Examen (2 heures, avec document)
Corrigé

Préambule : Répondre de manière concise et précise aux questions. Ne pas mettre de commentaires de documentation sauf s'ils sont nécessaires à la compréhension.

Il est conseillé de répondre directement dans le sujet quand c'est possible. Sinon, il est conseillé de mettre une marque sur le sujet (par exemple le numéro de l'exercice suivi d'une lettre majuscule : 1A, 1B, 2A, etc) et d'indiquer sur la copie la marque avec le texte (ou le code) associé.

Barème indicatif :

exercice	1	2	3	4
points	5	5	5	5

Exercice 1 Écrire une classe Main qui, étant donné le nom d'une classe donné comme premier argument de la ligne de commande, affiche dans l'ordre alphabétique le nom des attributs déclarés dans cette classe pour lesquels il existe au moins une méthode de même nom déclarée dans cette même classe.

Voici un exemple d'exécution (le texte de la classe Exemple1 est donnée au listing 1) :

```
> java Main Exemple1
a
d
z
>
```

Listing 1 – La classe Exemple1

```
1  class Exemple1 {
2      private double z;
3      private int a;
4      private char c;
5      public boolean d;
6
7      public void m() {}
8      void a(int c) {}
9      private void z() {}
10     void a() {}
11     void d(String s) {}
12 }
```

Solution :

```java
import java.lang.reflect.*;
import java.util.*;

class Main {

    public static void afficherNoms(String nomClasse) throws Exception {
        // Remarque : c'est une solution qui n'est pas la plus efficace...
        Class classe = Class.forName(nomClasse);
        SortedSet<String> nomsTrouves = new TreeSet<>();
        for (Method methode : classe.getDeclaredMethods()) {
            String nom = methode.getName();
            try {
                classe.getDeclaredField(nom);
                nomsTrouves.add(nom);
            } catch (NoSuchFieldException e) {
                // Pas d'attributs !
            }
        }

        for (String nom: nomsTrouves) {
            System.out.println(nom);
        }
    }

    public static void main(String[] args) throws Exception {
        assert args.length == 1;
        afficherNoms(args[0]);
    }

}
```

Listing 2 – L'interface Offre

```
1 public interface Offre {
2     int getNumero();
3     int getMontant();
4     double getTaux();
5 }
```

Financement participatif

Les exercices suivants, indépendants les uns des autres même s'ils traitent du même sujet, concernent une plateforme de financement participatif en se plaçant du point de vue d'un utilisateur anonyme de cette plateforme.

Exercice 2 : Les classes métier
Le diagramme de classe de la figure 1 présente les interfaces et les classes (une classe concrète par interface) constitutives de cette application.

