2004 年度 環境情報学科 オブジェクト指向アプローチ論 平常試験 （7 月 5 日）担当：大城

● 60 分間　● 33 問× 3 点+1 = 100 点 ●持ち込み一切不可

(注) 以下の問題で，同じアルファベットの空欄は同じ内容である (例えば[m]と[m]の内容は必ず同じ)。しかし，アルファベットの異なる空欄は，同じ内容かもしれないし，異なる内容かもしれない (例えば[m]と[n]は同じ内容かもしれないし，異なる内容かもしれない)。

1. 計 5 問

次の文において，空欄 A〜E に入れる語として適切なものを，選択肢ア〜ソから選んで回答欄 A〜E に書け。

Java をはじめとする多くのオブジェクト指向言語では，事物の[A]を表現するために[B]を定義する。たとえば，“鳥”という[A]を表すために，Bird という名前の[B]を定義することができる。

[B]によって事物の[A]を表すとき，その事物の持つ「振る舞い」と「情報」を表現することが重要になる。Java の場合，「振る舞い」は[C]によって表現し，「情報」は[D]と呼ばれる変数によって表現する。[C]や[D]のように，[B]の定義の中で定義されたものを[E]と呼ぶ。

ア プロセス	**イ** メンバ	**ウ** 欠点	**エ** メソッド	**オ** セッタ
カ 関数	**キ** カテゴリ	**ク** パッケージ	**ケ** クラス	**コ** フィールド
サ 記述	**シ** ゲッタ	**ス** 概念	**セ** 長所	**ソ** 属性

2. 計 4 問

次の文において，空欄 F〜I に入れる語として適切なものを，選択肢ア〜シから選んで回答欄 F〜I に書け。

クラスは，ユーザ (プログラマ) が定義できるオリジナルの「データ型」である。一般に，「型」情報から生成された実体を，[F]と呼ぶ。たとえば，「int 型の変数 i は，int 型の[F]である」などと言う。特に，クラス型の[F]のことを，[G]と呼ぶ。

それぞれの[G]は，その[H]の内容によって，異なる状態を持つことができる。言い換えれば，[B]が事物の[A]を表しているのに対し，[G]は個々の事物を表しているのである。たとえば，[B]Dog は「犬」という[A]を表し，Dog 型の[G]Pochi や[G]Tarou は，それぞれ「ポチ」や「タロウ」といった個々の犬を表していると考えることができる。

一般に，プログラムが正しく動作するためには，[G]を生成するとき，[G]の[H]を初期化したり，特別な前処理を行う必要がある。そのために，Java では[G]を初期化するための特別な[C]を定義することができる。この特別な[C]のことを，[I]と呼ぶ。

端的に言って，オブジェクト指向とは，事物の[A]を[B]で表現して，その[F]としての[G]を生成し，それぞれの[G]がお互いにメッセージの交信をすることによってプログラムが動作する，という考え方である。

ア 正体	**イ** ファイナライザ	**ウ** デストラクタ	**エ** コンスタンス
オ 継承	**カ** インスタンス	**キ** コンストラクタ	**ク** オブジェクト
ケ 実数	**コ** サブジェクト	**サ** フィールド	**シ** メソッド

3. 計 3 問

次の実行可能な Java プログラムを完成させるために，空欄 J〜L 部分に必要な内容を回答欄 J〜L に書け (注：1 個の空欄に複数の単語や記号が入ることもある).

```
[ J ] SampleClass {
  public [ K ] ( [ L ] ) {
    System.out.println( "hello!" );
  }
}
```

4. 計 4 問

問題 3 のプログラムのソースファイル名はどのようなものにすべきか。ソースファイル名を回答欄 M に書け。また，問題 3 のソースファイルをコンパイルするには，どのようにコマンドを入力すればよいか。ソースファイル名を含む完全なコマンドを回答欄 N に書け。

問題 3 のソースファイルをコンパイルすると，クラスファイルが生成される。このクラスファイル名を完全な形で回答欄 O に書け。問題 3 から生成されたクラスファイルを実行するには，どのようなコマンドを入力すればよいか。回答欄 P に，クラス名を含む完全なコマンドを書け。

