(10) 1. Écrire ce qui sera imprimé par le programme suivant. Chaque ligne imprimée vaut 1 point.

```
class Inher {
    public static void main (String [] args) {
        Voiture i = new Voiture(5.0);
        print("1=" + i.name1 + " " + i.name2);
        print("2=" + i.getName1() + " " + i.getName2());
        print("3=" + i.describe());
        print("4=" + i.x + " and " + i.y);
        Vehicule j = i;
        print("5=" + j.name1 + " " + j.name2);
        print("6=" + j.getName1() + " " + j.getName2());
        print("7=" + j.describe());
        print("8=" + ((Item)j).y + " and " + j.y + " and " + ((Voiture)j).y);
        print("9=" + j.y++ + " and " + j.x + " and " + ++j.x);
        print("10=" + ((Item)j).y + " and " + j.y + " and " + ((Voiture)j).y);
    }
    public static void print(String s)  { System.out.println(s); }
}
class Item {
    int x=9;
    int y=8;
    public String name1="Item";
    public String name2;
    Item() { x=84; name2="Item";}
    String getName1() { return name1; }
    String getName2() { return name2; }
    String describe() { return name1 + " has no parent.";}
}
class Vehicule extends Item {
    int y=3;
    public String name1="Vehicule";
    Vehicule() { name2="Vehicule";}
    String describe() { return name1  + " has parent " + super.describe(); }
}
class Voiture extends Vehicule {
    int y=16;
    public String name1="Voiture";
    Voiture() { name2="Voiture"; x=12; y=12;}
    Voiture(int i) { name2="Voiture"; x=i; y=x; }
    Voiture(double i) {name2="Voiture"; x=(int)(i*10.0);}
    String describe() { return name1 + " has parent " + super.describe(); }
}
```

Solution 1

```
1=Voiture Voiture
2=Item Voiture
3=Voiture has parent Vehicule has parent Item has no parent.
4=50 and 16
5=Vehicule Voiture
6=Item Voiture
7=Voiture has parent Vehicule has parent Item has no parent.
8=8 and 3 and 16
9=3 and 50 and 51
10=8 and 4 and 16
```

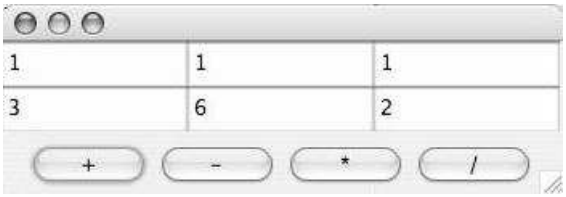

Figure 1: Résultat après avoir cliqué sur le bouton d'addition. $\frac{1}{3}$ a été ajouté à $\frac{1}{6}$ pour donner $\frac{1}{2}$ comme résultat.

(20) 2. Implanter le calculateur d'expressions fractionnelles de la Figure 1 dont voici la structure.

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
public class FracCalc extends JFrame implements ActionListener  {
    private JTextField tGaucheNum, tGaucheDenom;
    private JTextField tDroitNum, tDroitDenom, tReponseNum, tReponseDenom;
    private JPanel pFrac, pButton;
    private JButton bAdd, bSub, bMult, bDiv;
    public FracCalc() {
        pFrac = new JPanel();
        pFrac.setLayout(new GridLayout(2,3));
        tGaucheNum = new JTextField(6);
        tGaucheDenom = new JTextField(6);
        tDroitNum = new JTextField(6);
        tDroitDenom = new JTextField(6);
        tReponseNum = new JTextField(6);
        tReponseDenom = new JTextField(6);
        pFrac.add(tGaucheNum);
        pFrac.add(tDroitNum);
        pFrac.add(tReponseNum);
        pFrac.add(tGaucheDenom);
        pFrac.add(tDroitDenom);
        pFrac.add(tReponseDenom);
        pButton = new JPanel();
        bAdd = new JButton("+");
        bSub = new JButton("-");
        bMult = new JButton("*");
        bDiv = new JButton("/");
        pButton.add(bAdd);
        pButton.add(bSub);
        pButton.add(bMult);
        pButton.add(bDiv);
```

```
        this.getContentPane().setLayout(new BorderLayout());
        this.getContentPane().add(pFrac,BorderLayout.CENTER);
        this.getContentPane().add(pButton,BorderLayout.SOUTH);

        //  ... ajouter le code ici ....
    }
    public static void main (String args[]) {
        FracCalc fc = new FracCalc();
        fc.setSize(350,120);
        fc.setVisible(true);
    }
}
```

(10) (a) Écrire la `class Fraction` qui implante `FractionInterface`. Pour chaque opération, la `Fraction` résultante devrait être réduite à son plus petit dénominateur.