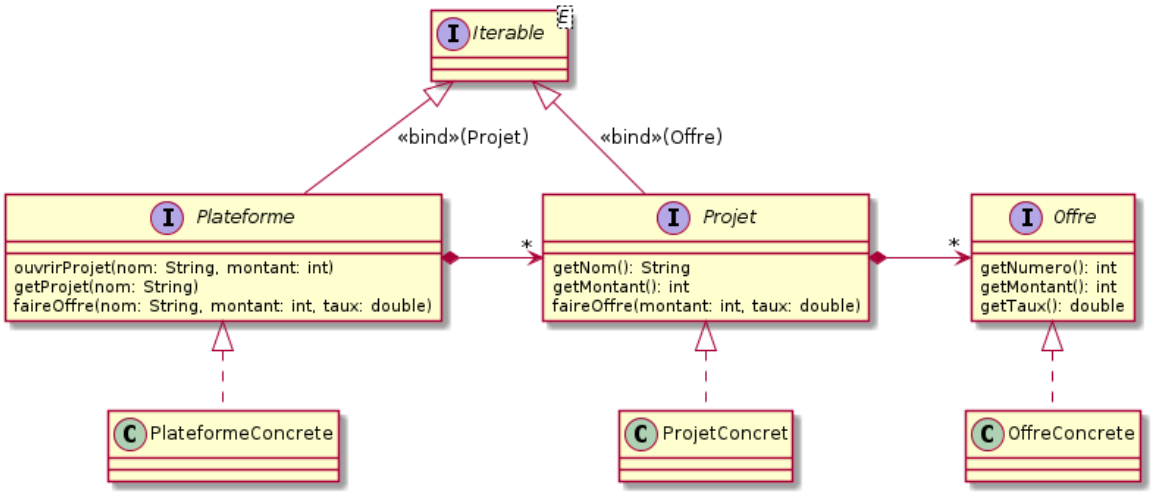

FIGURE 1 – Diagramme de classe de notre application

Une plateforme de financement participatif (de type *crowdlending*) permet à des individus de financer des entreprises en leur prêtant des sommes d'argent. Une **plateforme** (listing 4) propose plusieurs **projets** (listing 3). Chaque projet est identifié par son nom. Un projet demande à être financé pour un certain *montant* (entier, exprimé en euro). Pour chaque projet, des individus peuvent faire une ou plusieurs **offres** (listing 2) de prêt en choisissant le *montant* (au moins 20 euros) et un *taux* compris entre 4 % à 10 % (un réel). La plateforme applique des enchères inversées : seules les offres proposant le taux le plus faible seront retenues (dans la limite du montant demandé par l'entreprise).

2.1 *Exemple.* On considère la classe ExempleSujet (listing 5). Indiquer combien la plateforme propose de projets et combien d'offres recense chacun d'eux.
Solution : Il y a deux projets : XYZ et ABC. Le premier, XYZ, a 5 offres de prêt, le second 0.

Listing 3 – L'interface Projet

```java
public interface Projet extends Iterable<Offre> {
    /** Le nom du projet. */
    String getNom() ;

    /** Le montant demandé par le projet. */
    int getMontant();

    /** Ajouter un nouvelle offre. */
    void faireOffre(int montant, double taux);
}
```

Listing 4 – L'interface Plateforme

```java
public interface Plateforme extends Iterable<Projet> {
    /** Créer un nouveau projet à partir de son nom et du montant demandé. */
    Projet nouveauProjet(String nom, int montant);

    /** Enregistrer une offre de prêt pour un projet. */
    // On suppose que l'offre est possible.
    void faireOffre(String nom, int montant, double taux);

    /** Obtenir un projet à partir de son nom. */
    Projet getProjet(String nom);
}
```

Listing 5 – La classe ExempleSujet

```java
public class ExempleSujet {

    static public Plateforme getPlateformeSujet() {
        Plateforme exemple = new PlateformeConcrete();
        exemple.nouveauProjet("XYZ", 120000);
        exemple.faireOffre("XYZ", 50, 7.5);
        exemple.faireOffre("XYZ", 90, 8.3);
        exemple.faireOffre("XYZ", 20, 9.9);
        exemple.faireOffre("XYZ", 50, 5.0);
        exemple.faireOffre("XYZ", 68, 6.9);
        exemple.nouveauProjet("ABC", 30000);
        return exemple;
    }

    public static void main(String[] args) throws Exception {
        Plateforme exemple = getPlateformeSujet();
    }
}
```

Listing 6 – La classe OffreConcrete

```
1  public class OffreConcrete implements Offre {
2      private int numero;
3      private int montant;
4      private double taux;
5
6      public OffreConcrete(int numero, int montant, double taux) {
7          this.numero = numero;
8          this.montant = montant;
9          this.taux = taux;
10     }
11
12     public int getNumero() { return this.numero; }
13     public int getMontant() { return this.montant; }
14     public double getTaux() { return this.taux; }
15     public String toString() {
16         return String.format("%4d. %8d à%5.1f", this.numero, this.montant, this.taux);
17     }
18 }
```

2.2 *Projet.* La classe OffreConcrete est donnée au listing 6. Compléter la classe ProjetConcret (listing 7) sachant que :

1. les offres doivent être conservées dans une liste (`java.util.List`),
2. il est possible d'itérer sur les offres faites sur le projet (`Iterable<Offre>`).

