مكتب التكوين المهني وإنعاش الشغل

OFPPT

Office de la Formation Professionnelle
et de la Promotion du Travail
Direction de la Recherche et Ingénierie de la Formation

Examen de fin de passage
session Juillet 2018

Filière : Techniques de Développement Informatique
Niveau: TS
Durée : 5 heures

Épreuve : Synthèse
Variante : V1
Barème : / 120pts

❖ Partie I : Théorie (40 pts)

➢ Dossier 1: L'essentiel en technologies de l'information (12 pts)

NB: la calculatrice est strictement interdite.

1. Complétez le tableau suivant : 6 pts

Binaire	Octal	Décimal	Hexadécimal
11000011			
	67		
		67	
			67

2. Soit la table de vérité suivante :

a	b	c	F
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

a. Donner la fonction F puis simplifier la **analytiquement.** **3 pts**

b. Simplifier F moyennant le tableau de Karnaugh. **3 pts**

➢ <u>Dossier 2:</u> Analyse et conception orientée objet (16 pts)

On souhaite développer une application permettant de passer des tests de connaissance concernant les pays du monde.

Chaque pays a un nom et appartient à un continent (Afrique, Asie, Europe, etc.).

Le pays contient des villes ayant un nom et une population (nombre d'habitants). Le pays a une ville capitale. Pour simplifier, on suppose que deux villes différentes n'ont ni le même nom ni la même population.

Les utilisateurs de l'application peuvent passer des tests de connaissance. Chaque test contient des questions du genre : « Quelle est la capitale de … ? ». Les questions sont à choix multiple, c'est-à-dire que pour chaque question nous avons 3 réponses (2 villes générées automatiquement par l'application en plus de la capitale). On retient pour la question la réponse de l'utilisateur et on enregistre pour le test la date de passation et on lui affecte un numéro automatique.

Pour pouvoir utiliser l'application (passer les tests et consulter les villes, les pays et les continents) l'utilisateur doit s'authentifier.

Les administrateurs de l'application ont pour rôle de gérer les continents, les pays et les villes. Ils peuvent, aussi, se servir de l'application comme les utilisateurs précédemment décrits.

La gestion des utilisateurs est faite par un super administrateur (il effectue, bien évidemment, les autres opérations d'administration).

Les opérations de gestion se font après authentification.

Chaque utilisateur a un email et un mot de passe pour se connecter.

1. Elaborer le diagramme de cas d'utilisation. **7.5 pts**
2. Elaborer le diagramme de classes. **8.5 pts**

➢ <u>Dossier 3:</u> Programmation structurée (12 pts)

Comme il est indiqué dans le dossier 2, une ville a un nom, une population et appartient à un pays (chaine de caractères).

Ecrivez un algorithme permettant de :

1. Déclarer la structure **Ville**. 1 pt
2. Déclarer un tableau de villes. 1 pt
3. Demander à l'utilisateur de remplir le tableau par **n** villes. 2 pts

   n doit être **supérieur strictement à 0** et **ne peut pas dépasser la taille maximale du tableau**. 1 pt
4. Afficher **la population totale d'un pays dont le nom est donné** par l'utilisateur. 3 pts

5. Afficher la ville la plus peuplée. **4 pts**

❖ Partie II: Pratique (80 pts)

➢ Dossier 1: Programmation structurée (16 pts)

1. Soit la fonction suivante :

```
int f(int nombre)
{
    int i = 1;

    while(1 == 1)
    {
        if(nombre / 10 == 0) return i;

        nombre = nombre / 10;
        i++;
    }
}
```

Quelle sera la valeur retournée par chacun des appels suivants :

a. f(275) **2 pts**

b. f(27) **2 pts**

c. f(2) **2 pts**

2. **En s'inspirant de la fonction précédente,** écrire la fonction **chiffre(int nombre, int place)** qui retourne le chiffre qui se trouve à la place spécifiée. **4 pts**