```
public interface FractionInterface {
    Fraction add(Fraction that);
    Fraction subtract(Fraction that);
    Fraction multiply(Fraction that);
    Fraction divide(Fraction that);
    int getNumerator();
    int getDenominator();
}
```

(10) (b) Ajouter le code dans la structure de la classe précédente pour connecter l'interface usager aux 4 opérations à effectuer sur les fractions en utilisant les opérations de la classe `class Fraction`. Vous pouvez ajouter du code à la fin du constructeur `public FracCalc()` et aussi, si nécessaire, dans une méthode séparée.

Rappel

Pour simplifier les fractions, vous pouvez supposer l'existence de la méthode `Math.gcd(int n, int d)` qui retourne le plus grand dénominateur commun de deux nombres, p.e. `Math.gcd(16,12)` retourne `4`.

Voici les formules pour les opérations arithmétiques sur les fractions:

$$\frac{a}{b} + \frac{c}{d} = \frac{ad+cb}{bd} \qquad \frac{a}{b} - \frac{c}{d} = \frac{ad-cb}{bd}$$
$$\frac{a}{b} \times \frac{c}{d} = \frac{ac}{bd} \qquad \frac{a}{b} \div \frac{c}{d} = \frac{ad}{bc}$$

Solution 2

```
    ...
    bAdd.addActionListener(this);
    bSub.addActionListener(this);
    bMult.addActionListener(this);
    bDiv.addActionListener(this);
}
public void actionPerformed(ActionEvent e) {
  int gNum = Integer.parseInt(tGaucheNum.getText());
  int gDenom = Integer.parseInt(tGaucheDenomm.getText());
  int dNum = Integer.parseInt(dGaucheNum.getText());
  int dDenom = Integer.parseInt(dGaucheDenomm.getText());
  Fraction fGauche = new Fraction(gNum,gDenom);
  Fraction fDroit  = new Fraction(dNum,dDenom);
  Fraction fResult = null;
  if (e.getSource()==bAdd) {
    fResult=fGauche.add(fDroit);
  } else if (e.getSource()==bSub) {
    fResult=fGauche.sub(fDroit);
  } else if (e.getSource()==bMult) {
    fResult=fGauche.mult(fDroit);
  } else if (e.getSource()==bDiv) {
    fResult=fGauche.div(fDroit);
  }
  tReponseNum.setText(""+fResult.num());
  tReponseDenom.setText(""+fResult.denom());
}
class Fraction {
    private int num;
    private int denom;
    Fraction(int num, int denom) {
        this.num=num;
        this.denom=denom;
    }
    Fraction add(Fraction that) {
        int nDenom=denom*that.denom();
        int nNum=num*that.denom() + that.num()*denom();
        Fraction ans = new Fraction(nNum,nDenom);
        return ans.reduce();
    }
    Fraction sub(Fraction that) {
        int nDenom=denom*that.denom();
```

```
        int nNum=num*that.denom() - that.num()*denom();
        Fraction ans = new Fraction(nNum,nDenom);
        return ans.reduce();
    }
    Fraction mult(Fraction that) {
        int nDenom=denom*that.denom();
        int nNum=num*that.num();
        Fraction ans = new Fraction(nNum,nDenom);
        return ans.reduce();
    }
    Fraction div(Fraction that) {
        int nDenom=denom*that.num();
        int nNum=num*that.denom();
        Fraction ans = new Fraction(nNum,nDenom);
        return ans.reduce();
    }
    public int num() { return num;}
    public int denom() { return denom;}
    private Fraction reduce() {
       int gcd = gcd(this.num,this.denom);
       this.num/=gcd;
       this.denom/=gcd;
       return this;
    }
}
```

(30) 3. Écrire un programme pour traduire des textes de l'anglais au français. Votre programme reçoit deux paramètres:

- le nom d'un fichier pour un dictionnaire contenant des paires de mots anglais et français
- le nom d'un fichier contenant un texte en anglais.

Il faut d'abord lire le dictionnaire et le conserver dans une structure de données appropriée. Il faut ensuite imprimer la traduction en français mot à mot du texte anglais. Si un mot anglais n'a pas d'équivalent dans le dictionnaire, il suffit de retranscrire le mot anglais tel quel. Il faut traiter les exceptions qui peuvent se produire dans le traitement du fichier en imprimant un message d'erreur.

Exemple de contenu du fichier `dict.txt`:

