

## R3.04

### TD1 : Rétro-conception depuis Java

#### Principaux concepts discutés:

- Rétro-conception ;
- Relations entre classes.

#### Principaux apports techniques:

- Analyse d'un code source en vue d'en extraire la structure du modèle sous-jacent.

Dans chacun des exercices suivants, dessinez les diagrammes de classes de conception qui correspondent au code source.

#### Exercice 1

```
package company;
import java.util.ArrayList;

public class Company {

    private String name;
    private String address;
    private ArrayList<Employee> staff;

    public Company (String name, String address) {
        this.name = name;
        this.address = address;
        staff = new ArrayList<Employee>();
    }
}
```

---

```
package company;
public abstract class Employee {

    protected String firstName;
    protected String lastName;

    public Employee (String first, String last) {
        firstName = first;
        lastName = last;
    }

    public String getFirstName() {
        return firstName;
    }

    public String getLastName(){
        return lastName;
    }
}
```

```

    public String toString(){
        return firstName + ' ' + lastName;
    }

    public abstract double earnings();
}

package company;

public final class Boss extends Employee{

    private double weeklySalary;

    public Boss(String first, String last, double s) {
        super(first, last);
        this.setWeeklySalary(s);
    }

    public void setWeeklySalary(double s){
        weeklySalary= (s > 0 ? s : 0);
    }

    public double earnings() {
        return weeklySalary;
    }

    public String toString() {
        return "Boss: " + super.toString();
    }
}

```

---

## **Exercise 2**

```

package liste ;
import java.util.Enumeration;

public class ListeChaine {

    private Noeud racine;

    public Enumeration enumerate() {
        return new ListeChaineEnumeree (racine);
    } // end enumerate

} // end ListeChaine

```

---

```

package liste ;

public class Noeud {
    private Object contenu;
    private Noeud suivant;
}

```

```

    public Object contenu() { return contenu; }
    public Noeud suivant() { return suivant; }
}

```

---

```

package liste ;

import java.util.Enumeration;

public class ListeChaineEnumeree implements Enumeration {

    private Noeud noeudCourant;

    public ListeChaineEnumeree (Noeud racine) {
        noeudCourant = racine;
    }

    public boolean hasMoreElements() {
        return noeudCourant != null;
    }

    public Object nextElement() {
        Object unObjet = noeudCourant.contenu();
        noeudCourant = noeudCourant.suivant();
        return unObjet;
    }
}

```

---

### **Exercise 3**

```

package pass;
public interface StringHandler {
    public void processLetter(char c);
    public void processDigit(char c);
    public void processOther(char c); }

```

---

```

package pass;
public class PasswordSecurityHandler implements StringHandler {

    private int length;
    private boolean digit;
    private boolean otherCharacter;

    /** default constructor
     * sets length to 0, digit and otherCharacter flags to false
     */
    public PasswordSecurityHandler() {
        length = 0;
        digit = false;
        otherCharacter = false;
    }
}

```

```

/** processLetter method
 * @param c : character to process
 * adds 1 to length
 */
public void processLetter(char c){
    length++;
}

/** processDigit method
 * @param c : character to process
 * adds 1 to length, sets digit flag to true
 */
public void processDigit(char c) {
    length++;
    digit = true;
}

/** processOther method
 * @param c : character to process
 * adds 1 to length
 * sets otherCharacter flag to true
 */
public void processOther(char c){
    length++;
    otherCharacter = true; }

public void parse(String s) {
    if (s != null) {
        for (int i = 0; i < s.length(); i++) {
            char c = s.charAt(i);
            if (Character.isDigit(c))
                this.processDigit(c);
            else if (Character.isLetter(c))
                this.processLetter(c);
            else this.processOther(c);
        }
    }
    else System.out.println("String s is null");
}

/** securityLevel method
 * @return "weak" if password contains fewer than 6 characters
 * "strong" if password has at least 8 characters, at least one digit,
 * and at least one other character that is neither a letter nor a di
 * git "medium" otherwise
 */
public String securityLevel() {
    String result = null;
    if (length < 6) result="weak";
    else if (length >= 8 && digit && otherCharacter)
        result = "strong";
    else result = "medium";
    return result;
}

```

```
}  
import java.util.Scanner;  
import pass.PasswordSecurityHandler;  
  
public class PasswordSecurityHandlerClient {  
    public static void main (String [] args) {  
        Scanner scna = new Scanner(System.in);  
        System.out.println("A string password has at least 8  \n"  
            + "characters and contains at least one digit\n"  
            + "and one speical character.");  
        System.out.println("Enter a password >");  
        String password = scna.next();  
  
        PasswordSecurityHandler psh = new PasswordSecurityHandler();  
        psh.parse(password);  
        System.out.println(password + "'s security is " +  
            psh.securityLevel());  
    }  
}
```