5. 計 4 問

次のプログラムを完成させるために必要な各空欄 Q〜T の内容を，それぞれ回答欄 Q〜T に書け。なお，プログラムの仕様については，プログラム内のコメントに書かれているので注意すること。

```
1  [ J ] Elevator { // エレベーターを表すクラス Elevator
2    // 以下の floor, n は，クラスの外部からアクセスできない
3    [ Q ] int floor; //現在の階を表す floor
4    [ Q ] int n;     //現在の総乗員数を表す n
5
6    //以下のコンストラクタとメソッドはどこからでもアクセス可能
7    //floor と n を初期化するコンストラクタ Elevator
8    [ R ] Elevator( int floor, int n ) {
9      [ S ].floor = floor; [ S ].n = n;
10   }
11   [ R ] int getFloor() { return floor; }
12   [ R ] void moveTo( int floor ) { [ S ].floor = floor; }
13   [ R ] int getN() { return n; }
14   [ R ] void setN( int n ) { [ S ].n = n; }
15   [ R ] void pickUp( int num ) {
16     n += num;
17     System.out.println( "現在の乗員数は"+n+"人です");
18   }
19   [ R ] [ K ] ( [ L ] ) {
20     //1 階で乗客の乗っていないエレベーターオブジェクト e を生成し，
21     //3 階に移動して，客を 2 人載せる
22     Elevator e = [ T ]
23     e.moveTo( 3 ); e.pickUp( 2 );
24   }
25 }
```

6. 計 8 問

次の文章の空欄 U〜Z，a〜b に当てはまる内容を，選択肢ア〜ソから選んで回答欄 U〜Z，a〜b に書け。

一般的に，ソフトウェアを作成する場合，関連するデータと機能をひとつにまとめて，[U]と呼ばれる再利用可能な「部品」にする。クラスは，最小の[U]であると言える。Java では，関連するクラスを集めて，[V]と呼ばれるより大きい[U]を作成することができる。このようにプログラムの構成要素を部品化 ([U]化) することで，プログラムの見通しが良くなるだけでなく，プログラムの設計もしやすくなる。また，[U]ごとに独立して開発することも可能になる。

[U]を作成する際に重要になるのは，[U]内の構成要素へのアクセスを適切に制御することである。[U]内の構成要素に対する外部からのアクセスを無制限に許せば，[U]の機能が正常に動作することは期待はできなくなる。なぜならば，外部から[U]内の構成要素に対して不正な操作が行われるかもしれないからである。

そのために，Java ではクラスのメンバへのアクセスを制御できるようになっている。たとえば，問題 5 のプログラムでも各メンバに対して，アクセスの指定を行っている。なお，一般に[D]は，クラス外部からアクセス[W]ようにすべきである，とされている。そのため，問題 5 の 11〜14 行目のように，必要ならば[D]へのアクセスを行う専用の[C]を用意する。このような[C]を[X]と呼ぶ。特に，[D]の値を設定する[X]を[Y]と呼び，[D]の値を返す[X]を[Z]と呼ぶ。

[U]の外部からアクセスできる構成要素は，その[U]の[a]部分になる。一方，その[U]の外部からアクセスできない構成要素は，その[U]の[b]部分となる。[a]部分は，外部から利用される部分であり，あまり変更はできない。変更すると，外部から[a]部分を利用している箇所を変更に合わせて書き換えなくてはならなくなり，これは大変な作業量になってしまうためである。それに対し，[b]部分は変更を自由に行うことができる。

ア グループ　**イ** できない　**ウ** インタフェイス　**エ** 隠蔽　**オ** ライブラリ
カ ボックス　**キ** アクセッサ　**ク** ポッパー　**ケ** 実装　**コ** パッケージ
サ ゲッタ　**シ** モジュール　**ス** トランスファー　**セ** できる　**ソ** セッタ