Exemples :

nombre	place	Chiffre à retourner
275	1	5
	2	7
	3	2
	4	0
	5	0

3. Ecrivez un programme permettant de :

a. Lire un entier e **compris entre 1 et 99999** jusqu'à ce que la réponse convienne. **2 pts**

b. Afficher, **en utilisant la fonction f**, un message indiquant si e appartient à l'un des intervalles suivants : [1, 9], [10, 99], [100, 999], [1000, 9999] ou [10000, 99999]. **4 pts**

Exemples :

- Si e = 2, on affiche : 2 appartient à l'intervalle [1, 9].
- Si e = 167, on affiche : 167 appartient à l'intervalle [100, 999].
- etc.

➢ <u>Dossier 2:</u> Programmation événementielle et orientée objet (64 pts)

Une ville est caractérisée par son nom et sa population (nombre d'habitants) ; la classe **Ville** est définie donc comme suit :

```
class Ville
{
    public string Nom;
    public int Population;

    public Ville(string nom, int population)
    {
        Nom = nom;
        Population = population;
    }

    public override string ToString()
    {
        return Nom + " (" + Population.ToString() + " habitants)";
    }
}
```

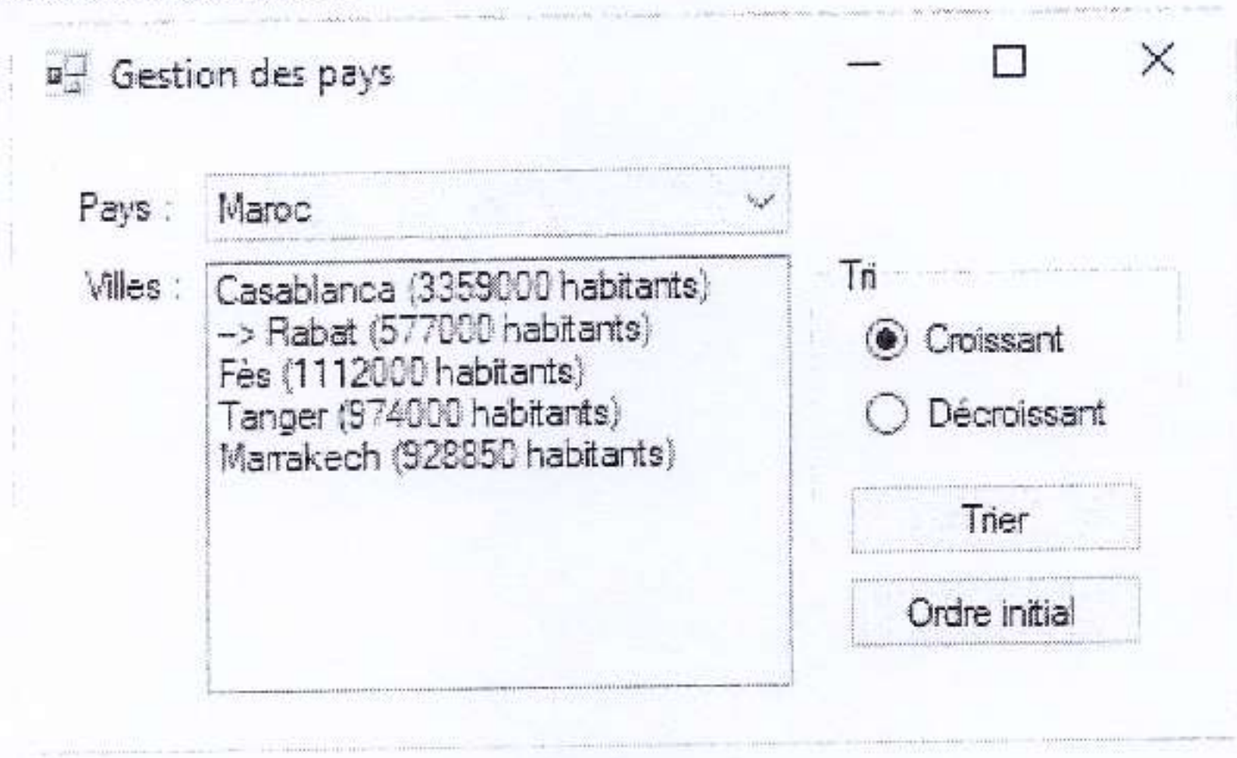
On souhaite créer la classe **Pays** qui se caractérise par son nom, la liste des villes et la ville capitale de ce pays.