```
garcon boy
fille girl
table table
porte door
fenetre window
le the
est is
sont are
pres near
sous under
sur over
vieux old
jeune young
et and
```

Exemple d'un fichier à traduire `story.txt`:

```
the young girl is near the door
the boy is under the table
the window and the table and the door are old and beautiful
```

Résultat de l'exécution:

```
% java Translate dict.txt story.txt
le jeune fille est pres le porte
le garcon est sous le table
le fenetre et le table et le porte sont vieux et beautiful
```

Pour vous simplifier la vie, vous pouvez écrire le programme à l'intérieur de la méthode suivante: `public static void main(String [] args)` method:

```
import java.util.*;
import java.io.*;
class Translate {
    public static void main (String [] args) {
    // ... insérer du code ici
    }
}
```

L'évaluation est répartie comme suit:

(15) (a) un programme qui fonctionne

(5) (b) traitement des exceptions

(5) (c) choix approprié des structures de données

(5) (d) traitement de fichier

Rappel

Entrées-sorties `BufferedReader in = new BufferedReader(new FileReader(fnDict));`
`BufferedReader` contient la méthode `readLine()` qui retourne une ligne comme un `String` ou `null` lorsqu'on est à la fin du fichier.

Tokenization `StringTokenizer st = new StringTokenizer(l);`
`StringTokenizer` contient deux méthodes

`hasMoreTokens()` qui retourne `true` s'il reste des tokens à traiter

`nextToken()` qui retourne le prochain token comme une `String`

Exceptions `FileReader` peut lever `FileNotFoundException` et `BufferedReader` peut lever `IOException`; les deux doivent être traitées.

Solution 3

```
import java.util.*;
import java.io.*;
class Translate {
    public static void main (String [] args) {
        String fnDict=args[0];
        String fnText=args[1];
        TreeMap dict=new TreeMap();
        try {
            String l;
            BufferedReader in=null;
            //dictionary
            in = new BufferedReader(new FileReader(fnDict));
            while((l=in.readLine())!=null && l.trim().length()>0){
                StringTokenizer st = new StringTokenizer(l);
                String en=st.nextToken();
                String fr=st.nextToken();
                dict.put(fr,en);
            }
            in.close();
        } catch (FileNotFoundException e) {
            System.out.println("Dictionary " + fnDict + " not found");
            System.exit(0);
        } catch (IOException e)
            System.out.println("IO Exception reading " + fnDict);
            System.exit(0);
        }
        try {
            String l;
            BufferedReader in=null;
            in = new BufferedReader(new FileReader(fnText));
            while((l=in.readLine())!=null && l.trim().length()>0){
                StringTokenizer st = new StringTokenizer(l);
                while (st.hasMoreTokens()) {
                    String fr = st.nextToken();
                    String en = (String)dict.get(fr);
                    if (en!=null)
                        System.out.print(en+ " ");
                    else
                        System.out.print(fr+ " ");
                }
                System.out.println("");
```

