2001-02-08

2000 年度計算機言語 II 定期試験

準備　この講義の期間中、自分がおもに Java を動作させた環境で java -version というコマンドを実行させた結果を書け。(採点は行わないが、以下の設問の採点に影響してくる場合がある)

問題 1　オブジェクト指向言語の継承についての以下の説明のどこが間違っているか指摘せよ。

> オブジェクト指向プログラミングでは、継承によってほかのオブジェクトから任意の性質を引き継いでくることができるので、書くべきコードの量が少なくなる。たとえば、原点を始点とする 2 次元平面上のベクトル (x, y) (x, y は実数) を表すクラス TwoDimenVector を設計する場合を考えよう。このとき、2 つのベクトルの和・差はそれぞれの座標の和・差をとればよいから、実数を表すクラスから和・差のメソッドを継承すれば TwoDimenVector で和・差のメソッドを実装する必要はない。

問題 2　以下の問いに答えよ。

1. スタックを表すインタフェース Stack の実現として、AStack と LStack とを実装した (別紙)。これらの実装は、スタックという抽象データ型の実現という観点から見てまったく同じ振る舞いをするといえるか。異なるところがある場合はその点を指摘し、どちらがよい実装であるかを理由を添えて述べよ。
2. JDK1.2 から、連結リストを実装するクラス LinkedList (パッケージ名 java.util) が用意されている。このクラスの主要メソッドは以下の通りである。これを用いて、Stack インタフェースを実現するクラス LinkedStack を実装せよ。エラー処理は考えなくてよい。

メソッド名	機能
void addFirst(Object o)	o を連結リストの先頭に追加する
void addLast(Object o)	o を連結リストの最後に追加する
Object getFirst()	連結リストの先頭の要素を返す。リストからは取り除かない。
Object getLast()	連結リストの最後の要素を返す。リストからは取り除かない。
Object removeFirst()	連結リストの先頭の要素をリストから削除し、返す。
Object removeLast()	連結リストの最後の要素をリストから削除し、返す。
int size()	連結リスト中の要素数を返す。

問題 3　以下の問いのうちいずれか一問を選択して答えよ。

1. WWW でプログラムを実行させる方法として、アプレット以外に CGI(Common Gateway Interface) という手法がある。これと Java アプレットとを、プログラムの実行の仕組みという観点から対比して説明し、実現可能な機能やセキュリティなどの視点から比較せよ。
2. プログラミング言語には、シンボル (名前) の衝突を防いだり、変数・関数を隠蔽する機能が用意されている。たとえば C 言語では、局所変数や static 宣言などがそれにあたる。Java 言語で用意されている情報隠蔽機能・シンボル衝突回避機能について列挙し、簡潔かつ具体的に説明せよ。

問題 4　実際に Java 言語の処理系を動かしてみたとき生じた問題のうち、Java 言語特有のものをいくつか挙げ、その原因を述べよ。

```
public interface Stack {
        public void clear(); // スタックをクリア
        public void push(Object it); // it をスタックに push
        public Object pop(); // スタックから pop
        public Object topValue(); // スタックのいちばん上の要素を返す
        public boolean isEmpty(); // スタックが空なら true を返す
}

public class AStack implements Stack {    // Array based stack class
        private static final int defaultSize = 10;
        private int size;               // Maximum size of stack
        private int top;                // Index for top Object
        private Object [] listarray;    // Array holding stack Objects

        AStack() { setup(defaultSize); }
        AStack(int sz) { setup(sz); }
        public void setup(int sz) {
          size = sz;  top = 0; listarray = new Object[sz];
        }
        public void clear() {           // Remove all Objects from stack
          top = 0;
        }
        public void push(Object it) {   // Push Object onto stack
          listarray[top++] = it;
        }
        public Object pop() {           // Pop Object from top of stack
          return listarray[--top];
        }
        public Object topValue() {      // Return value of top Object
          return listarray[top-1];
        }
        public boolean isEmpty() {      // Return true if stack is empty
          return top == 0;
        }
}

public class LStack implements Stack {       // Linked stack class
        private Link top;                   // Pointer to list header
        public LStack() { setup(); }        // Constructor
        public LStack(int sz) { setup(); }  // Constructor: ignore sz
        private void setup() { top = null; } // Initialize stack
        public void clear() { top = null; } // Remove Objects from stack
        public void push(Object it) {        // Push Object onto stack
          top = new Link(it, top);
        }
        public Object pop() {               // Pop Object at top of stack
          Object it = top.element();
          top = top.next();
          return it;
        }
        public Object topValue()  {         // Get value of top Object
          return top.element();
        }
        public boolean isEmpty() {          // Return true if empty stack
          return top == null;
        }
}
```