1. Définir les attributs de la classe ainsi que leurs propriétés sachant que : 4.5 pts
   a. Le nom **ne peut pas contenir des chiffres** et doit comporter **au moins 4 caractères**, sinon **on génère une exception.** 3 pts
   b. La capitale **doit faire partie des villes**, sinon **elle faut l'ajouter.** 3 pts
2. Ajouter les constructeurs suivants :
   a. Un constructeur permettant d'initialiser le nom. 1.5 pt
   b. Un constructeur permettant d'initialiser le nom et la capitale. 2 pts
3. Ajouter la propriété **NombreVilles** permettant de retourner le nombre de villes du pays en cours. 2 pts
4. Ajouter la méthode **Rechercher(String nomVille)** renvoyant l'indice de la ville dont le nom est passé en paramètre s'elle appartient au pays en cours et **-1** sinon. 3 pts
5. Ajouter la méthode **Ajouter(Ville ville)** permettant d'ajouter la ville en paramètre à la liste s'elle n'existe pas auparavant. 3 pts
6. Ajouter la méthode **Supprimer(String nomVille)** permettant de supprimer la ville dont le nom est passé en paramètre. **S'elle s'agit de la capitale, Il faut remettre à null le champ Capital.** 4 pts

7. Ajouter la méthode **TrierParPopulation(String ordre) retournant la liste des villes** triée par population dans l'ordre croissant ou décroissant (le paramètre ordre prend la valeur "croissant" ou "décroissant"). **5 pts**

8. Ajouter la méthode **PopulationTotale()** retournant le nombre d'habitants du pays. **3 pts**

9. Soit le formulaire suivant :



Avec :

Contrôle	Type	Libellé
Maroc	ComboBox	comboBox_Pays
Casablanca (3359000 habitants) --> Rabat (577000 habitants) Fès (1112000 habitants) Tanger (974000 habitants) Marrakech (928850 habitants)	ListBox	listBox_Villes
Croissant	RadioButton	radioButton_Croissant
Décroissant	RadioButton	radioButton_Décroissant
Trier	Button	button_Trier
Ordre initial	Button	button_OrdreInitial

a. Déclarer une collection des objets **Pays**. **1 pt**

b. Ecrire le code s'exécutant au chargement qui permet de : **1 pt**

   i. Remplir la collection par les pays suivants : **6 pts**

Pays	Villes		Capitale
	Nom	Population	
Maroc	Casablanca	3359000	Rabat
	Rabat	577000	
	Fès	1112000	

	Tanger	974000	
Tunisie	-	-	-
Jordanie	Amman	994199	Amman
	Zarqa	395227	
	Irbid	250645	

