Questions de révision pour l'examen intra.

1. Expliquez ce que veut dire chacun des mots réservés suivants :

abstract interface public private protected final this super static

2. Considérez les deux classes suivantes :

```
class A {
    public A () {
        x = 7;
    }
    public A (int y) {
        this.y = y;
    }
    private int x;
    private int y = 3;
} // A
```

```
class B extends A {
    public B ( int z ) {
        super ( 999 );
        this.z = 10;
    }
    private int z;
} // B
```

Dessinez les contenus des variables `a1`, `a2` et `b` après l'exécution des instructions suivantes :

```
A a1 = new A ();        A a2 = new A (15);        B b = new B (100);
```

3. Considérez les classes suivantes :

```
public class ElementGraphique {
    public void afficher() { // ____________________________________________
        System.out.println ( "de ElementGraphique." );
    }
    public void uneMethode ( ElementGraphique e ) { // ____________________________
        System.out.println ( "uneMethode de ElementGraphique" );
    }
}
public class Triangle extends ElementGraphique {
    public void uneMethode ( Object e ) { // ______________________________________
        System.out.println ( "uneMethode de Triangle" );
    }
}
public class Quadrilatere extends ElementGraphique {
    public void uneMethode ( ElementGraphique e ) { // ____________________________
        System.out.println ( "uneMethode de Quadrilatere" );
    }
}
public class Rectangle extends Quadrilatere {
    public void afficher() { // ___________________________
        System.out.println ( "de Rectangle." );
    }
    public void uneMethode ( Rectangle r ) { // ___________________
        System.out.println ( "uneMethode de Rectangle" );
    }
}
```

(a) Indiquez à côté de chacune des méhodes, s'il y a lieu, s'il s'agit d'une redéfinition ou d'une surdéfinition.

(b) Indiquez ce qui sera affiché après exécution des instructions suivantes :

```
Triangle         t = new Triangle();
Quadrilatere     q = new Rectangle();
ElementGraphique e = new Quadrilatere();
Rectangle        r = new Rectangle();
t.afficher();
```

```
        q.afficher();
        e.afficher();
        t.uneMethode(q);
        q.uneMethode(t);
        r.uneMethode(q);
```

4. Considérez les cinq classes suivantes :

```
public class Personne {

    private String nom;

    public Personne ( String nom ) {
        this.nom = nom;
        System.out.println ( "Je suis " + nom + ", " + ident());
  }

    public String ident () {
        return "une personne.";
    }

    public String toString() {
        return "le nom est " + nom;
    }
}

public class Etudiant extends Personne {

    private String code;

    public Etudiant ( String nom, String code ) {
        super ( nom );
        this.code = code;
    }

    public String ident () {
        return "un étudiant";
    }
}

public class Enseignant extends Personne {

    private String matricule;

    public Enseignant ( String nom, String matricule ) {
        super ( nom );
        this.matricule = matricule;
    }

    public String ident () {
        return "un enseignant";
    }

    public String getMatricule () {
        return matricule;
    }
}

public class Prof extends Enseignant {
```

```
    private String departement;

    public Prof ( String nom, String mat, String dept ) {
        super ( nom, mat );
        this.departement = dept;
    }

    public String ident () {
        return "un prof.";
    }
}

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

        // Partie (a)

        Personne toto = new Personne ( "x" );
        Personne totoEt = new Etudiant ( "y", "ZZ" );
        Enseignant totoEns = new Enseignant ( "z", "A" );
        Personne totoP = new Prof ( "w", "AA", "mth" );

        // Partie (b)

        Personne machin = new Prof ( "m", "X13", "math" );
        System.out.println ( machin.getMatricule() );

        Personne tintin = new Etudiant ( "tintin", "R24" );
        System.out.println ( ((Enseignant)tintin).getMatricule() );

        Prof tournesol = new Prof ( "tryphon", "H2O", "?" );
        System.out.println ( tournesol.getMatricule() + tournesol );

    }
}
```

(a) Donnez le résultat de l'exécution des 4 premières lignes de la méthode `main`.
(b) Pour chaque groupe de deux instructions qui suivent les 4 premières lignes de la méthode `main`, indiquez si ça **compile**. Si oui, donnez le **résultat de l'exécution**.

5. Écrivez votre code permanent sur les lignes prévues à cet effet :

```
String code = "  ____  ____  ____  ____  ____  ____  ____  ____  ____  ____  ____  ____ ";
                0     1     2     3     4     5     6     7     8     9     10    11
```

Veuillez indiquer le **contenu** de la liste **après chaque** instruction. Pour chaque instruction, vous devez utiliser la liste obtenue de l'instruction précédente.

```
ArrayList<Character> liste = new ArrayList<Character>();

liste.add ( code.charAt(2) );
liste.add ( 0, code.charAt(3) );
liste.add ( 2, code.charAt(0) );
liste.set ( 1, '*' );
liste.add ( code.charAt(5) );
liste.add ( 2, code.charAt(7) );
```

```
liste.remove ( (Character)'A' );
liste.add ( liste.remove ( 0 ) );
liste.set ( 2, liste.get ( 2 ) );
liste.add ( liste.indexOf ( 'D' ) + 1, code.charAt(7) );
```

6. Écrivez les caractères de votre code permanent sur chacun des tirets puis indiquez ce que l'appel de `questionPileFile ( s );` va afficher.

```
static String s = "___ ___ ___+ ___ * + ___ ___ ___ ___ + + * ___ * ___ * * + ___ * ___ + ";

                                          // contient 24 caractères dont 6 * et 6 +

public static void questionPileFile ( String s ) {
    Pile<Character> p = new PileArrayList<Character>();
    File<Character> f = new FileArrayList<Character>();
    for ( int i = 0; i < s.length(); ++i ) {
        char c = s.charAt(i);
        if ( c == '*' && !p.estVide() ) {
            System.out.print ( p.depiler() );
        } else if ( c == '+' && !f.estVide() ) {
            System.out.print ( f.defiler() );
        } else {
            p.empiler ( c );
            f.enfiler ( c );
        }
    }
}
```

7. Considérez les interfaces `Pile` et `File` vues au cours. Considérez que les implémentations suivantes existent : `PileConcrete` et `FileConcrete`. Supposez aussi que la méthode `taille` n'existe pas. Écrivez les méthodes suivantes :

```
/**
* Retourne une nouvelle pile copie identique à p.
* @param p pile à copier
*/
public static Pile copier ( Pile p );

/**
* Retourne une nouvelle file copie identique à f.
* @param f file à copier
*/
public static File copier ( File f );

/**
* Retourne une nouvelle pile ayant les mêmes éléments mais en ordre inverse.
* @param p pile à copier
*/
public static Pile copierInverse ( Pile p );

/**
* Retourne une nouvelle file ayant les mêmes éléments mais en ordre inverse.
* @param f file à copier
*/
public static File copierInverse ( File f );
```