Solution :

Listing 7 – La classe ProjetConcret

```
1  import java.util.*;
2
3  public class ProjetConcret implements Projet {
4      private String nom;
5      private int montant;
6      ... // les offres
7
8      public ProjetConcret(String nom, int montant) {
9          this.nom = nom;
10         this.montant = montant;
11         ...
12     }
13
14     public String getNom() { return this.nom; }
15     public int getMontant() { return this.montant; }
16
17     public void faireOffre(int montant, double taux) {
18         ...
19     }
20
21     public Iterator<Offre> iterator() {
22         ...
23     }
24
25     @Override public String toString() {
26         StringBuilder result = new StringBuilder();
27         result.append("Projet␣" + this.nom + "(" + this.montant + ")");
28         for (Offre offre : this) {
29             result.append("\n␣␣-␣" + offre);
30         }
31         return result.toString();
32     }
33 }
```

```java
import java.util.*;

public class ProjetConcret implements Projet {
    private String nom;
    private int montant;
    private List<Offre> offres;

    public ProjetConcret(String nom, int montant) {
        this.nom = nom;
        this.montant = montant;
        this.offres = new ArrayList<>();
    }

    public String getNom() { return this.nom; }
    public int getMontant() { return this.montant; }

    public void faireOffre(int montant, double taux) {
        int numero = this.offres.size() + 1;
        this.offres.add(new OffreConcrete(numero, montant, taux));
    }

    public Iterator<Offre> iterator() {
        return this.offres.iterator();
    }

    @Override public String toString() {
        StringBuilder result = new StringBuilder();
        result.append("Projet " + this.nom + "(" + this.montant + ")");
        for (Offre offre : this) {
            result.append("\n  - " + offre);
        }
        return result.toString();
    }
}
```

2.3 Compléter la classe PlateformeConcrete (listing 8).

Solution :

Listing 8 – La classe PlateformeConcrete

```
1  import java.util.*;
2
3  public class PlateformeConcrete implements Plateforme {
4      ...
5
6      public Projet nouveauProjet(String nom, int montant) {
7          ...
8      }
9
10     public void faireOffre(String nom, int montant, double taux) {
11         ...
12     }
13
14     public Projet getProjet(String nom) {
15         ...
16     }
17
18     public Iterator<Projet> iterator() {
19         ...
20     }
21
22     @Override public String toString() {
23         StringBuilder result = new StringBuilder();
24         for (Projet projet : this) {
25             result.append("\n" + projet);
26         }
27         return result.toString();
28     }
29 }
```

```java
import java.util.*;

public class PlateformeConcrete implements Plateforme {

    private Map<String, Projet> projets = new TreeMap<>();

    public Projet nouveauProjet(String nom, int montant) {
        assert ! this.projets.containsKey(nom);
        Projet nouveau = new ProjetConcret(nom, montant);
        this.projets.put(nom, nouveau);
        return nouveau;
    }

    public void faireOffre(String nom, int montant, double taux) {
        assert this.projets.containsKey(nom);
        this.projets.get(nom).faireOffre(montant, taux);
    }

    public Projet getProjet(String nom) {
        assert this.projets.containsKey(nom);
        return this.projets.get(nom);
    }

    public Iterator<Projet> iterator() {
        return this.projets.values().iterator();
    }

    @Override public String toString() {
        StringBuilder result = new StringBuilder();
        for (Projet projet : this) {
            result.append("\n" + projet);
        }
        return result.toString();
    }
}
```

Exercice 3 : Interface utilisateur

On souhaite proposer à l'utilisateur l'interface présentée à la figure 2. Elle permet à un utilisateur de faire une nouvelle offre de prêt sur un projet donné. En haut, ligne rappelle le nom du projet et le montant demandé. Au centre, les différentes offres déjà faites sont affichées. En bas, l'utilisateur peut saisir le montant qu'il souhaite prêter et le taux. Il peut cliquer « annuler » pour renoncer à prêter ou « préter » pour faire une nouvelle offre qui sera ajoutée au projet pour le montant et le taux saisis.

