1. (1 pt) Soit le code suivant :

```
public class Note {
    public enum Designation { C, D, E, F, G, A, B }
    public enum Modification { AUCUNE, BEMOL, DIESE }

    private Designation  _designation;
    private int          _octave;
    private Modification _modification;
}
```

Écrivez une méthode `equals` qui retourne `true` si deux notes sont identiques, `false` sinon. Deux notes sont identiques si les trois champs de la classe contiennent les mêmes valeurs.

```
public boolean equals( Object a_objet ) {
    boolean resultat = false;

    if( a_object instanceof Note ) {
        Note n = (Note) a_object;
        resultat = _designation == n._designation &&
                   _octave == n._octave &&
                   _modification == n._modification;
    }

    return resultat;
}
```

2. Soit les classes et interfaces suivantes :

```
interface Iterable< E >
interface Collection< E > implements Iterable< E >
interface List< E > implements Collection< E >

class AbstractCollection< E > implements Collection< E >
class AbstractList< E > extends AbstractCollection< E > implements List< E >
class AbstractSequentialList< E > extends AbstractList< E >
class LinkedList< E > extends AbstractSequentialList< E >
```

   (a) ($\frac{1}{2}$ pt) La classe `LinkedList< E >` contient la méthode `addAll( Collection< E > c)`. Encerclez les classes se qualifient comme argument à cette méthode.

   - `Object` : non
   - `AbstractCollection< E >` : oui
   - `AbstractSequentialList< E >` : oui
   - `LinkedList< E >` : oui

   (b) ($\frac{1}{2}$ pt) Nous écrivons une méthode ayant la signature suivante :

   `m1( LinkedList< List< E > > c)`

   Encerclez les classes se qualifient comme argument à cette méthode.

   - `LinkedList< Object >` : non
   - `LinkedList< AbstractCollection< E > >` : non
   - `LinkedList< AbstractSequentialList< E > >` : oui
   - `LinkedList< LinkedList< E > >` : oui