Examen final - solutionnaire mini révision en classe

1. Listes chainées

Dites ce que la méthode suivante affiche à la console :

A)

```
public static void traceListe () {

   Maillon<Integer> m1 = new Maillon<Integer>
                                 (2, new Maillon<Integer>(4));
   Maillon<Integer> m2 = new Maillon<Integer>(8, m1.suivant());
   Maillon<Integer> m3 = new Maillon<Integer>(5);
   m1.modifierSuivant(new Maillon<Integer>(7));
   m2.suivant().modifierSuivant(m1.suivant());
   Maillon<Integer> m4 = m2.suivant();
   m4.suivant().modifierSuivant(m4);
   m4.suivant().modifierInfo(6);
   m3.modifierSuivant(m2.suivant().suivant());
   m2.modifierSuivant(m3);
   m1.modifierSuivant(m2);
   m4.modifierSuivant(null);

   while (m1 != null) {
       System.out.print(m1.info() + "  ");
       m1 = m1.suivant();
   }
}
```

Réponse : 2 8 5 6 4

B)

```
public static void traceListe2() {

   Maillon<Integer> m = new Maillon<>(5);
   Maillon<Integer> m2 = m;
   Maillon<Integer> m4 = new Maillon<>(9, new Maillon<>(8));
   Maillon<Integer> m5 = m4.suivant();
   Maillon<Integer> m3;

   for (int i = 1; i < 5; i++) {
      m2.modifierSuivant(new Maillon<>(i));
      m2 = m2.suivant();
   }

   m3 = m.suivant();
   m.modifierSuivant(m2);
   m2 = m3.suivant();
   m.suivant().modifierSuivant(m2.suivant());
   m2.suivant().modifierSuivant(m2);
   m3.modifierInfo(6);
   m2.modifierSuivant(m3);
   m3.modifierSuivant(m);
   m5.modifierSuivant(new Maillon<>(7));
```

Mélanie Lord
lord.melanie@uqam.ca

```
    m5.suivant().modifierSuivant(m3);
    m4 = new Maillon<>(10, m4);
    m2.modifierSuivant(new Maillon<>(1));

    while (m4 != null) {
        System.out.print(m4.info() + " ");
        m4 = m4.suivant();
    }
}
```

Réponse : 10 9 8 7 6 5 4 3 2 1

2. Techniques de recherche

Soit le tableau suivant :

```
[ 8, 11, 12, 15, 22, 31, 42, 44, 47, 50, 51, 56, 58, 59, 63, 65, 70, 72, 81, 87 ]
```

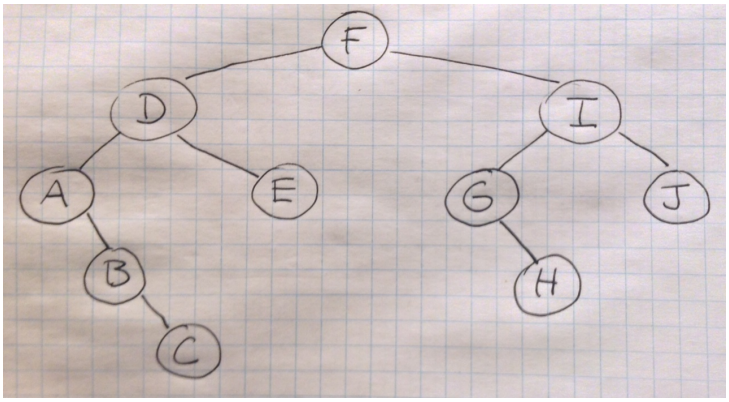
Donnez les éléments consultés lors de la recherche du nombre 13, en effectuant une :

- Recherche séq. non ordonnée : 8 11 12 15 22 31 42 44 47 50 51 56 58 59 63 65 70 72 81 87
- Recherche séq. ordonnée : 8 11 12 15
- Recherche binaire : 50 22 11 12 15

3. ABR

A) Dessinez l'arbre binaire de recherche construit par la suite d'instructions suivantes :

```
TdaAbr<Character> abr =
            new ArbreBinaireRecherche<Character>();
abr.inserer('F');
abr.inserer('D');
abr.inserer('I');
abr.inserer('E');
abr.inserer('G');
abr.inserer('J');
abr.inserer('A');
abr.inserer('B');
abr.inserer('C');
abr.inserer('H');
```



B) Dans l'arbre binaire construit en A), donnez la suite d'éléments obtenue en faisant le parcours :