3.1 Indiquer, en annotant la figure 2, les composants graphiques Swing utilisés pour définir cette interface utilisateur.

Solution : TODO...

3.2 Compléter le code du listing 9 pour construire l'interface de la figure 2.

On utilisera les méthodes Integer.parseInt et Double.parseDouble pour convertir une chaîne respectivement en un entier ou un réel. Ces méthodes lèvent l'exception NumberFormatException si la chaîne ne peut pas être convertie (mauvais format). Dans ce cas, on mettra la couleur de fond de la zone saisie en rouge pour montrer le problème à l'utilisateur (méthode setBackground

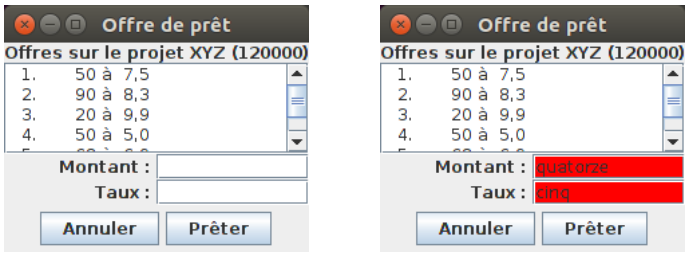

FIGURE 2 – Interface utilisateur souhaitée

de JTextField qui prend en paramètre une couleur, par exemple Color.RED pour le rouge).

Listing 9 – La classe OffreSwing pour l'interface utilisateur pour faire une Offre

```
1  import javax.swing.*;
2  import java.awt.*;
3  import java.awt.event.*;
4
5  public class OffreSwing {
6      final private Projet projet;
7      final JFrame fenetre;
8      final private JTextField valeurMontant = new JTextField(6);
9      final private JTextField valeurTaux = new JTextField(6);
10     final private JButton boutonAnnuler = new JButton("Annuler");
11     final private JButton boutonPreter = new JButton("Prêter");
12     final private JTextArea offres = new JTextArea();
13
14     public OffreSwing(Projet projet) {
15         this.projet = projet;
16         this.fenetre =  new JFrame("Offre de prêt");
17         offres.setEditable(false);
18
19         fenetre.pack();
20         fenetre.setVisible(true);
21     }
22 }
```

Solution :

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

public class OffreSwing {
    final private Projet projet;
    final JFrame fenetre;
    final private JTextField valeurMontant = new JTextField(6);
    final private JTextField valeurTaux = new JTextField(6);
    final private JButton boutonAnnuler = new JButton("Annuler");
    final private JButton boutonPreter = new JButton("Prêter");
    final private JTextArea offres = new JTextArea();

    public OffreSwing(Projet projet) {
        this.projet = projet;
        this.fenetre =  new JFrame("Offre de prêt");
        offres.setEditable(false);
        Container c = fenetre.getContentPane();
        c.setLayout(new BorderLayout());

        JLabel titre = new JLabel("Offres sur le projet " + projet.getNom()
                + " (" + projet.getMontant() + ")");
        c.add(titre, BorderLayout.NORTH);

        // Ajouter les offres
        for (Offre offre : projet) {
            offres.append(String.valueOf(offre) + "\n");
        }
        c.add(new JScrollPane(offres), BorderLayout.CENTER);


        JPanel paneauOffre = new JPanel(new BorderLayout());
        c.add(paneauOffre, BorderLayout.SOUTH);

        JPanel donnees = new JPanel(new GridLayout(2,2));
        paneauOffre.add(donnees, BorderLayout.NORTH);
        donnees.add(new JLabel("Montant :", SwingConstants.RIGHT));
        donnees.add(valeurMontant);
        donnees.add(new JLabel("Taux :", SwingConstants.RIGHT));
        donnees.add(valeurTaux);

