CHALMERS TEKNISKA HÖGSKOLA
Datavetenskap TDA546

Tentamen i Objektorienterad programmering E

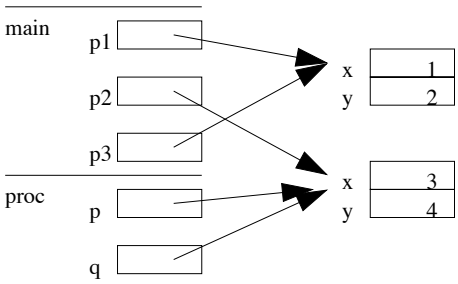
Tisdagen 11 januari 2011, 8.30 – 12.30.
Lösningsförslag.

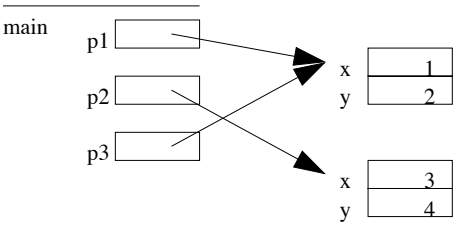
1.
```
public class Uppgift1 {

    public static int count(int n, int[] a) {
        int res = 0;
        for (int i=0; i< a.length; i++)
            if (a[i]==n) res++;
        return res;
    }

    public static void main(String[] args) {
        int[] a = new int[args.length - 1];
        int n = Integer.parseInt(args[0]);
        for (int i=0; i<a.length; i++)
            a[i] = Integer.parseInt(args[i+1]);
        System.out.println(n + " förekommer "
                           + count(n,a) + " gånger");
    }
}
```

2. Situationen just innan man lämnar subrutinen `proc`, som anropats från `main` ser ut som i den vänstra bilden nedan:





Koden i `proc` har gjort att bägge parametrarna `p` och `q` pekar på samma objekt, det som också `p2` i `main` pekar på. Notera dock att `p1` inte ändrats. När vi lämnat `proc` och kommer tillbaka till `main` "försvinner" `p` och `q` och vi har bilden till höger ovan. Utskriften blir

```
1
3
1
```

3. En möjlighet är följande:

```
public class Path {

    private Point[] points;
    private int count;

    public Path(Point p, int size) {
        points = new Point[size];
        count = 0;
        addPoint(p);
    }

    public void addPoint(Point p) {
        points[count] = p;
        count++;
    }

    public Point[] getPoints() {
        Point[] res = new Point[count];
        for (int i=0; i < count; i++)
            res[i] = points[i];
        return res;
    }
}
```

Här har vi valt att som tillstånd ha ett fält som innehåller `Point`-objekt; vi behöver då också en heltalsvariabel som håller rätt på hur stor del av fältet som utnyttjas vid varje tidpunkt. Vi har också valt att lägga till ytterligare en parameter till konstrueraren; denna anger fältets storlek.

Metoden `getCounts` returnerar här en kopia av den del av fältet som utnyttjas. En till synes enklare variant vore att returnera `points` direkt.

Detta har två allvarliga nackdelar: Man gör det möjligt för utomstående från att manipulera tillståndet, genom att t ex ta bort punkter, och man returnerar hela fältet, även den del som inte utnyttjas.

En alternativ lösning skulle kunna ha en `List` som tillstånd; i så fall är det naturliga att `getPoints` returnerar en iterator:

```
public class Points {

   private LinkedList<Point> points;

   public Path(Point p) {
      points = new LinkedList<Point>();
      points.add(p);
   }

   public void addPoint(Point p) {
      points.add(p);
   }
```

```
    public Iterator<Point> getPoints() {
        return points.iterator();
    }
}
```

I detta fall vore det ännu bättre om metoden bytte namn till `iterator` så att klassen implementerar `Iterable<Point>`.

4.
```
public class GlassFilter implements ImageFilter {

    public String getMenuName() {return "Glass";}

    public void apply(int[][][] src, int[][][] dest) {
for (int x=0; x< src.length; x++)
    for (int y=0; y<src[0].length; y++) {
                int xn = disturb(x,5,src.length-1);
                int yn = disturb(y,5,src[0].length-1);
                for (int c=0; c<3; c++)
                    dest[x][y][c] = src[xn][yn][c];
            }
    }

    private static int disturb(int x, int d, int max) {
        int nx = x + (int)(d*Math.random());
        return Math.max(0,Math.min(nx,max));
    }
}
```

5. (a)
```
private class ResetListener implements ActionListener {
    public void actionPerformed(ActionEvent e) {
        timer.stop();
        model.reset();
        world.repaint();
    }
}
```

   (b)
```
public class Main {

    public static void main(String[] args) {
        JFrame f = new JFrame("Crystallization simulation");
        f.add(new CrystalView(new CrystalModel(150)));
        f.pack();
        f.setVisible(true);
    }
}
```

   (c) Nödvändiga ändringar:
   Lägg till en tillståndsvariabel

```
private int count;
```

   i klassen `CrystalModel`, tillsammans med en metod

```
public int getCount() {return count;}
```

   Dessutom måste man se till att räkna upp denna variabel varje gång en partikel läggs till kristallen. I `if`-satsen

```
if (!crystal[x][y]) {
    crystal[x][y] = true;
}
```

i metoden `step` lägger man till `count++;` till kroppen.

Man måste också lägga till en rad sist i `paintComponent` i `CrystalWorld`:

```
g.drawString("" + model.getCount(),10,size-10);
```