- préfixe : FDABCEIGHJ
- infixe : ABCDEFGHIJ
- postfixe : CBAEDHGJIF

Mélanie Lord
lord.melanie@uqam.ca

C) Codez les méthodes suivantes qui retournent une chaine de caractères contenant les éléments du parcours préfixe, infixe et postfixe.

```
public static String contenuPrefixe(Noeud<Character> racine) {
   String s = "";
   if (racine != null) {
      //traitement racine
      s = racine.info() + "";
      //aller a gauche
      s = s + contenuPrefixe(racine.gauche());
     //aller a droite
      s = s + contenuPrefixe(racine.droite());
   }
   return s;
}

public static String contenuInfixe(Noeud<Character> racine) {
   String s = "";
   if (racine != null) {
      //aller a gauche
      s = contenuInfixe(racine.gauche());
      //traitement racine
      s = s + racine.info();
      //aller a droite
      s = s + contenuInfixe(racine.droite());
   }
   return s;
}


public static String contenuPostfixe(Noeud<Character> racine) {
   String s = "";
   if (racine != null) {
      //aller a gauche
      s = contenuPostfixe(racine.gauche());

      //aller a droite
      s = s + contenuPostfixe(racine.droite());

      //traitement racine
      s = s + racine.info();
   }
   return s;
}
```

D) Codez la méthode récursive suivante :

```
/**
 * Compte le nombre de feuilles de l'arbre donne par racine.
 * @param racine la racine de l'arbre dont on veut calculer le nombre de
 *               feuilles.
 * @return le nombre de feuilles de l'arbre dont la racine est donne en
 *         parametre.
 */
public static int nbrFeuilles (Noeud racine) {
   int nbr = 0;
   if (racine != null) {
      if (racine.gauche() == null && racine.droite() == null) {
         nbr = 1;
      }
      nbr = nbr + nbrFeuilles(racine.gauche())
                + nbrFeuilles(racine.droite());
   }
   return nbr;
}
```

E) Dites en vos propres mots ce que fait la méthode suivante :

```
public static int mystere (Noeud<Integer> racine, int i) {
   int res;
   if (racine != null)  {
      if (!racine.info().equals(i)) {
         if (racine.info().compareTo(i) < 0) {
            res = mystere(racine.droite(), i);
         } else {
            res = mystere(racine.gauche(), i);
         }
         if (res != -1) {
            res = res + 1;
         }
      } else {
         res = 0;
      }
   } else {
      res = -1;
   }
   return res;
}
```

Cette méthode retourne la longueur du chemin du noeud racine au noeud contenant l'élément i. Si i n'est pas dans l'arbre, elle retourne -1.

Mélanie Lord
lord.melanie@uqam.ca

4. Tris

En regardant la première passe (en bleu) des tris suivants, dites de quel tri il s'agit (sélection, insertion, à bulles ou Quicksort) et complétez la passe 2.

A) Tri à bulles

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
50	11	81	77	22	31	42	9	23	54	3	34	72	59	10	12	70	35	18
11	50	77	22	31	42	9	23	54	3	34	72	59	10	12	70	35	18	81
11	50	22	31	42	9	23	54	3	34	72	59	10	12	70	35	18	77	81

B) Tri sélection

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
50	11	81	77	22	31	42	9	23	54	3	34	72	59	10	12	70	35	18
3	11	81	77	22	31	42	9	23	54	50	34	72	59	10	12	70	35	18
3	9	81	77	22	31	42	11	23	54	50	34	72	59	10	12	70	35	18

C) Tri Quicksort

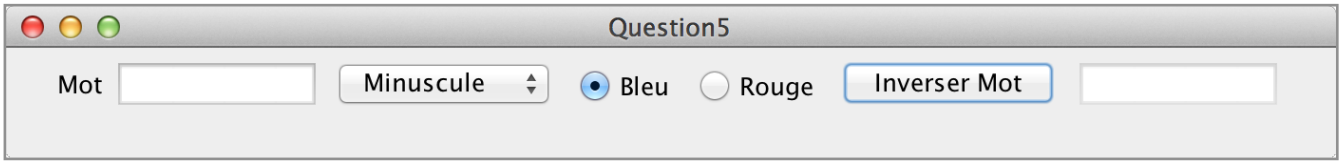
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
50	11	81	77	22	31	42	9	23	54	3	34	72	59	10	12	70	35	18
10	11	18	35	22	31	42	9	23	12	3	34	50	59	72	54	70	77	81
9	3	10	35	22	31	42	18	23	12	11	34	50	59	72	54	70	77	81

D) Tri insertion

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
50	11	81	77	22	31	42	9	23	54	3	34	72	59	10	12	70	35	18
11	50	81	77	22	31	42	9	23	54	3	34	72	59	10	12	70	35	18
11	50	81	77	22	31	42	9	23	54	3	34	72	59	10	12	70	35	18

5. Interface graphique

Créez la fenêtre suivante (taille 700 x 80 pixels) et ajoutez les fonctionnalités décrites ci-dessous.