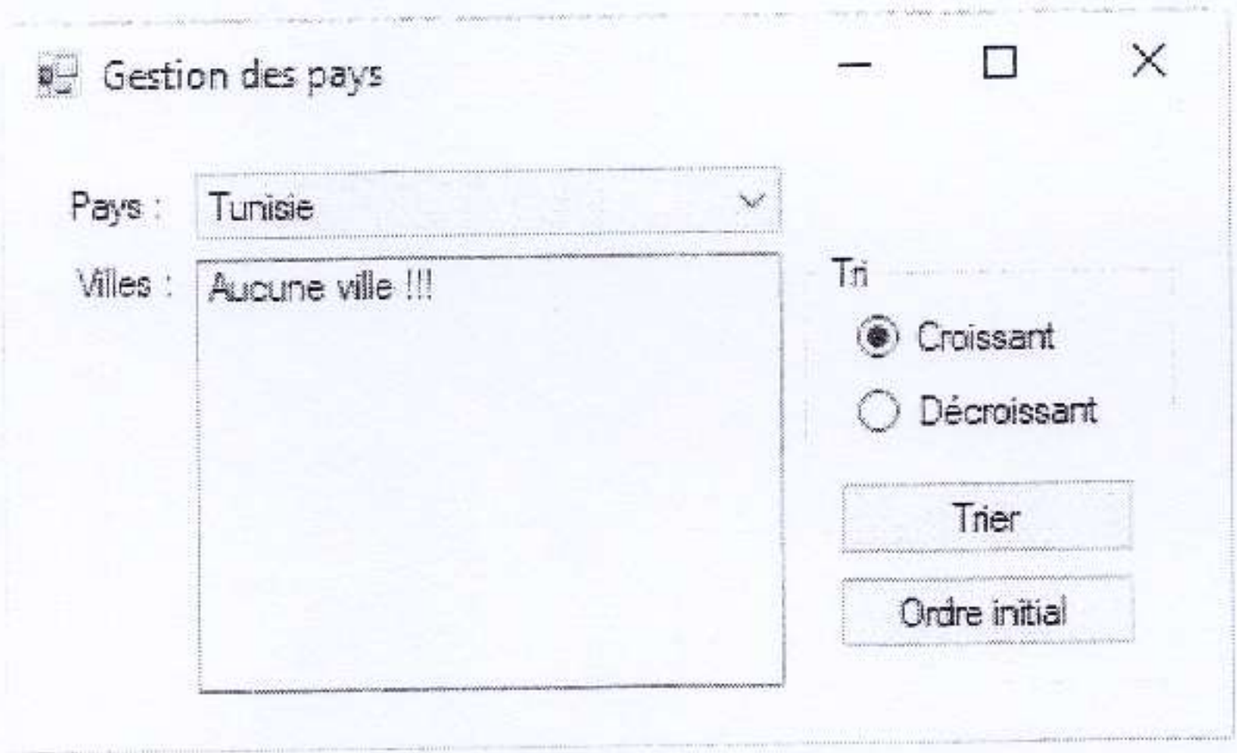
ii. Charger, **à partir de la collection**, le ComboBox par les noms des pays. **3 pts**

c. Ecrire le code permettant d'afficher les villes du pays sélectionné. **4 pts**

i. Remarquez que la ville capitale est affichée différemment des autres villes (commence par le signe - ->): **2 pts**

```
Casablanca (3359000 habitants)
--> Rabat (577000 habitants)
Fès (1112000 habitants)
Tanger (974000 habitants)
Marrakech (928850 habitants)
```

ii. Si le pays ne contient aucune ville, on affiche le message « Aucune ville !!! ». **2 pts**



d. Programmer le bouton **Trier** permettant de trier les villes dans l'ordre choisi. **4 pts**

e. Programmer le bouton **Ordre initial** permettant de rétablir l'ordre initiale des villes. **4 pts**

f. Ecrire le code permettant **la confirmation** de la fermeture du formulaire. **3 pts**

Examen de fin de passage
session Juillet 2018

Filière : Techniques de Développement Informatique

Niveau: TS

Durée : 5 heures

Épreuve : Synthèse

Variante : V2

Barème : / 120pts

❖ Partie I : Théorie (40 pts)

➢ Dossier 1: L'essentiel en technologies de l'information (12 pts)

NB: la calculatrice est strictement interdite.

1. Complétez le tableau suivant : **6 pts**

Binaire	Octal	Décimal	Hexadécimal
111100			
	131		
		131	
			131

2. Soit la table de vérité suivante :

a	b	c	F
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	0

a. Donner la fonction F définie par la table précédente. **2 pts**

b. Simplifier la fonction F **analytiquement**. **2 pts**

c. Simplifier F moyennant le tableau de Karnaugh. **2 pts**

➢ <u>Dossier 2:</u> Analyse et conception orientée objet (16 pts)

Nous souhaitons développer une application permettant de gérer les régions des pays de l'union du Maghreb arabe.