7. 計 3 問

次のプログラムを完成させるために必要な各空欄 c〜e の内容を，それぞれ回答欄 c〜e に書け。なお，プログラムの仕様については，プログラム内のコメントに書かれているので注意すること。

```
1    [ J ] Animal {  // 動物を表すスーパークラス Animal
2      double x, y;   // 居場所を表す x, y
3
4      Animal( double xpos, double ypos ) { // コンストラクタ
5        x = xpos; y = ypos;
6      }
7       void cry(){} // 鳴き声をあげる行為を表すメソッド cry()
8    }
9    [ J ] Cat [ c ] { // 猫を表す Animal のサブクラス Cat
10     Cat( double x, double y ) { [ d ]( x, y ); }
11     void cry() {
12       System.out.println( "にゃあ" ); // 「にゃあ」となく
13     }
14   }
15
16   [ J ] Dog [ c ] { // 犬を表す Animal のサブクラス Dog
17     Dog( double x, double y ) { [ d ]( x, y ); }
18     void cry() {
19       System.out.println( "わん" ); // 「わん」と吠える
20     }
21   }
22
23   [ J ] Zoo { // 動物園を表すクラス Zoo
24     // 動物に鳴いてもらうメソッド letItCry()
25     void letItCry( [ e ] a ) {
26       a.cry(); // オブジェクト a の cry() を呼んで鳴いてもらう
27     }
28     public [ K ] ( [ L ] ) {
29       Zoo aZoo = new Zoo(); // 動物園オブジェクトを生成
30       // 位置 (0, 0) に猫オブジェクトを生成
31       Cat aCat = new Cat( 0.0, 0.0 );
32       // 位置 (5, 5) に犬オブジェクトを生成
33       Dog aDog = new Dog( 5.0, 5.0 );
34       aZoo.letItCry( aCat ); // 猫に鳴いてもらう
35       aZoo.letItCry( aDog ); // 犬に鳴いてもらう
36     }
37   }
```

8. 計 2 問

次の文章の空欄 f〜g に当てはまる内容を，選択肢ア〜コから選んで回答欄 f〜g に書け。

オブジェクト指向で最も重要な考え方は，[f]関係を表現する「継承」である。「B is a A」とは，「B は A の一種」という関係を表す。たとえば，「猫 is a 動物」は「猫は動物の一種」という意味であり，同様に「犬 is a 動物」は「犬は動物の一種」という意味である。問題 7 のプログラムでは，「猫 is a 動物」という関係を表すために，動物を表す Animal というスーパークラスを作り，そのサブクラスとして猫を表す Cat を定義している。

なぜ，[f]関係を表すのに継承が使われるのだろうか。B が A のサブクラスである場合，サブクラス B は，A が持っているメンバをすべて継承する。その結果，サブクラス B は A が持っている機能をすべて持っていることになる。つまり，サブクラス B は，スーパークラス A の「一種である」と考えることができるのである。

したがって，Java では「サブクラス is a スーパークラス」という[f]関係が成立していると考えて，サブクラスのオブジェクトは，スーパークラスのオブジェクトとして扱うことが可能になっているのである。

サブクラスでは，スーパークラスから継承したメンバだけでなく，サブクラス独自のメンバを追加することもできる。また，サブクラスでメソッドを[g]することで，スーパークラスとは異なった独自の振る舞いをサブクラスにさせることができる。

重要なのは，ソフトウェアの多くの部分を，スーパークラスで記述することである。スーパークラスで記述された部分は，サブクラスのオブジェクトを一括して扱うことができ，サブクラスが増えたとしても変更する必要がない。

問題 7 のプログラム例では，25〜27 行目のクラス Zoo の letItCry() メソッドは，スーパークラス Animal だけで記述されていて，Cat 型オブジェクトや Dog 型オブジェクトを扱うことができる。また，このプログラムで新たに「馬」を扱う必要がでてきた場合は，Animal のサブクラスとして「馬」を表す Horse を新しく定義すればよい。こうして，Animal のサブクラスが新しく定義されても，スーパークラス Animal を使って記述された letItCry() メソッドを変更する必要はないのである。

ア アンダーフロー　**イ** 相互　**ウ** friend　**エ** is-a　**オ** オーバロード
カ オーバーフロー　**キ** 依存　**ク** is-implemented-with　**ケ** has-a　**コ** オーバライド

学生番号 | | | | | | |　　氏　名＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Y

Z

a

b

c

d

e

f

g