```
            }
            in.close();
        } catch (FileNotFoundException e) {
            System.out.println("Text " + fnText + " not found");
            System.exit(0);
        } catch (IOException e) {
            System.out.println("IO Exception reading " + fnText);
            System.exit(0);
        }
    }
}
```

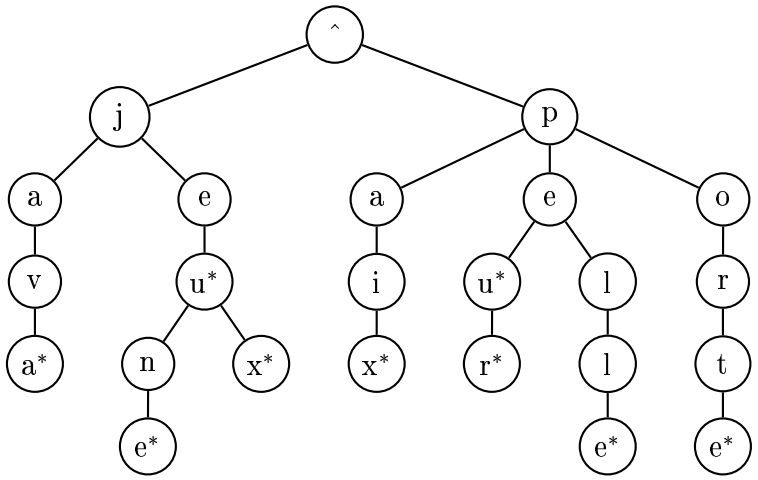

Figure 2: Un trie contenant les mots *java, jeu, jeune, jeux, paix, porte, peu, peur, pelle*

(40) 4. Un *trie* est une structure d'arbre spéciale où chaque noeud peut avoir jusqu'à 26 enfants (un par lettre de l'alphabet). Un *trie* peut être utilisé pour implanter un correcteur d'orthographe en conservant les mots du dictionnaire dans le trie et en les cherchant par la suite. Un mot apparaît dans le dictionnaire quand sa recherche termine à un noeud acceptant (montré avec $*$ dans la figure 2).

Voici une implantation partielle d'un trie.

```
import java.util.*;
public class Trie {
    private TrieNode root;
    public Trie () { root = new TrieNode('.');}
    public void insert (String s){ root.insert(s.toLowerCase(),0); }
    public boolean checkSpelling(String s) { return root.checkSpelling(s.toLowerCase(
    public void printTrie(){ root.printTrie(0);}
    public int countLetters() { return root.countLetters();}
    public int countWords() { return root.countWords(); }
}
class TrieNode {
    protected final int LCOUNT = 26;
    protected TrieNode [] children = new TrieNode[26];
    protected int childCount=0;
    protected char element;
    protected boolean isWord=false;
    public TrieNode (char charVal) {
        for (int i=0;i<LCOUNT;i++) {
           children[i]=null;
```

```
            }
            element=charVal;
        }
        public void printTrie(int depth) {
            if (element!='.') {
                if (isWord) {
                    System.out.print(""+Character.toUpperCase(element));
                } else {
                    System.out.print(""+element);
                }
            }
            boolean indent=false;
            for (int i=0;i<LCOUNT;i++) {
                if (children[i]!=null) {
                    if (indent) {
                        for (int d=0;d<depth;d++) {
                            System.out.print(".");
                        }
                    }
                    children[i].printTrie(depth+1);
                    indent=true;
                }
            }
            if (childCount==0) System.out.println("");
        }
    }
```

(10) (a) Qu'imprime l'appel `public void printTrie()` dans `Trie` pour le trie de la figure 2. Écrivez lisiblement votre réponse.

(10) (b) Implanter `public int countWords()` dans `TrieNode` qui compte les mots dans le trie.

(10) (c) Implanter `public void insert(String s, int x)` dans `TrieNode` qui insère `String s` dans le trie. Expliquer la nécessité du deuxième paramètre `int x`.

(10) (d) Implanter `public boolean checkSpelling(String s, int x)` dans `TrieNode`. Cette méthode retourne `true` si `s` apparaît dans le trie et `false` sinon.

Rappel

La méthode `toUpperCase()` dans `String` retourne la version majuscule d'une chaîne.

La méthode `toLowerCase()` dans `String` retourne la version minuscule d'une chaîne.

La difference entre `System.out.println(String s)` et `System.out.print(String s)` est que la deuxième méthode n'imprime pas une nouvelle ligne après `String s`.

Solution 4

```
javA
.eUnE
...X
paX
.ellE
..UR
.ortE

public int countWords() {
    int sz=0;
    if (isWord) sz++;
    for (int i=0;i<LCOUNT;i++) {
        if (children[i]!=null) {
            sz+=children[i].countWords();
        }     }
    return sz;
}

public void insert (String word, int pos) {
    if (pos==word.length()) {
        isWord=true;
        return;
    }
    int idx= word.charAt(pos)-'a';
    if (children[idx]==null) {
        children[idx]=new TrieNode(word.charAt(pos));
        childCount++;
    }
    children[idx].insert(word,++pos);
}


public boolean checkSpelling(String word, int pos) {
    if (pos==word.length()) return isWord;
    int idx= word.charAt(pos)-'a';
    if (children[idx]==null) return false;
    return children[idx].checkSpelling(word,++pos);
}
```