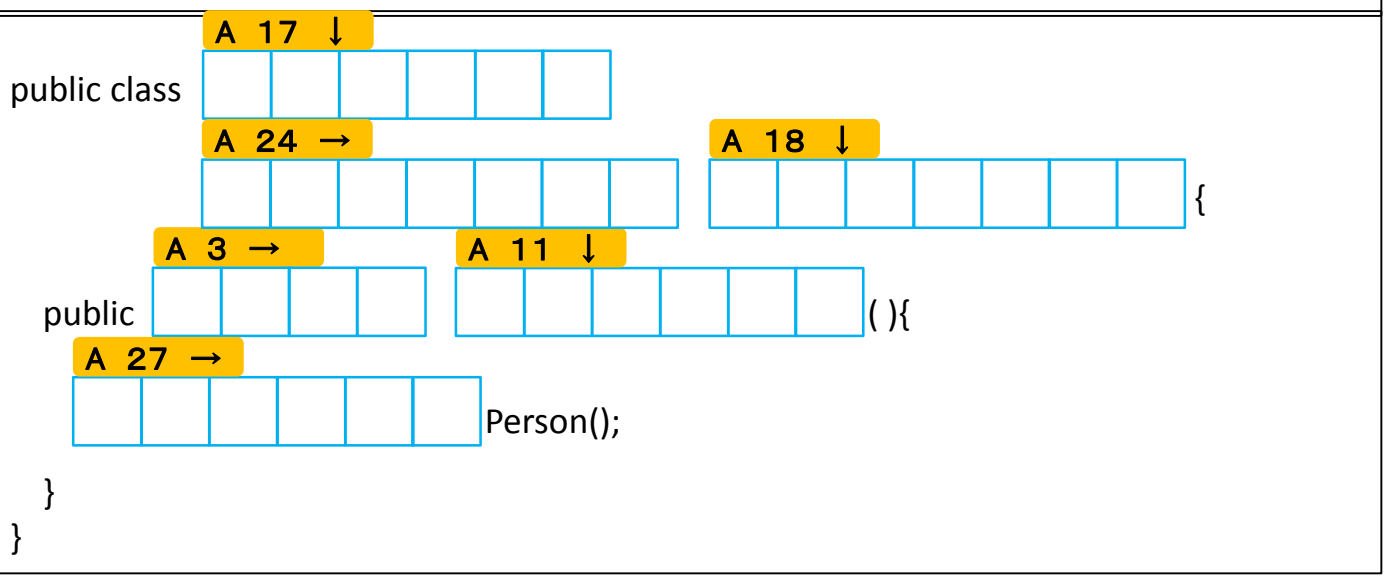
（注）鍵に目印があれば
CRIMINAL、
目印が無ければ
INNOCENT、
と表示されます。

10 犯人の証（3／4）

プログラム（つづき）

プログラムが正しく動作するようにすること
※ヒント：クラス図を参照すること





10 犯人の証（4／4）

プログラム（つづき）

プログラムが正しく動作するようにすること
※ヒント：クラス図を参照すること

```
A 47 →
public interface [        ] {
    A 6 →                    A 23 →
    public [        ]        [        ] ( );
    A 3 →                    A 20 ↓
    public [        ]        [        ] ( );
    A 6 →                    A 9 ↓
    public [        ]        [        ] ( );
}

A 47 →
public class KeyCarson implements [        ] {
    String myKeyMark;
    public String getName( )     { return "Carson"; }
                                 A 28 →
    public void   lookYourKey( ) { myKeyMark = “ [        ] "; }
    public String getKeyMark( )  { return myKeyMark; }
}

A 47 →
public class KeyIzabel implements [        ] {
    String myKeyMark;
    public String getName( )     { return “Izabel"; }
                                 A 33 →
    public void   lookYourKey( ) { myKeyMark = “ [        ] "; }
    public String getKeyMark( )  { return myKeyMark; }
}

A 47 →
public class KeyJonson implements [        ] {
    String myKeyMark;
    public String getName( )     { return “Jonson"; }
                                 A 45 →
    public void   lookYourKey ( ) { myKeyMark =“ [        ] "; }
    public String getKeyMark( )  { return myKeyMark; }
}
```