        JPanel boutons = new JPanel(new FlowLayout());
        paneauOffre.add(boutons, BorderLayout.SOUTH);
        boutons.add(boutonAnnuler);
        boutons.add(boutonPreter);

        // Définir le comportement
        boutonAnnuler.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent ev) {
                fenetre.dispose();
            }
        });

        boutonPreter.addActionListener(new ActionPreter());

        MouseListener blanchir = new MouseAdapter() {
            @Override public void mousePressed(MouseEvent ev) {
```

```
59                  ((JTextField) ev.getSource()).setBackground(Color.WHITE);
60              }
61          };
62          valeurMontant.addMouseListener(blanchir);
63          valeurTaux.addMouseListener(blanchir);
64
65          fenetre.pack();
66          fenetre.setVisible(true);
67          fenetre.setDefaultCloseOperation(WindowConstants.DISPOSE_ON_CLOSE);
68      }
69
70      private class ActionPreter implements ActionListener {
71          @Override
72          public void actionPerformed(ActionEvent ev) {
73              // Récupérer le montant
74              boolean error = false;
75              int montant = 0;
76              try {
77                  montant = Integer.parseInt(valeurMontant.getText());
78                  if (montant < 20) {
79                      throw new NumberFormatException("montant doît être > 20");
80                  }
81              } catch (NumberFormatException e) {
82                  valeurMontant.setBackground(Color.RED);
83                  System.out.println(e.getMessage());
84                  error = true;
85              }
86
87              // Récupérer le montant
88              double taux = 0;
89              try {
90                  taux = Double.parseDouble(valeurTaux.getText());
91                  if (taux < 4.0 || taux > 10.0) {
92                      throw new NumberFormatException("taux doit être entre 4.0 et 10.0");
93                  }
94                  if (taux * 10 != Math.round(taux * 10)) {
95                      throw new NumberFormatException("Taux : un seul chiffre après la virgule");
96                  }
97              } catch (NumberFormatException e) {
98                  valeurTaux.setBackground(Color.RED);
99                  System.out.println(e.getMessage());
100                 error = true;
101             }
102
103             // Ajouter une offre
104             if (! error) {
105                 projet.faireOffre(montant, taux);
106                 System.out.println(projet);
107                 fenetre.dispose();
108             }
109         }
110     }
111
112     public static void main(String[] args) {
113         EventQueue.invokeLater(new Runnable() {
114             public void run() {
115                 Plateforme exemple = ExempleSujet.getPlateformeSujet();
116                 System.out.println(exemple.getProjet("XYZ"));
```

```
117                     new OffreSwing(exemple.getProjet("XYZ"));
118                 }
119             });
120         }
121     }
```

...continue sur la page suivante...

Listing 10 – Le fichier crowdlending.dtd

```
<?xml version="1.0" encoding="UTF-8"?>

<!ELEMENT crowdlending (projet+)>
<!ELEMENT projet (offre*)>
<!ATTLIST projet
    montant CDATA #REQUIRED
    nom CDATA #REQUIRED>
<!ELEMENT offre EMPTY>
<!ATTLIST offre
    montant CDATA #REQUIRED
    taux CDATA #REQUIRED
    numéro CDATA #REQUIRED
    état CDATA #REQUIRED>
```

Listing 11 – Le document XML exemple-sujet.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE crowdlending SYSTEM "crowdlending.dtd">

<crowdlending>
    <projet nom="XYZ">
    <montant>120000</montant>
    <offre numéro="1" montant="50" taux="7.5" état="EN-COURS" />
    <offre numéro="2" montant="120" taux="10.0" état="EN-COURS" />
    <offre numéro="3" montant="20" taux="9.9" état="EN-COURS" />
    </projet>
    <projet nom="ABC">
    <montant>30000</montant>
    </projet>
</crowdlending>
```

Exercice 4 : Sérialisation en XML

On considère la DTD du listing 10 et le document XML du listing 11.

