Exam ID1018 19 March 2014 — optional part: solutions

ID1018

July 9, 2014

Exam — optional part: solutions

Tasks

Task 1 (4 points)

```
// sort sorts elements in a vector according to their size,
// from the least up to and including the greatest
public static void sort (int[] elements)
{
  int    first = 0;
  int    last = elements.length - 1;
  int    least;
  int    current;

  while (first < last)
  {
    least = first;
    current = first + 1;
    while (current <= last)
    {
      if (elements[current] < elements[least])
        least = current;

      current++;
    }

    int    p = elements[first];
    elements[first] = elements[least];
    elements[least] = p;

    first++;
  }
}
```

Task 2 (2 points + 1 point + 1 point)

a) (2 points)

It is written as `Rectangle2D.Double`. This means that class `Double` is defined as an inner class to `Rectangle2D`.

The reference `rect` of type `Rectangle2D` refers to an object of type `Double`. This means that class `Double` is a subclass to class `Rectangle2D`.

b) (1 point)

The vector `shapes` contains references of type `Shape`. The references can refer to objects of all classes that implement this interface.

An object of class `Rectangle2D.Double` and an object of class `Ellipse2D.Double` are in the vector `shapes`. This means that both these classes implement the interface `Shape`.

c) (1 point)

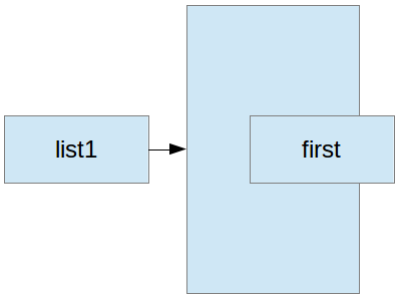
The method `draw` has a parameter of type `Shape`. That means it can draw all figures of type `Shape` (objects of all classes that implement interface `Shape`). Since the classes `Rectangle2D.Double` and `Ellipse2D.Double` implements interface `Shape`, the method draw can draw figures of these types:

```
Shape   r = new Rectangle2D.Double (10.0, 20.0, 60.0, 40.0);
Shape   e = new Ellipse2D.Double (70.0, 80.0, 40.0, 60.0);
g.draw (r);
g.draw (e);
```

Task 3 (1 point + 3 points)

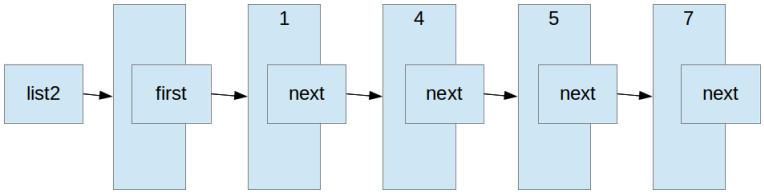
a) (1 point)

```
List    list1 = new List ();
```



b) (3 points)

```
List    list2 = new List ();
int[]    v = {1, 4, 5, 7};
for (int i = 0; i < v.length; i++)
    list2.add (v[i]);
```



Task 4 (2 points + 2 points)

a) (2 points)

```
// countDigits accepts a character sequence and returns
// the number of digits in the sequence.
public static int countDigits (CharSequence cs)
{
    int     count = 0;
    char    c = 0;
    int     len = cs.length ();
    for (int i = 0; i < len; i++)
    {
        c = cs.charAt (i);
        if (Character.isDigit (c))
            count++;
    }

    return count;
}
```

b) (2 points)

```
String    s = new String ("ten 10 fifty 50");
int    count1 = countDigits (s);
```

```
StringBuilder    sb = new StringBuilder ("E1D2C3B4A5");
int    count2 = countDigits (sb);
```

Task 5 (1 point + 1 point + 2 points)

a) (1 point)

In the best case the sought element is in the first position in the sequence. In that case only one comparison is performed. The time complexity of the algorithm is:

$$b(n) = 1$$

b) (1 point)

The worst case happens in two situations: when the sought element is not in the sequence and when the sought element is the last element in the sequence. In this situation all elements in the sequence are compared. The time complexity of the algorithm is:

$$w(n) = n$$

c) (2 points)

If the sought element is in the first position there will be one comparison, if it is on the second position there will be two comparisons, and so on. If the element is in the last position there will be n comparisons. The probability for each case is $\frac{1}{n}$. The total number of comparisons are:

$$\frac{1}{n} + \frac{2}{n} + \frac{3}{n} + \ldots + \frac{n}{n} = \frac{(1 + 2 + 3 + \ldots + n)}{n} = \frac{\frac{n(n+1)}{2}}{n} = \frac{n+1}{2}$$

The time complexity of the algorithm for the average case is:

$$a(n) = \frac{n+1}{2}$$