Seminar Objectives

Inspection AND Feature Tour Testing

Topics

- **Inspection**
- **Tour testing (Feature)**

Assignment 1 - 10-minutes - Discussion

Topics

- Similarities/Differences between code inspection and walkthroughs.
- Inspection/Walkthroughs team and duties.
- Types of errors to be discovered. Name at least three and give an example for each type.
- Pair programming

Assignment 2 – 60 minutes – Inspection

Inspect the documents (problem statement, design, source code) for the received problem.

Inspection refers to the analysis and the highlighting of the current state of the documents into a report.

Inspection may cause modification of the analyzed documents, like:

- Clarification of the problem statement.

- Modification of the design and/or the source code.

- Use the same available documents from the Laboratory 1 assignments.

- For the identification of the ambiguities/defects the following *check-lists* will be used:

  - o a. **Statement problem:** Lab01_RequirementsPhaseDefectsChecklist
  - o b. **Design:** Lab01_DesignPhaseDefectsChecklist
  - o c. **Source code**: Lab01_ProgramCodingPhaseDefectsChecklist
- For the inspected documents/artifacts a report will be realized (Lab01_Review Form).

Assignment 3 – 30 minutes – Feature Tour Testing
Tour testing
- **Lecture 1: Function Testing, Tours, & A Taxonomy of Techniques**
- http://www.testingeducation.org/BBST/testdesign/
- Video part A (9 mins, 14 secs) - starting 4:45 to 9:15 minutes.
- Slides - Slide set for all Test Design course lectures – slides 9->21 (->41)

Problem statement. Write a program that reads natural numbers n1, n2, …, nk and prints the longest sequence ns, ns+1, …, nd, with 1≤s≤d≤k, that contains only prime numbers.

Problem design. The program must have: a subalgorithm that reads the given numbers, a function that verifies if a natural number is prime;a subalgorithm that computea the indexies s and d, 1≤s≤d≤k, with the property that ns, ns+1, …, nd are prime numbers; a subalgorithm that prints the numbers ns, ns+1, …, nd.

Source code.

```
1 public class LongestPrimeSequence {
2      private ArrayList l;
3      private int start, length;
4      public LongestPrimeSequence(){
5              System.out.println("Long. Seq. empty ...");
6      }
7      public void setSequence(ArrayList l){
8              this.l=l;
9      }
10     public LongestPrimeSequence(ArrayList newL){
11             this.l = newL;
12             this.start=-1;
13             this.length=0;
14     }
15     public int getStart(){return this.start;}
16     public int getLength(){return this.length;}
17   public boolean isPrime(int n) throws ValueException{
18     boolean b = true;
19     if(n<0){
20             throw new ValueException("data not valid");
21     }
22     if(n<2){
23             b=false;
24     }
25     else{
26             int i=2;
27             while (i< (n/2)){
28                     if ((n % i) == 0){
29                             b=false;
30                     }
31                     else
32                             b=true;
33                     i++;
34             }
35     }
36     return b;
37 }
```

```
38 public void SolveLongestSequence() throws ValueException{
39    int posI=-1, lengthI=0, i=0;
40    int posF=-1, lengthF=0;
41    while(i<this.l.size()){
42            if(isPrime((int)this.l.get(i))==true){
43                    if(posI==-1){
44                            posI=i;
45                            lengthI=1;
46                    }
47                    else
48                            lengthI++;
49            }
50            else{
51                    if(lengthI>lengthF){
52                            lengthF=lengthI;
53                            posF = posI;
54                    }
55            }
56            i++;
57    }
58    this.start =posF;
59    this.length=lengthF;
60  }
61 }
```