4.1 Pourquoi le document XML du listing 11 n'est pas valide par rapport à la DTD (listing 10) ?

Solution : Parce que la DTD définit un attribut "montant" sur l'élément projet et pas d'élément « montant ».

Comment faire pour être sûr que les noms de projet sont bien tous différents ?

Solution : On peut mettre **ID** pour le type de l'attribut « nom » de l'élément « projet ».

Comment faire pour indiquer que les valeurs de l'attribut « état » de l'élément « offre » ne peut prendre que les valeurs « EN-COURS », « NON-RETENUE » ou « RETENUE » ?

Solution : Il faut remplacer le type **CDATA** de cet attribut par :

```
(EN-COURS | NON-RETENUE | RETENUE)
```

Indiquer les modifications à faire à la DTD pour que le document soit valide.

Solution : DTD corrigée :

```
<?xml version="1.0" encoding="UTF-8"?>

```

Listing 12 – L'interface Serialiseur

```
1  public interface Serialiseur {
2
3      /** Sérialiser les données de la plateforme en XML sur le flot out. */
4      void serialiser(Plateforme plateforme, java.io.Writer out)
5              throws java.io.IOException;
6  }
```

Listing 13 – La classe SerialiseurConcret

```
1  import org.jdom.*;
2  import org.jdom.output.*;
3
4  public class SerialiseurConcret implements Serialiseur {
5
6      public void serialiser(Plateforme plateforme, java.io.Writer out)
7              throws java.io.IOException
8      {
9          // Création du document XML
10         Element racine = new Element("crowdlending");
11         DocType docType = new DocType("crowdlending", "crowdlending.dtd");
12         Document document = new Document(racine, docType);
13
14         ...
15
16         // Production du fichier XML
17         XMLOutputter sortie = new XMLOutputter(Format.getPrettyFormat());
18         sortie.output(document, out);
19     }
20 }
```

```
3  <!ELEMENT crowdlending (projet+)>
4  <!ELEMENT projet (montant, offre*)>
5  <!ATTLIST projet
6      nom ID #REQUIRED>
7  <!ELEMENT montant (#PCDATA)>
8  <!ELEMENT offre EMPTY>
9  <!ATTLIST offre
10     montant CDATA #REQUIRED
11     taux CDATA #REQUIRED
12     numéro CDATA #REQUIRED
13     état (EN-COURS | NON-RETENUE | RETENUE) #REQUIRED>
```

4.2 Compléter le code du listing 13 qui réalise l'interface Serialiseur (listing 12) et qui permet de produire un document XML pour une plateforme donnée.

Solution :

```java
import org.jdom.*;
import org.jdom.output.*;

public class SerialiseurConcret implements Serialiseur {

    public void serialiser(Plateforme plateforme, java.io.Writer out)
            throws java.io.IOException
    {
        // Création du document XML
        Element racine = new Element("crowdlending");
        DocType docType = new DocType("crowdlending", "crowdlending.dtd");
        Document document = new Document(racine, docType);

        for (Projet projet : plateforme) {
            // Création de l'élément 'projet"
            Element projetElt = new Element("projet");
            racine.addContent(projetElt);
            projetElt.setAttribute("nom", projet.getNom());

            // Création du fils 'montant'
            Element montantElt = new Element("montant");
            projetElt.addContent(montantElt);
            montantElt.setText("" + projet.getMontant());

            // Création des fils 'offre'
            for (Offre offre : projet) {
                projetElt.addContent(new Element("offre")
                        .setAttribute("numéro", "" + offre.getNumero())
                        .setAttribute("montant", "" + offre.getMontant())
                        .setAttribute("taux", "" + offre.getTaux())
                        );
            }
        }


        // Production du fichier XML
        XMLOutputter sortie = new XMLOutputter(Format.getPrettyFormat());
        sortie.output(document, out);
    }
}
```