Le champ Mot permet à l'utilisateur d'entrer une chaine de caractères et est vide à l'initialisation de l'application. Par défaut, les caractères entrés dans ce champ sont bleus (`Color.BLUE`).

La liste déroulante contient les valeurs "Minuscule" et "Majuscule" et la valeur sélectionnée par défaut est "Minuscule". À la sélection d'une valeur dans cette liste, la chaine dans le champ Mot se transforme en minuscules ou en majuscules, selon le choix fait.

Les boutons radio permettent de sélectionner la couleur des caractères entrés dans le champ Mot. Par défaut, le bouton sélectionné est Bleu. Au clic d'un de ces boutons, la couleur du texte dans le champ Mot change (s'il y a lieu) selon la couleur sélectionnée. Seulement un des deux boutons peut être sélectionné.

Le bouton Inverser Mot permet d'inverser le mot entré dans le champ Mot.

Au clic de ce bouton :

Si le champ Mot est vide, le champ à la droite du bouton Inverser Mot est effacé (s'il contenait quelque chose) et le message d'erreur suivant est affiché :

Si le champ contient une chaine de caractères non vide, celle-ci est affichée, à l'envers, dans le champ à la droite du bouton Inverser Mot (en noir).

Le champ à droite du bouton Inverser Mot est actif, mais ne doit pas pouvoir être modifié.

SOLUTION :

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

public class Question5 implements ActionListener {
   //constante
   private final String [] tabChoixCasse = {"Minuscule", "Majuscule"};
   private final String MSG_ERR = "Le champ Mot doit contenir un mot à inverser.";

   //attributs d'instance
   private JFrame fenetre = new JFrame();
   private JLabel jLMot = new JLabel("Mot");
   private JTextField jTFMot = new JTextField(8);  //8 colonnes
   private JComboBox jCBChoixCasse = new JComboBox(tabChoixCasse);
   private JRadioButton jRBleu = new JRadioButton("Bleu");
   private JRadioButton jRRouge = new JRadioButton("Rouge");
   private ButtonGroup groupeCouleurs = new ButtonGroup();
   private JButton jBInverser = new JButton("Inverser Mot");
   private JTextField jTFMotInverse = new JTextField(8);

   public Question5 () {
      init();
   }

   private void init() {

      fenetre.setLayout(new FlowLayout());
      fenetre.setBounds(400, 400, 700, 80);
      fenetre.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      fenetre.setTitle("Question5");

      jTFMot.setForeground(Color.BLUE);
      jRBleu.setSelected(true);
      groupeCouleurs.add(jRBleu);
      groupeCouleurs.add(jRRouge);
      jTFMotInverse.setEditable(false);

      fenetre.getContentPane().add(jLMot);
      fenetre.getContentPane().add(jTFMot);
      fenetre.getContentPane().add(jCBChoixCasse);
      fenetre.getContentPane().add(jRBleu);
      fenetre.getContentPane().add(jRRouge);
      fenetre.getContentPane().add(jBInverser);
      fenetre.getContentPane().add(jTFMotInverse);
      fenetre.setVisible(true);

      //ajout des ecouteurs
      jCBChoixCasse.addActionListener(this);
      jRBleu.addActionListener(this);
      jRRouge.addActionListener(this);
      jBInverser.addActionListener(this);
   }

   public void actionPerformed (ActionEvent e) {
      if (e.getSource() == jCBChoixCasse) {
         choixCasse();
      } else if (e.getSource() == jRBleu) {
```

```
            jTFMot.setForeground(Color.BLUE);
        } else if (e.getSource() == jRRouge) {
            jTFMot.setForeground(Color.RED);
        } else {  //bouton inverser
            inverserMot();
        }
    }

    private void choixCasse() {
        String texte = jTFMot.getText();
        if (jCBChoixCasse.getSelectedItem().equals(tabChoixCasse[0])) {
            jTFMot.setText(texte.toLowerCase());
        } else {
            jTFMot.setText(texte.toUpperCase());
        }
    }

    private void inverserMot() {
        //valider mot
        String mot = jTFMot.getText();
        if (mot.isEmpty()) {
            JOptionPane.showMessageDialog(fenetre, MSG_ERR, "ERREUR",
                                                    JOptionPane.ERROR_MESSAGE);
            jTFMotInverse.setText("");
        } else {
            jTFMotInverse.setText(motInverse(mot));
        }
    }

    private String motInverse (String mot) {
        String inverse = "";
        for (int i = mot.length() - 1 ; i >= 0 ; i--) {
            inverse = inverse + mot.charAt(i);
        }
        return inverse;
    }

    public static void main (String [] args) {
        new Question5();
    }

}
```