2003 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 22 日）担当：大城

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

1. 計 4 問

次の文において，空欄 A〜D に入れる語として適切なものを，選択肢ア〜コから選んで回答欄 A〜D 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，[A] を定義することによって，事物の概念をプログラム的に表現する。たとえば，“車”という概念を表すために，Car という名前の [A] を定義することができる。[A] 定義の中では様々なものを定義でき，それを [B] と呼ぶ。
>
> クラスで事物の概念を表すとき，その事物の持つ「情報」と「振る舞い」を表現することが重要になる。Java の場合，「情報」は [C] と呼ばれる変数 [B] によって表現し，「振る舞い」は [D] と呼ばれる [B] によって表現する。

ア 関数　イ カテゴリ　ウ パッケージ　エ クラス　オ フィールド
カ プロセス　キ メンバ　ク ビヘイビア　ケ メソッド　コ アトリビュート

2. 計 4 問

次の文において，空欄 E〜H に入れる語として適切なものを，選択肢ア〜コから選んで回答欄 E〜H 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，[E] と呼ぶ。たとえば，「int 型の変数 i は，int 型の [E] である」などと言う。特に，クラス型の [E] のことを，[F] と呼ぶ。
>
> それぞれの [F] は，その [C] 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，[A] が事物の概念を表しているのに対し，[F] は個々の事物を表しているのである。たとえば，[A] Cat は「猫」という概念を表し，Cat 型の [F] Tama や [F] Mike は，それぞれ「タマ」や「ミケ」といった個々の猫を表していることになる。
>
> 一般に，プログラムが正しく動作するためには，[F] を生成するとき，[F] の [C] 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 [F] を初期化するための特別な [D] を定義することができる。この特別な [D] のことを，[G] と呼ぶ。
>
> 端的に言って，オブジェクト指向とは，事物の概念を [A] で表現して，その [E] としての [F] を生成し，それぞれの [F] 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。また，オブジェクト指向の重要な考えに，[A] の [H] がある。

ア 実数　イ 正体　ウ コンストラクタ　エ デストラクタ
オ ファイナライザ　カ 継承　キ オブジェクト　ク サブジェクト
ケ インスタンス　コ コンスタンス

3. 計 4 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 I〜L 部分に必要な内容を回答欄 I〜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).

```
[I] SimpleClass {
  public [J] void [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，クラスファイル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  class Human { //人間を表すクラス Human
2    // 以下の age，gender は，クラスの外部からアクセスできない
3    [Q] int age;    //年齢を表す age
4    [Q] int gender;//性別を表す gender(0:男性，1:女性)
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    //フィールド age と gender を初期化するコンストラクタ Human
8    [R] Human( int age, int gender ) {
9      [S].age = age;  [S].gender = gender;
10   }
11   [R] int getAge() { return age; }
12   [R] void setAge( int age ) { [S] .age = age; }
13   [R] int getGender() { return gender; }
14   [R] void talk() {
15     System.out.println( "I'm "+age+" years old.");
16   }
17   [R] [J] void [K] ( [L] ) {
18     Human Taro = [T] //17 歳男性を表す Taro オブジェクトを生成
19     Taro.talk();  // talk() メソッドを呼び出す
20   }
21 }
```

※ gender フィールドの値は，0 のとき男性を，1 のとき女性を表すことに注意。

6. 計 4 問

次の文章の空欄 U〜X に当てはまる内容を，選択肢ア〜コから選んで回答欄 U〜X 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。
>
> [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 のプログラムでは，フィールドはクラス外からアクセスできないように指定されているのに対し，コンストラクタやメソッドはクラス外部からアクセスできるように指定されている。
>
> なお，一般にフィールドは，クラス外部からアクセスできないようにすべきである，とされている。そのかわり，問題 5 の 11〜13 行目のように，フィールドへのアクセスを行う専用のメソッドを用意する。このようなメソッドを [W] 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 [X] 部分になる。一方，[U] 外部からアクセスできない構成要素は，その [U] の実装部分となる。[X] 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 [X] 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，実装部分は変更を自由に行うことができる。
>
> [U] の [X] 部分を最少にすることで，[U] の部品としての独立性は高まり，各 [U] の改良や差し替えなども柔軟に行うことが可能になるし，前述したように，プログラム全体の構造も見通しが良くなるのである。

ア ボックス　イ アクセッサ　ウ インプリメンテーション　エ ゲッタ　オ セッタ
カ パッケージ　キ モジュール　ク インタフェイス　ケ 隠蔽　コ ライブラリ

7. 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 Y,Z,a,b の内容を，それぞれ回答欄 Y,Z,a,b 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   [ Y ] class Animal { //動物を表す抽象クラス 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     [ Y ] void cry(); // 鳴き声をあげる行為を表す抽象メソッド
8   }
9   class Cat [ Z ] Animal { // 猫を表す Animal のサブクラス Cat
10    Cat( double x, double y ) { [ a ]( x, y ); }
11    void cry() {
12      System.out.println( "にゃあ" ); // 「にゃあ」となく
13    }
14  }
15
16  class Dog [ Z ] Animal { // 犬を表す Animal のサブクラス Cat
17    Cat( double x, double y ) { [ a ]( x, y ); }
18    void cry() {
19      System.out.println( "わん" ); // 「わん」と吠える
20    }
21  }
22
23  class 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( [ b ] theAnimal ) {
26      theAnimal.cry();//動物オブジェクトの cry() を呼んで鳴いてもらう
27    }
28    [ R ] [ J ] void [ 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 5 問

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

オブジェクト指向で最も重要な考え方は，[c]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[d]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，[c]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，[d]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 [d]」という[c]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，[d]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，[d]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[e]することで，[d]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，[d]で記述することである。[d]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，[d]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，[d]Animal を使って記述された letItCry() メソッドは変更する必要がない。

なお，多くの場合，[d]は，メソッドの詳細な定義が不可能なことが多い。これは，たいていの[d]が，抽象度の高い[f]を表しているためである。問題 7 の例では，「動物」は[f]であり，その具体的な振る舞い (メソッド) を定義することは難しい。たとえば，「動物」がどのように「鳴く」かは，具体的に定義できない。一方，「猫」や「犬」という概念は，「“にゃあ”と鳴く」，「“わん”と鳴く」というように，具体的な振る舞いを定義できる。

前述のように，[f]を表すクラスは，振る舞い (メソッド) を適切に定義できない。その場しのぎで何かしらの定義を行っても，そのクラスのオブジェクトが生成されて，メソッドが呼ばれた場合に，不都合が起きる可能性がある。そこで，[g]クラスという[f]を表すクラスが用意されている。[g]クラスのオブジェクト生成は，禁止される。また，[g]クラスでは，[g]メソッドという定義部分のない特別なメソッドを宣言できるようになっている。[g]メソッドは，サブクラスで[e]することによって，定義を与えることができる。

ア 具象　イ 一般概念　ウ スーパークラス　エ has-a 関係　オ オーバライド
カ 抽象　キ 特殊概念　ク 内部クラス　ケ is-a 関係　コ オーバロード