EXAMEN FINAL

02 heures-Documents interdits

La lisibilité de votre code sera prise en considération. Une copie non soignée ne sera pas corrigée.

Partie 3 : Modélisation et Evaluation d'une formule logique (11.5 pts)

En logique propositionnelle, une formule logique sous forme normale conjonctive est une conjonction de clauses. Chaque clause est une disjonction de littéraux, et chaque littéral est une variable (littéral positif) ou la négation d'une variable (littéral négatif).

Exemple: Soit $V = \{v_1, v_2, v_3\}$ un ensemble de variables, et soit $F = C_1 \wedge C_2 \wedge C_3$ où:

$$C_1 = v_1 \vee \overline{v_2}$$
$$C_2 = \overline{v_1} \vee \overline{v_2} \vee v_3$$
$$C_3 = v_2 \vee \overline{v_3}$$

A/ Modéliser une formule logique (6.75pt)

1. En page 5, vous est fourni le code java de la classe `Main` contenant la méthode `main` dans laquelle nous créons la formule donnée dans l'exemple. Lisez-le attentivement puis répondez aux questions suivantes:
   a) Quelles sont les classes utilisées dans ce programme ?
   b) Donnez le tableau décrivant chacune de ces classes de la même manière que vous avez l'habitude de le faire en TP.

Attention:

- Vous devez préciser le modificateur d'accès et le type de vos attributs. Ce dernier doit être **concret**.
- Les tableaux doivent comporter tous les constructeurs, accesseurs, mutateurs et méthodes **nécessaires** pour que le programme donné compile et s'exécute sans aucune erreur.

2. Maintenant que la formule F est créée, nous désirons l'afficher à l'écran de sorte que le résultat obtenu soit le suivant:

```
La formule donnée dans l'exemple est :

{   C1 = v1 OU !v2
    C2 = !v1 OU !v2 OU v3
    C3 = v2 OU !v3
}
```

a) Les collections utilisées dans la conception que vous avez proposée en question 1 garantissent-elles un tel affichage **à chaque exécution du programme?** Expliquez clairement pourquoi.
b) Si votre réponse est "Non", expliquez, sans donner le code java, ce que vous devez modifier pour garantir cet affichage.

B/ Calculer la valeur de vérité d'une formule logique (4.75pt)

Afin d'évaluer une formule logique, nous devons attribuer une valeur booléenne à chaque variable de l'ensemble V. Ceci s'appelle une assignation. On dit qu'une assignation A satisfait une formule F si la valeur de vérité de F dans A est "vrai". Dans l'exemple donné précédemment, l'assignation [(v_1,vrai), (v_2,faux) , (v3, faux)] satisfait la formule F, alors que l'assignation [(v_1,vrai), (v_2,faux), (v_3, vrai)] ne la satisfait pas.

1. Nous ajoutons la classe Assignation définie comme suit:

```
import java.util.*;

public class Assignation {
        Map<Variable,Boolean> valeursVariables;

        //Les « ..... » correspondent aux types concrets des collections

        public Assignation (.....<Variable>ensembleVariables){
        valeursVariables = new.....<Variable,Boolean>();
                for (Variable v: ensembleVariables){
                        valeursVariables.put(v,null);
                }
        }
}
```

On veut ajouter à cette classe la méthode `affecterBooleen` qui affecte une valeur booléenne à une variable donnée. Cette méthode doit lancer une exception `ErreurAffectation`si la variable possède déjà une valeur booléenne.

a) Donnez le code java de cette méthode.
b) Donnez le code java de la classe `ErreurAffectation`
c) Donnez la portion de code de la méthode main qui crée l'assignation $[(v_1,vrai),(v_2,faux),(v_3, vrai)]$ et affiche, en cas d'erreur, le message : `"Vous voulez attribuer deux valeurs logiques à la même variable ! "`

2. Afin de savoir si une assignation satisfait la formule F, nous ajoutons une méthode `evaluerFormule` qui calcule la valeur de vérité d'une formule dans une assignation donnée. Cette méthode fait appel à la méthode `evaluerClause` qui fait à son tour appel à la méthode `evaluerLitteral`

a) Dans quelle classe doit se trouver chacune de ces trois méthodes.
b) Donnez le code java de chacune d'elles.

Note: Pour répondre aux questions de la partie B, vous pouvez utiliser les méthodes suivantes fournies par le langage java

Boolean(boolean value)	Allocates a Boolean object representing the value argument.
Boolean(String s)	Allocates a Boolean object representing the value true if the string argument is not null and is equal, ignoring case, to the string "true".
boolean booleanValue()	Returns the value of this Boolean object as a boolean primitive.
V get(Object key)	Returns the value to which the specified key is mapped, or null if this map contains no mapping for the key.
boolean containsKey(Object key)	Returns true if this map contains a mapping for the specified key.
boolean containsValue(Object value)	Returns true if this map maps one or more keys to the specified value

Code java de la classe Main (Question 1)

```
import java.util.*;

public class Main {
public static void main (String args[]){

        //Les « ……. » correspondent aux types concrets des collections

        Variable v1, v2, v3;
        Clause c1, c2, c3;
        .......ensembleVariables;
        .......ensembleClauses;
        Formule F;

        /* création des variables*/

        v1 = new Variable("v1");
        v2 = new Variable("v2");
        v3 = new Variable("v3");

        /*Création des clauses*/

        // Clause 1

        c1 = new Clause("C1");
        c1.ajouterLitteral(new Litteral(v1,TypeLitteral.positif));
        c1.ajouterLitteral(new Litteral(v2,TypeLitteral.negatif));

        //Clause 2

        Litteral v1_negatif, v2_negatif, v3_positif;
        v1_negatif = new Litteral();
        v1_negatif.variable = v1;
        v1_negatif.setType(TypeLitteral.negatif);
        v2_negatif = new Litteral (v2, TypeLitteral.negatif);
        v3_positif = new Litteral ("v3", TypeLitteral.positif);

        c2 = new Clause("C2");

        c2.ajouterLitteral(v1_negatif);
        c2.ajouterLitteral(v2_negatif);
        c2.ajouterLitteral(v3_positif);

        //clause 3

        .......<Litteral> litteraux_C3 = new ...... <Litteral>();
        litteraux_C3.add(new Litteral(v2,TypeLitteral.positif));
        litteraux_C3.add(new Litteral(v3,TypeLitteral.negatif));

        c3= new Clause("C3", litteraux_C3);

        // création de la formule F

        ensembleVariables = new ....... <Variable>();
        ensembleVariables.add(v1); ensembleVariables.add(v2); ensembleVariables.add(v3);

        ensembleClauses = new ........ <Clause>();
        ensembleClauses.add(c1); ensembleClauses.add(c2); ensembleClauses.add(c3);

        F = new Formule (ensembleVariables, ensembleClauses);
    }
}
```