EXAMEN
PROJET SYNTHESE en POO

Gestion d'un bunker après une invasion de zombies!

Voici les concepts à utiliser: l'encapsulation, la composition, l'héritage, le polymorphisme, les classes abstraites, les interfaces, les fichiers et plusieurs autres concepts vus dans les projets et les laboratoires nouveautés.

Voici l'organisation du personnel du bunker

<u>Forces de sécurité</u>:

Milice — *matricule, nom, prénom, sexe, date de naissance, vivant ou non, grade, arme, poste, année de service, nombre de plainte.*

Armée — *matricule, nom, prénom, sexe, date de naissance, vivant ou non, grade, arme, poste, année de service, nombre de sortie à l'extérieur, nombre de victime, matricule de tous les hommes sous ces ordres (un maximum de 5).*

<u>Gestionnaire</u>:

Scientifique — *matricule, nom, prénom, sexe, date de naissance, vivant ou non, nombre d'employé, secteur, nombre de projet, liste de tous les projets (max 3).*

Administrateur — *matricule, nom, prénom, sexe, date de naissance, vivant ou non, nombre d'employé, secteur, projet ou il est affecté, titre*

<u>Employé de Maintenance</u>:

Ingénieur — *matricule, nom, prénom, sexe, date de naissance, vivant ou non, secteur, spécialité, date de fin d'étude, niveau d'étude.*

Ouvrier — *matricule, nom, prénom, sexe, date de naissance, vivant ou non, secteur, quart de travail (minuit à 8, 8 à 4 ou 4 à minuit), genre de travail, nombre de secteur où il peut travailler.*

Concernant la date:
vous pouvez vous servir de la classe ClDate ou du GregorianCalendar.

<u>En premier lieu vous devez élaborer le diagramme des classes en UML de votre application et le faire approuver par votre enseignant avant de commencer la rédaction du projet.</u>

Votre application doit permettre la mise à jour de l'information des différentes ressources. Votre menu doit offrir les services suivants: (Ceci correspond à votre classe TestBunker).

Au démarrage du logiciel, *les données contenues dans le fichier* ***"bunker.ser"*** *devront être chargées dans une ArrayList contenant ainsi tous les survivants du bunker.*
Chargement automatique du fichier ArrayList ← bunker.ser

1- ***Ajouter un survivant;***
Saisissez des données pour chacune des classes instanciables.
Prévoir au moins 4 objets de chaque classe pour obtenir un résultat significatif.

2- ***Supprimer un survivant, tout en le laissant dans la base de données centrale;***

3- ***Modifier un survivant;***
Pour une modification vous pouvez supprimer une entrée existante et en créer une nouvelle ou modifier directement la ressource existante.

4- ***Rechercher et visualiser un survivant (objet) à partir de son matricule ou son nom.***
S'il existe, visualiser toutes les informations de celui-ci, sinon inscrire un message;

5- ***Afficher en mode console les humains (soit les 3 listes) suivants:***
la liste des Forces de sécurité en ordre de date de naissance,
la liste des Gestionnaire en ordre de date de naissance ainsi que
la liste des Employé de maintenance en ordre de date de naissance;

Tous les humains *doivent être* ***triés*** *de la date naissance la plus ancienne à la date de naissance la plus récente, en cas d'égalité de date,* ***trier les humains en ordre alphabétique de nom;***

6- *Écrire dans un* ***fichier texte*** *chacune des trois listes :*
le fichier ***"Forces de sécurité.txt"*** *pour la liste de toutes les Forces de sécurité,*
le fichier ***"Gestion.txt"*** *pour la liste de tous les Gestionnaires ainsi que*
le fichier ***"Employé de maintenance.txt"*** *pour la liste des Employés de maintenance;*

La première ligne de texte, pour chacun de ces fichiers, doit contenir son nom (soit le nom du fichier) ainsi que la date et l'heure de la création de celui-ci.
Veuillez disposer les données dans les fichiers texte de manière à faciliter la lecture des informations. Insérer: System.getProperty ("line.separator"); dans la méthode toString ( )).
Les fichiers sont créés dans le répertoire où se trouve l'exécutable de votre projet;

Lorsque vous quitterez l'application*, vous devez conserver le tout (soit la base de données – pour nous c'est l'ArrayList -- qui contient toutes les informations de tous les humains) dans un fichier objet* ***"bunker.ser". Cette classe doit implémenter l'interface Serializable****.*

Options supplémentaires à apporter au projet.
Ajoutez au moins 10 options au projet.

- *Développez le projet en environnement graphique (compte pour les 10 options).*
- *Utilisez JUnit pour effectuer des validations.*
- *Utilisez les REGEX pour un ou plusieurs attributs.*
- *Donnez la possibilité à l'utilisateur de choisir le nom de son fichier texte et du fichier serializable ainsi que le chemin d'accès pour sauvegarder ceux-ci.*
- *Ajoutez une ou des interfaces utiles au projet.*
- *Utilisez l'instruction enum pour un ou plusieurs attributs.*
- *Ajoutez des attributs et / ou des méthodes au projet.*
- *Utilisez quelques services offerts par la classe Collections qui pourraient être utiles au projet.*
- *Triez les humains sur d'autres champs que ceux demandés.*
  L'utilisateur pourrait même avoir l'opportunité de choisir le champ sur lequel il désire effectuer un tri. Utilisez l'interface Comparable et / ou Comparator.
- *Utilisez les try / catch pour effectuer des validations.*
- *Insérer des validations: pas de doublon sur un attribut particulier.*
- *Créez un exécutable du projet.*
- *Ajoutez une ou plusieurs nouveautés au projet qui n'est pas mentionné, mais qui devront être approuver par le professeur avant.*

Suggestion d'une procédure pour bien effectuer le projet

Voici quelques conseils sur la manière de réaliser le projet :

1. *Produisez la représentation **UML** de votre projet avec l'utilitaire que vous désirez.*

2. *Créez un projet en Java et définir toutes les classes du logiciel dans ce projet.*
   Au début, les classes sont des coquilles que vous allez remplir au fur et à mesure que le logiciel va se développer.

