Programmer som Data - Assignment 8

Bastjan Rosgaard Sejberg, Søren Kastrup, Weihao Chen Nyholm-Andersen

November 2023

9.1

A

```
.method public hidebysig static void  SelectionSort(int32[] arr) cil managed
  {
    // Code size       57 (0x39)
    .maxstack  4
    .locals init (int32 V_0,            // Loop counter (outer loop)
             int32 V_1,                 // Store index of min element
             int32 V_2,                 // Loop counter (inner loop)
             int32 V_3)                 // temp store for swapping
    IL_0000:  ldc.i4.0                  // initialize V_0 w/ 0
    IL_0001:  stloc.0                   // Store value on V_0 stack
    IL_0002:  br.s       IL_0032        // Beginning of outer loop

    // Runs the outer loop
    IL_0004:  ldloc.0                   // loads V_0 val to stack
    IL_0005:  stloc.1                   // store V_1 value, tracking index of min element
    IL_0006:  ldloc.0
    IL_0007:  ldc.i4.1                  // Loads 1 onto the stack
    IL_0008:  add                       // adds the top two values on the stack (then push)
    IL_0009:  stloc.2                   // store in V_2 (inner loop)
    IL_000a:  br.s       IL_001a        // beginning of inner loop

    IL_000c:  ldarg.0                   // loads arr onto stack
    IL_000d:  ldloc.2                   // loads value V_2 onto stack
    IL_000e:  ldelem.i4                 // load the corresponding value at index V_2 in "arr"
    IL_000f:  ldarg.0                   // loads array onto the stack
    IL_0010:  ldloc.1                   // V_1 onto stack
    IL_0011:  ldelem.i4                 // load the corresponding value at index V_2 in "arr"
    IL_0012:  bge.s      IL_0016        // branches if >= IL_0016

    IL_0014:  ldloc.2                   // Loads V_2 onto stack
    IL_0015:  stloc.1                   // Pops curr val (min element) then store index in V_1
    IL_0016:  ldloc.2
    IL_0017:  ldc.i4.1                  // Loads 1 onto stack
    IL_0018:  add                       // adds the top two values on the stack (then push)
    IL_0019:  stloc.2                   // update v_2 for next iteration of loop
    IL_001a:  ldloc.2                   // load v_2 value
    IL_001b:  ldarg.0                   // load 'arr' onto stack
    IL_001c:  ldlen                     // get 'arr' length
    IL_001d:  conv.i4                   // conv length to int32
    IL_001e:  blt.s      IL_000c        // branch to inner loop beginning if V_2 < arrlen

    IL_0020:  ldarg.0                   // Load 'arr' to stack
    IL_0021:  ldloc.0                   // Load val of V_0 to stack
    IL_0022:  ldelem.i4                 // Load int at V_0 index from 'arr'
    IL_0023:  stloc.3                   // store val in V_3, temp store for swapping elements
    IL_0024:  ldarg.0                   // Load 'arr' to stack
    IL_0025:  ldloc.0                   // Load V_0 val onto stack
    IL_0026:  ldarg.0                   // Load V_1 onto stack
    IL_0027:  ldloc.1                   // Load V_1 val onto stack
    IL_0028:  ldelem.i4                 // Load int at index V_1 from 'arr'
    IL_0029:  stelem.i4                 // Store val at index V_1 of 'arr'
    IL_002a:  ldarg.0                   // Load 'arr' onto stack
    IL_002b:  ldloc.1                   // Load val v_1 onto stack
    IL_002c:  ldloc.3                   // Load val from V_3 (orig val at index V_0)
    IL_002d:  stelem.i4                 // Store the val in the element at index V_1 of 'arr'
    IL_002e:  ldloc.0                   // Load V_0 onto stack
    IL_002f:  ldc.i4.1                  // Load '1' onto the stack
    IL_0030:  add                       // adds the top two values on the stack (then push)
    IL_0031:  stloc.0                   // Update V_0 for next iter of outer loop, pop curr
    stack val
    IL_0032:  ldloc.0                   // Load V_0 onto stack
    IL_0033:  ldarg.0                   // Load 'arr onto stack
    IL_0034:  ldlen                     // get arr length
    IL_0035:  conv.i4                   // conv length to int32
    IL_0036:  blt.s      IL_0004        // branch to outer loop beginning if V_2 < arrlen

    IL_0038:  ret                       // Return method
  } // end of method Selsort::SelectionSort
```