Chaque région a un nom et appartient à un pays (Maroc, Algérie, Tunisie, etc.).

Les régions contiennent des circonscriptions administratives ayant un nom et une population (nombre d'habitants). La région a une circonscription principale (un chef-lieu). Pour simplifier, on suppose que deux circonscriptions différentes n'ont ni le même nom ni la même population.

Les utilisateurs de l'application peuvent passer des tests de connaissance. Chaque test contient des questions du genre : « Quelle est la circonscription principale de … ? ». Les questions sont à choix multiple, c'est-à-dire que pour chaque question nous avons 3 réponses (2 circonscriptions générées automatiquement par l'application en plus de la circonscription principale). On retient pour la question la réponse de l'utilisateur et on enregistre pour le test la date de passation et on lui affecte un numéro automatique.

Les administrateurs de l'application ont pour rôle de gérer les pays, les régions et les circonscriptions. Ils peuvent, aussi, se servir de l'application comme les utilisateurs précédemment décrits.

La gestion des utilisateurs est faite par un responsable (il effectue, bien évidemment, les autres opérations d'administration).

Pour pouvoir utiliser l'application (passer les tests et consulter et gérer les circonscriptions, les régions et les pays) l'utilisateur doit s'authentifier.

Chaque utilisateur a un login et un mot de passe pour se connecter.

1. Elaborer le diagramme de cas d'utilisation. **7.5 pts**
2. Elaborer le diagramme de classes. **8.5 pts**

➢ Dossier 3: Programmation structurée (12 pts)

Comme il est indiqué précédemment, une circonscription a un nom, une population et appartient à une région (chaine de caractères).

Ecrivez un algorithme permettant de :

1. Déclarer la structure **Circonscription**. 1 pt
2. Déclarer un tableau de circonscriptions. 1 pt
3. Demander à l'utilisateur de remplir le tableau par n circonscriptions. 2 pts
   n doit être **supérieur strictement à 0** et **ne peut pas dépasser la taille maximale du tableau.** 1 pt
4. Afficher **la population totale d'une région dont le nom est donné** par l'utilisateur. 3 pts
5. Afficher **la circonscription la moins peuplée.** 4 pts

❖ Partie II: Pratique (80 pts)

➢ Dossier 1: Programmation structurée (16 pts)

1. Soit la fonction suivante :

```
int f(int nombre)
{
    char nombreToString[10];

    sprintf(nombreToString, "%d", nombre);

    return strlen(nombreToString);
}
```

Rappel : La fonction **sprintf** fonctionne comme printf, mais au lieu d'écrire sur la sortie standard (la console), elle écrit dans une chaine de caractères.

Exemple : Le contenu de la variable ch (de type chaine de caractères) après l'appel suivant :

```
sprintf(ch, "%d+%d=%d", 2, 3, 5);
```

est :

2	+	3	=	5	\0

Quelle sera la valeur retournée par chacun des appels suivants :

   a. f(3) 2 pts
   b. f(38) 2 pts
   c. f(387) 2 pts

2. **En utilisant la fonctions sprintf**, écrire la fonction **estUniforme(int nombre)** qui vérifie que nombre est **uniforme**. 4 pts

Un nombre uniforme est un entier naturel **formé par la répétition d'un seul chiffre.**

Exemples :

- 8, 11, 333 et 77777 sont tous des nombres uniformes.
- -55, 15 et 387 ne sont pas des nombres uniformes.

