PRG – ETSINF – THEORY – Academic year 2013/14
Retake Second Partial Exam – June 23rd, 2014 – Duration: 1h 50m

1. 2.5 points Given a text file whose name is in the parameter `String filename` and a word, you have to write a method for checking if the word is contained in the file. The method should deal with the possibility that exceptions of the class `FileNotFoundException` could be thrown. This kind of exception is commonly thrown when no file with the given name exists. In such cases the method must show a message to the user on the screen and return `false`.

Solution:

```
public static boolean isTheWordInTheFile( String filename, String word )
{
    try {
        Scanner input = new Scanner( new File( filename ) );
        boolean found = false;
        while( input.hasNext() && !found ) {
            String w = input.next();
            found = w.equals( word );
        }
        input.close();
        return found;
    }
    catch( FileNotFoundException e )
    {
        System.out.println( "File " + filename + " not found!" );
        return false;
    }
}
```

2. 2.5 points Given the class

```
class NodePerson {
   int dni;
   String name;
   NodePerson next;

   NodePerson( int i, String s, NodePerson n ) {
      dni = i;
      name = s;
      next = n;
   }
}
```

You have to add the following static methods to this class:

1. (1.25 points) `count()`, such that, given a sequence of objects of the class `NodePerson` and a `String w`, returns the number of objects (nodes) whose attribute `name` contains the `String w`. The `String` class has a method with the profile `boolean contains( String s )` for checking if `s` is contained in the `String` over which the method is executed.
2. (1.25 points) `search()`, such that, given a sequence of objects of the class `NodePerson` and an integer `n`, checks if it exists at least one node whose attribute `dni` is equal to `n`. If a node is found then the method returns the attribute `name`, otherwise `"Unknown person"` should be returned.

Solution:

```
public static int count( NodePerson p, String s )
{
    int counter = 0;
    NodePerson temp = p;
    while( temp != null ) {
        if (temp.name.contains(s) ) counter++;
        temp = temp.next;
    }
    return counter;
}

public static String search( NodePerson p, int i )
{
    NodePerson temp = p;
    while( temp != null ) {
        if ( temp.dni == i ) return temp.name;
        temp = temp.next;
    }
    return "Unknown person";
}
```

3. 2.5 points We need to add a new method into class `ListIPIntLinked` for inserting at the interest point, besides the existing method `insert(int)`. But the new method should perform the insert operation if the value to be inserted doesn't exist in the list. The name of the new method should be `insertWithNoDuplicates(int)`. If the value to be inserted already exists in the list the method must throw an exception of the class `IllegalArgumentException`. This class is predefined in Java and it is derived from the class `RuntimeException`. If the exception is thrown the message must be `"The value already exists!"` followed by the value to be inserted. See the following example:

   - If the contents of the list is `9 1 4 2 9` and we want to insert the value `0`, the final configuration of the list should be `9 1 0 4 2 9`.
   - If the contents of the list is `9 1 4 2 9` and we want to insert the value `2`, the list must remain untouched and an exception of the class specified above must be thrown with the message `"The value already exists! 2"`.

   To be done:

   1. (2.25 points) Write the method `insertWithNoDuplicates( int value )` following the above description. Notice that if `value` is not in the list the existing `insert( int )` method can be used.

      Solution:

      ```
      public void insertWithNoDuplicates( int value )
      {
          NodeInt temp = first;
          while( temp != null && temp.getValue() != value ) temp = temp.getNext();
          if ( temp != null )
              throw new IllegalArgumentException( "The value already exists! " + value );
          this.insert( value );
      }
      ```

   2. (0.25 points) In the case the new method was executed from the method `main()` of any class, would it be mandatory to write the code calling the method within the mechanism `try-catch` or to propagate the

exception by means of adding the corresponding code into the header of the `main()` method? Explain your answer.

Solution: No, because no *checked* exceptions can be thrown.

4. 2.5 points Given a class named `Exam`, we need adding to it a new method for obtaining the maximum value stored in a list, an object of the class `QueueIntLinked q`. At the end of the operation `q` should be in the same state that it was before calling the new method, i.e., `q` must contain the same values and in the same disposition. If the queue is empty the new method should throw an exception of the class `NoSuchElementException` with the message `"Empty queue: no maximum value!"`. See the following example:

   - If `q` is `<- 4 -2 9 8 <-`, must return `9` and the state of `q` must be `<- 4 -2 9 8 <-`.
   - If `q` is `<- -2 <-`, must return `-2` and leave `q` as it was: `<- -2 <-`.
   - If `q` is `<- <-`, `q` must remain as empty and an exception should be thrown.

Solution:

```
public static int maximum( QueueIntLinked q )
{
    int n = q.size();
    if ( n == 0 ) throw new NoSuchElementException( "Empty queue: no maximum value!" );
    int e = q.dequeue();
    int max = e;
    q.enqueue(e); n--;
    while( n > 0 ) {
        e = q.dequeue();
        max = ( e > max ) ? e : max;
        q.enqueue(e); n--;
    }
    return max;
}
```