   Saisie des données pour chacune des classes instanciables.
   Prévoir au moins 4 objets pour chaque classe afin d'obtenir un résultat significatif.
   Voir la démarche des pages suivantes pour réaliser un projet à trois classes.

3. *Développez toutes les classes instanciables et abstraites.*
   Assemblez toutes les parties du projet (composition, héritage, classe Test, etc.).
   *Testez **une à une** chacune des classes instanciables dans la classe Test du projet.*

3. *Effectuez des ajouts, des suppressions, des recherches ainsi que des modifications.*

4. *Effectuez les tris nécessaires.*

5. *Produisez les trois fichiers texte.*
   Affichez ceux-ci en mode console et en fichier texte.

6. *Produisez le fichier biblio.ser permettant de charger et de sauvegarder l'ArrayList.*

À remettre

- le projet, soit tous les .java de toutes les classes du projet;
- le diagramme des classes du projet en UML;
- un document word indiquant toutes les options ajoutées au projet.

- ***Faire exécuter le projet, en présence de l'enseignant, avant de le remettre sur le Léa de celle-ci.***
- ***Tout projet non exécutable entraîne automatiquement la note 0.***
- ***Tout dossier vide ou fichiers manquants à votre nom sur Léa, entraîne automatiquement la note 0.***

Bon succès!

Saisie de données -- Tableau à ArrayList
Exemple 3 classes

public class Ville

```
{
protected String nom;
protected int    nbrHabitant;

public Ville ( )
        {
        nom        = "nil";
        nbrHabitant = 0;
        }

public Ville ( Ville obj )
        {
        nom        = obj.getNom ( );
        nbrHabitant = obj.getNbrHab ( );
        }

public Ville ( String vNom, int vNbrHab )
        {
        nom        = vNom;
        nbrHabitant = vNbrHab;
        }

public void setNom ( String vNom )
        {
        nom = vNom;
        }

public String getNom ( )
        {
        return nom;
        }

public void setNbrHab ( int vNbrHab )
        {
        nbrHabitant = vNbrHab;
        }

public int getNbrHab ( )
        {
        return nbrHabitant;
        }

public String toString ( )
        {
        return " nom= " + nom + " nombre habitants= " + nbrHabitant +
                System.getProperty ( "line.separator" );
        }}
import java.util.*;
```

```
public class BDVilles
{
protected ArrayList listeVilles;

public BDVilles ( )
        {
        listeVilles = new ArrayList ( );
        }


public BDVilles ( Ville tabVilles [ ] )
        {
        listeVilles  = new ArrayList ( ) ;
        for ( int ctr=0; ctr < tabVilles.length ; ctr++ )
              listeVilles.add ( tabVilles [ ctr ] );
        }


public ArrayList getListeVilles ( )
        {
        return listeVilles;
        }

public void setListeVilles ( ArrayList arrayV )
        {
         listeVilles = arrayV;
        }

public void ajouter ( Object obj )
        {
        Ville uneVille = ( Ville ) obj;
        listeVilles.add ( uneVille );
        }

public void supprimer ( Object obj )
        {
        Ville uneVille = ( Ville ) obj;
        listeVilles.remove ( uneVille );
        }

public void supprimer ( int position )
        {
        listeVilles.remove ( position );
        }
```

```
public void modifier ( int position, Object obj )
        {
        Ville uneVille = ( Ville ) obj;
        listeVilles.set ( position, uneVille );
        }


public int indexOf ( Object obj )
        {
        Ville uneVille = ( Ville ) obj;
        return listeVilles.indexOf ( uneVille );
        }


public Object lire ( Object obj )
        {
        Ville uneVille = ( Ville ) obj;
        return listeVilles.get ( this.indexOf ( uneVille ) );
        }


public Object lire ( int position )
        {
        return listeVilles.get ( position );
        }


public int size ( )
        {
        return listeVilles.size ( );
        }


public String toString ( )
        {
        return listeVilles.toString ( );
        }

}
```

```
public class TestArrayTableau

{

        private  static  Ville  donnees [ ]  =
            {
                  new Ville   ( "trois-Rivieres",  20000   ),
                  new Ville   ( "Sherbrooke",     50000   ),
                  new Ville   ( "Washington",    100000  ),
                  new Ville   ( "Québec",          400000  )
                  };

        private  static  BDVilles  uneBD   =   new  BDVilles  ( donnees );



        public  static  void  main ( String[ ]  args )
        {


        System.out.println  (   "\n uneBD = "    +    uneBD.toString  (   )  );

        Ville   ott    =   new  Ville   ( "Ottawa", 200000 );
        uneBD.ajouter ( new Ville (  "Montreal",   60000  )  );


        System.out.println  (  " La ville à la position 2 est: "   +   uneBD.lire  ( 2 )  );


        uneBD.supprimer ( 3 ) ;

        uneBD.ajouter ( ott );


        System.out.println (   "\n la position de ottawa est= "     +
                             (  uneBD.indexOf   ( ott  )   +   1   )    ) ;


        System.out.println  (  "\n"    +   uneBD.toString (  )  );


        }
}
```