Listing 1: Selsort.il

*Refer to the file **Selsort.il** in the folder **Exercise_9.1***

B

```
  public static void SelectionSort(int[]);
    descriptor: ([I)V                                                  // [I = int[], V = void
    flags: (0x0009) ACC_PUBLIC, ACC_STATIC                             // public, static
    Code:
      stack=4, locals=4, args_size=1                                   // stack: i, least, j, tmp
         0: iconst_0             // const 0                            // for($1; $2; $3)
         1: istore_1             // set i                              // $1 = int i = 0
         2: iload_1              // get i              #L0
         3: aload_0              // get arr
         4: arraylength          // call arr.length
         5: if_icmpge     57     // if lt goto 57 (#L4)                // $2 = i < arr.length
         8: iload_1              // get i
         9: istore_2             // set least                          // int least = i
        10: iload_1              // get i                              // for($4; $5; $6)
        11: iconst_1             // const 1
        12: iadd                 // calc i + 1
        13: istore_3             // set j                              // $4 = int j = i+1
        14: iload_3              // get j              #L1
        15: aload_0              // get arr
        16: arraylength          // call arr.length
        17: if_icmpge     37     // if lt goto 37 (#L3)                // $5 = j < arr.length
        20: aload_0              // get arr
        21: iload_3              // get j
        22: iaload               // get arr[j]                         // $7 = arr[j]
        23: aload_0              // get arr
        24: iload_2              // get least
        25: iaload               // get arr[least]                     // $8 = arr[least]
        26: if_icmpge     31     // if lt goto 31 (#L2)                // if($7 < $8)
        29: iload_3              // get j
        30: istore_2             // set least                          // least = j
        31: iinc          3, 1   // increase j          #L2             // $6 = j++
        34: goto          14     // goto 14 (#L1)
        37: aload_0              // get arr            #L3
        38: iload_1              // get i
        39: iaload               // get arr[i]
        40: istore_3             // set arr[i]                         // int tmp = arr[i]
        41: aload_0              // get arr
        42: iload_1              // get i
        43: aload_0              // get arr
        44: iload_2              // get least
        45: iaload               // get arr[i], arr[least]
        46: iastore              // set arr[i]                         // arr[i] = arr[least]
        47: aload_0              // get arr
        48: iload_2              // get least
        49: iload_3              // get tmp
        50: iastore              // set arr[least]                     // arr[least] = tmp
        51: iinc          1, 1   // increase i                         // $3 = i++
        54: goto          2      // goto 2 (#L0)
        57: return               // return             #L4
      LineNumberTable:                                                 // links src to code (debug)
        line 21: 0
        line 22: 8
        line 23: 10
        line 24: 20
        line 25: 29
        line 23: 31
        line 26: 37
        line 21: 51
        line 28: 57
      StackMapTable: number_of_entries = 5                             // type check reference
        frame_type = 252 /* append */
          offset_delta = 2
          locals = [ int ]
        frame_type = 253 /* append */
          offset_delta = 11
          locals = [ int, int ]
        frame_type = 16 /* same */
        frame_type = 250 /* chop */
          offset_delta = 5
        frame_type = 249 /* chop */
          offset_delta = 19
```

