計算機ネットワーク工学 I・テスト問題用紙

（’03 年 7 月 31 日・ 13:00 〜 14:30）

解答上、その他の注意事項

I. 問題は、問 I〜VI まである。

II. 解答用紙の右上の欄に学籍番号・名前を記入すること。

III. 解答欄を間違えないよう注意すること。

IV. 解答欄がマス目になっている場合は、1 字に 1 マスを用いること。特に空白にも必ず 1 マスを用いること

V. 解答中の文字 (特に a と d) がはっきりと区別できるよう注意すること。

VI. ノート・プリント・参考書などは持ち込み可である。プリントは 1 冊（やむを得ない場合は 2 冊）にまとめること。

VII. 合格は 100 点満点中 60 点以上とする。
(配点 — 期末テスト 70 点, レポート各 15 点)

I. 次の文章の空欄を埋めるのにもっとも適切な言葉を指示された選択肢から選べ。

(i) Java で TCP や UDP による通信を行なうプログラムを記述するためには、いくつかの種類のソケットを用いる。通信要求を（受動的に）待ち受ける側をサーバ、最初に通信を（能動的に）要求する側をクライアントと呼ぶことにすると、ソケットを生成するために通常使用するコンストラクタは次の表のようになる。（括弧の中は対応する位置の引数の型を示す。）

	クライアント	サーバ
TCP（コネクション型）	(1)	(2)
UDP（コネクションレス型）	(3)	(4)

また、DatagramSocket(int) という型のコンストラクタの場合、この引数の意味は (5) である。

(1)〜(4) の選択肢

(A) DatagramSocket()　　(B) Socket(String, int)
(C) ServerSocket(int, int)　　(D) DatagramSocket(int)

(5) の選択肢

(A) 自分のホスト名（IP アドレス）　　(B) 自分のポート番号
(C) 接続相手のホスト名（IP アドレス）　　(D) 接続相手のポート番号

(ii)

「1+2+3+4+5」のように「+」記号で結ばれた数字からなる文字列から数を 1 つずつ取り出しながら、足し算の途中結果を右のように表示したい。

```
0+1=1
1+2=3
3+3=6
6+4=10
10+5=15
```

StringTokenizer クラスを用いて、このプログラムを作成すると次のようになる。ただし、入力の文字列はコマンドラインの第 1 引数（args[0]）として与えることにする。

```
import java.util.*;

public class TokenizerTest {
    public static void main(String[] args) {
        StringTokenizer st = new StringTokenizer(args[0], "+");
        int sum=0;
        while( (6) ()) {
            int n = (7) ( (8) ());
            int sum1=sum+n;
            System.out.println(""+sum+"+"+n+"="+sum1);
            sum=sum1;
        }
    }
}
```

(6)〜(8) の選択肢

(A) Integer.parseInt　(B) getParameter　(C) st.hasMoreTokens
(D) st.nextToken　(E) st.countTokens　(F) g.drawString

II. Java はゴミ集めの機能をもち、C や C++は標準ではゴミ集めの機能を持っていない。ゴミ集めについて簡単に説明し、ゴミ集めを採用することの長所と短所について簡単に述べよ。

III. 次のプログラムは、画面上にある図形を描くアプレットである。

```
import java.awt.*;
import java.applet.*;

public class Hatena extends Applet {
    public void paint(Graphics g) {
        int i;
        int x=200, y=200, dx=1, dy=0;
        for(i=0; i<15; i++) {
            int dx1, dy1;
            g.drawLine(x, y, x+dx, y+dy);
            x += dx;      y += dy;
            dx1 =  dx+dy; dy1 = -dx+dy;
            dx  =  dx1;   dy = dy1;
            // here
        }
    }
}
```

変数 i の値がそれぞれ、(1). 1 と (2). 4 の時、ループの最後の地点（「// here」で示されている地点）での変数 x と y の値はいくらか？

IV. 次のプログラムはバブルソートというアルゴリズムの動作をアニメーションで示すアプレットである。

```
import java.applet.*;
import java.awt.*;

public class BubbleSort extends Applet implements Runnable {
  int unit=20
  int[] args = { 5, 4, 3, 2, 1};
  int n = args.length;
  Thread thread=null;
  int i=0, j=n-1;

  public void start() {
    if (thread == null) {
      thread = new Thread(this);
      thread.start();
    }
  }

  public void stop() {
    thread = null;
  }

  public void paint(Graphics g) {
    int k;

    for(k=0; k<args.length; k++) {
      g.fillRect(0, k*unit, args[k]*unit, unit);
    }
  }

  public void run() {
      for (i=0; thread!=null && i<n-1; i++) {
          for (j=n-1; j>i; j--) {
              try {
                  Thread.sleep(1000);
              } catch (InterruptedException e) {
                  return;
              }
              if (args[j-1]>args[j]) { // スワップする。
                  int tmp=args[j-1];
                  args[j-1]=args[j];
                  args[j]=tmp;
              }
              repaint();
          }
      }
  }
}
```

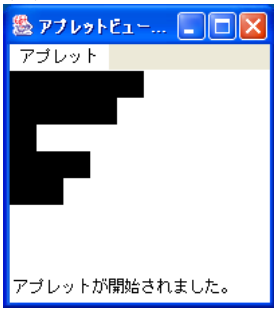
次に、このプログラムのスクリーンショットらしき画像をいくつか示す。このなかで

(1) i==1 && j==3
(2) i==2 && j==3

の時のものを次の選択肢の中から選べ。(ただし、選択肢の中には、このプログラムのスクリーンショットでないものも混じっている。)