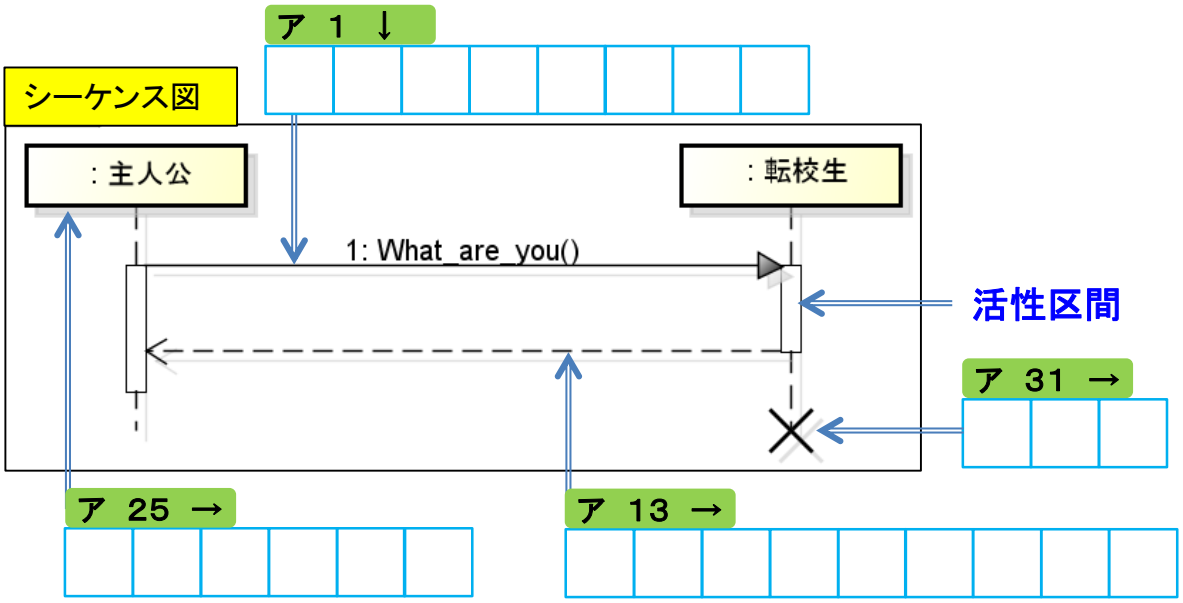
A 35 ↓→

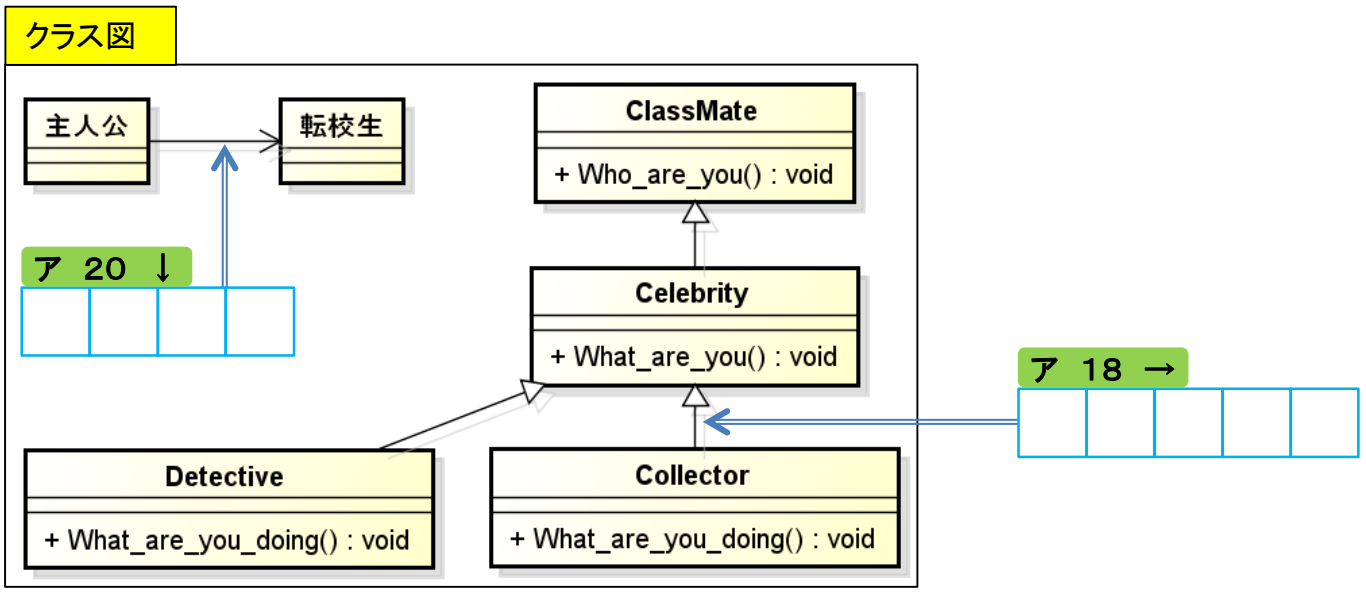
主人公「ということは [   ] が、宝を盗んだのね。」
黒羽「これで犯人が全員わかりましたね。おめでとうございます。」

11 転校生の正体

黒羽「犯人が逮捕されて、宝石もまもなく見つかるでしょう。」
主人公「そうね、これもあなたのおかげだわ。ありがとう。」
黒羽「単なる趣味ですから。」
主人公「それにしても、あなたいったい何者なの？」

●シーケンス図とクラス図の構成要素の名称





●上のシーケンス図とクラス図からわかること

上のクラス図では、「転校生」の親クラスに関する記述が抜けている。

「転校生」の親クラス（＝スーパークラス）は、

である。