Listing 2: Selsort.jvmbytecode

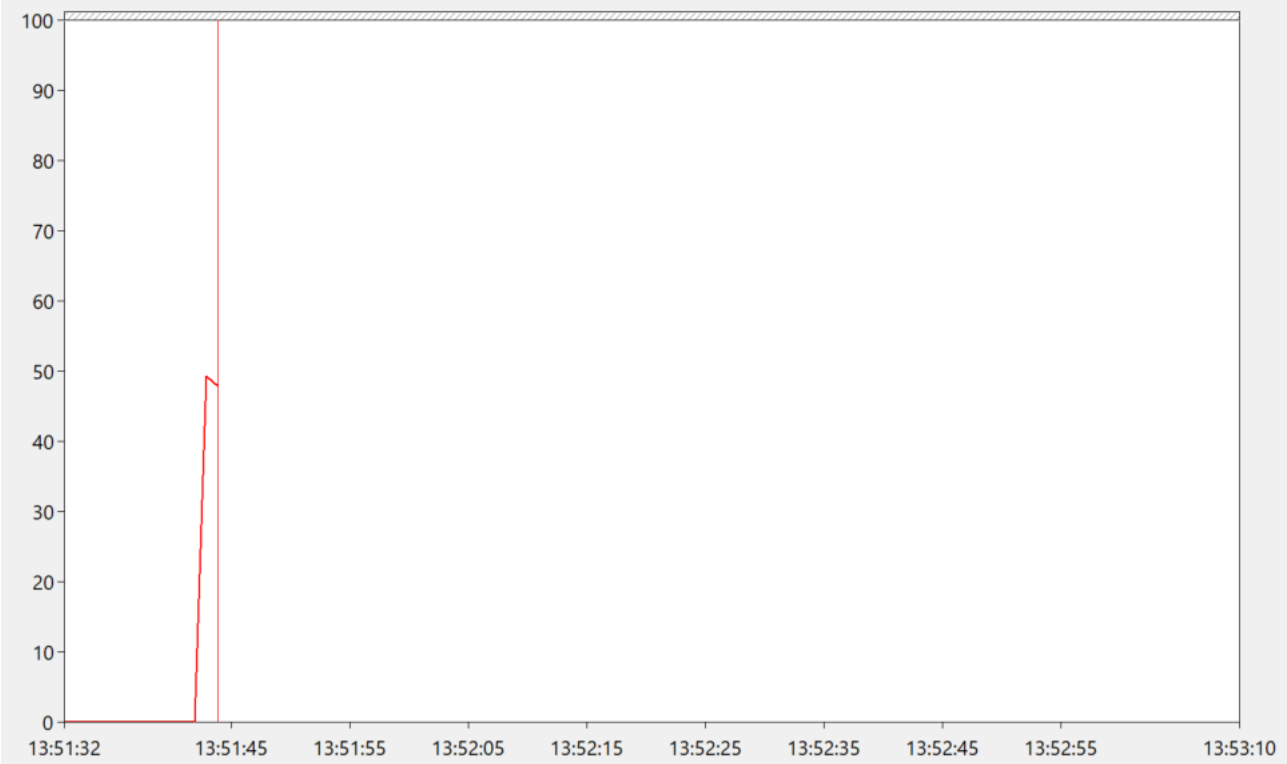
Refer to the file ***Selsort.jvmbytecode*** *in the folder* ***Exercise_9.1***

9.2

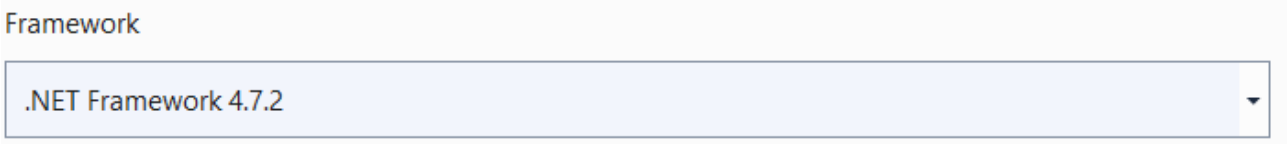
i

Build done.

ii



It uses about 50%



iii

We could not find but the purpose is to compare the programs to see which one has the better garbage collection. As mentioned in the exercise, a well written application will only have about 0-10% while comparing this to the previous exercise's program, StringConcatSpeed we clearly see that it is not an optimal application.