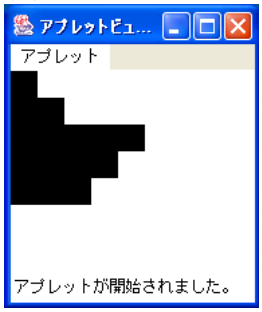
選択肢

(A)

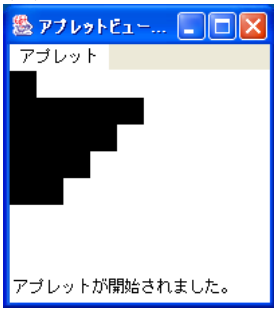


(B)

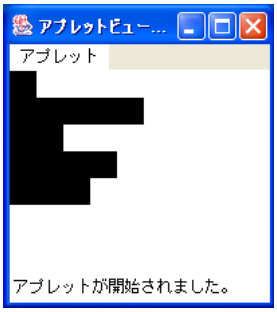
(C)



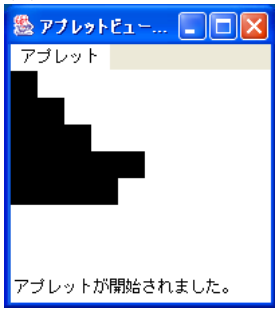
(D)



(E)

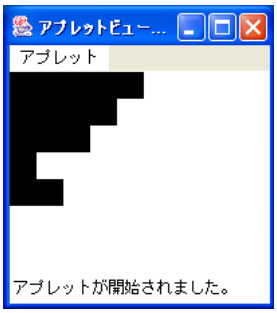


(F)

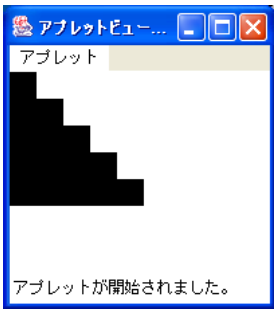


参考（はじめとおわりの状態）

はじめ



おわり



V. 次の Point クラスを継承して、

```
class Point {
   public int x, y;
   public Point(int x0, int y0) {
     x = x0; y = y0;
   }
   public void move(int dx, int dy) {
      x += dx;
      y += dy;
   }

   public void print() {
     System.out.print("(" + x + ", " + y + ")");
   }
}
```

GridPoint クラスを定義する。GridPoint クラスは、1 つの新しい int 型のフィールド grid を持つ。print メソッドは再定義されていて、x, y の正確な値の代わりに、それよりも小さい grid の倍数のうち、正確な値にもっとも近いものを表示する。例えば grid が 4 で x が 9, かつ y が 15 の時、(8, 12) と表示する。(x, y が負の数の場合は考慮する必要はない。)

(1)〜(3) の空欄を埋めてクラスの定義を完成させよ。

```
class [      (1)      ] {
   public [      (2)      ]

   public GridPoint(int x0, int y0, int g0) {
      super(x0, y0);
      grid=g0;
   }
   public void print() {
      [                (3)                ]
   }
}
```

さらに以下の文章の (4), (5) の空欄を埋めよ。

「もし次のようなテスト用プログラムがあるとすると、

```
  ...
  Point[] pts = new Point[2];
  pts[0] = new Point(1, 2);
  pts[1] = new GridPoint(9, 15, 4);
  int i;
  for (i=0; i<pts.length; i++) {
    pts[i].print();
  }
  ...
```

この部分では “ (4) ” と出力される。これは、print メソッドが動的束縛されているためである。もし、Java が C++のように静的束縛を採用していれば、“ (5) ” と出力されるはずである。」

VI. 以下はキーを打つと画面上の文字列（“Hello”）の色が変化する簡単なアプレットのプログラムである。“R” キーが押されると赤色、“G” キーが押されると緑色、“B” キーが押されると青色、その他のキーが押されると黒色になる。空欄を埋め、プログラムを完成させよ。

解答は下のプログラムの空欄を埋めるものを、次のアイテムから適当なものを選んで書け。

① extends Applet
② g.drawString("Hello", 50, 50);axs
③ implements KeyListener
④ repaint();
⑤ if
⑥ else
⑦ addKeyListener(this);
⑧ if (k=='R') { c=Color.red; }
⑨ if (k=='G') { c=Color.green; }
⑩ if (k=='B') { c=Color.blue; }
⑪ c=Color.black;
⑫ g.setColor(c);
ⓐ {
ⓑ }

ただし、一つの解答欄に一つのアイテムとは限らず、複数のアイテムを並べて書く必要がある場合がある。

```
import java.awt.*;
import java.awt.event.*;
import java.applet.*;

public class KeyColor (1) {
    Color c;

    public void init() {
        (2)
    }

    public void paint(Graphics g) {
        (3)
    }

    public void keyPressed(KeyEvent e) {
        int k = e.getKeyCode();
        (4)
    }

    public void keyReleased(KeyEvent e) {}
    public void keyTyped(KeyEvent e) {}
}
```

計算機ネットワーク工学 I・テスト解答用紙 ('03 年 7 月 31 日)

学籍番号		氏名	

I. （2 点 ×8）

(1).		(2).		(3).		(4).	
(5).		(6).		(7).		(8).	

II. （6 点）

……
……
……
……
……
……
……
……

III. （8 点）

(1). x =	y =
(2). x =	y =

IV. (4 点 ×2)

(1).		(2).	

V. （4 点 ×5）

(1).	
(2).	
(3).	
(4).	
(5).	

VI. (3 点 ×4)

(1).	
(2).	
(3).	
(4).	

授業・テストの感想

……

……

……

……

……

……

……

……

……

……

……

……

……