COMPUTER PROGRAMMING – ETSINF – ACADEMIC YEAR 2016/2017

Second mid term lab exam – June 5, 2017 – duration 1 hour

Notice: The maximum mark of this exam is 10 points, but his specific weight in the final grade is **1,2 points**

NAME:	LAB GROUP:

1. 5 points We need to improve the class `Bank` from lab practice 4 in order to manage transfer remittances. A remittance is in a text file where each line contains three data separated by white spaces: *source account*, *target account* and *amount*.

```
sourceAccount    targetAccount    amountInEuros
```

where `sourceAccount` and `targetAccount` are integers in the range $[10000, 99999]$ corresponding to valid accounts existing in the bank, and `amountInEuros` is a real positive value.

The three data in a line indicate a transfer of the amount in euros that should be made from the source account to the target account.

What to do: Assuming that all the needed classes have been properly imported in the class `Bank`, write a new method in this class with the following profile:

```
public void makeTransfer( Scanner remittance )
```

that taking an object of the class `Scanner` already initialised and passed as parameter, reads all the lines one at a time and carries out all the transfers. Each transfer consists in withdrawing the amount in euros from the source account and deposit the same amount into the target account.

In case of any error in a line, the three data in that line should be ignored and a message should be shown in order to inform the user. The remaining lines should be processed if they are correct, i.e., a line with wrong data should be ignored but the method must continue processing the remaining lines.

Notice: Recall that the method `withdraw(double)` from class `Account` can throw an exception of the class `NoMoneyException` if the amount to be withdrawn is greater than the available balance in the account. Also, methods from class `Scanner` can thrown an exception of the class `InputMismatchException` when trying to read a value if the next available token does not match with the correct syntax required by the data type to be read.

Moreover, recall that method `getAccount(int)` from class `Bank` returns the reference to a valid object of the class `Account` if an account with the provided identifier exists, otherwise it returns `null`. Nevertheless, method `getAccount(int)` will never return `null` in this exercise because all the accounts in the provided file with the remittance exist in the bank.

Solution:

```
public void makeTransfer( Scanner remittance )
{
    int sourceAccount = 0, targetAccount = 0;
    double amount = 0;
    while( remittance.hasNext() ) {
        try {
            sourceAccountId = remittance.nextInt();
            targetAccountId = remittance.nextInt();
            amount = remittance.nextDouble();
            Account sourceAccount = this.getAccount( sourceAccountId );
            Account targetAccount = this.getAccount( targetAccountId );
            sourceAccountId.withdraw( amount );
```

```
                targetAccountId.deposit( amount );
            }
            catch( InputMismatchException e ) {
                System.err.println( "line with wrong data." );
            }
            catch( NoMoneyException e ) {
                System.err.println( "account " + sourceAccountId
                                    + " has no enough balance." );
            }
            finally {
                remittance.nextLine();
            }
        }
    }
```

2. [5 points] Taking as starting point the class `StringSet` studied in lab practice 5:

```
public class StringSet {

    private StringNode first;
    private int size;

    /** Creates an empty set. */
    public StringSet()
    {
        this.first = null;
        this.size = 0;
    }

    ....
}
```

where the linked sequence whose first element is referenced by the attribute `first` contains all the elements in the set. As you know, all the methods in this class are implemented for ensuring that all the stored strings are sorted in ascending order and no repeated values exist in the set.

What to do: Implement a new method with the following profile:

```
public StringSet subset( char ch )
```

that returns an object of the same class with all the elements in the set whose first character is `ch`.

Your implementation must fulfil $T_{subset}(n) \in O(n)$, where n is `this.size`.

Recall that method `charAt(int pos)` of the class `String` returns a value of type `char` corresponding to the character at the position given by the parameter `pos`.

Solution:

```
public StringSet subset( char ch )
{
```

```
        StringSet subset = new StringSet();
        StringNode p = first;

        while( p != null  &&  p.getValue().charAt(0) < ch ) p=p.getNext();

        StringNode q = null;

        while( p != null  &&  p.getValue().charAt(0) == ch ) {

            StringNode sn = new StringNode( p.getValue() );

            if ( q == null ) {
                subset.first = sn;
            } else {
                q.setNext(sn);
            }
            q = sn;

            subset.size++;
            p=p.getNext();
        }
        return subset;
    }
```