3. Ecrivez un programme permettant de :

   a. Lire un entier e **compris dans l'intervalle [1, 100000[** jusqu'à ce que la réponse convienne. **2 pts**

   b. Afficher, **en utilisant la fonction f**, un message indiquant si e appartient à l'un des intervalles suivants : [1, 10[, [10, 100[, [100, 1000[, [1000, 10000[ou [10000, 100000[. **4 pts**

   Exemples :

   - Si e = 7, on affiche : 7 appartient à l'intervalle [1, 10[.
   - Si e = 259, on affiche : 259 appartient à l'intervalle [100, 1000[.
   - etc.

➢ **Dossier 2: Programmation événementielle et orientée objet (64pts)**

Soit la classe **Circonscription** définie comme suit :

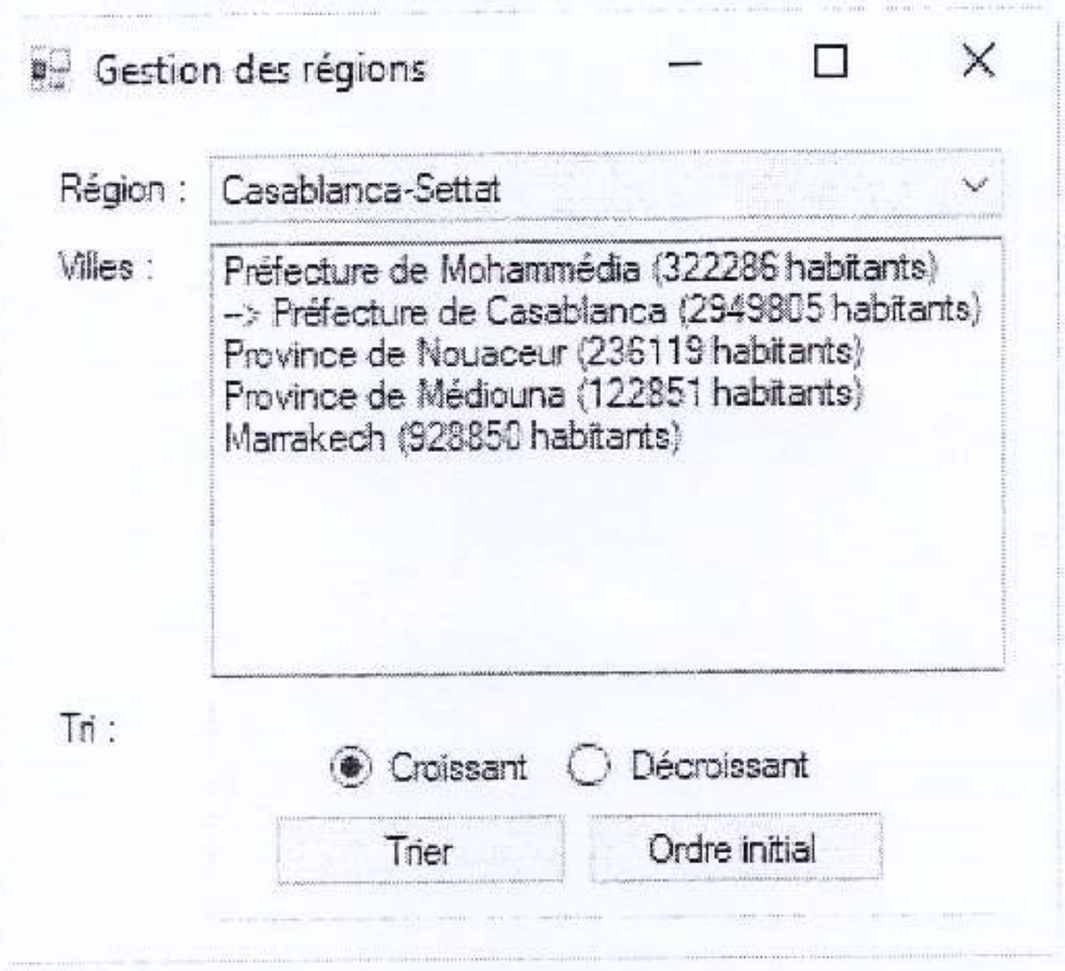
```
class Circonscription
{
    public string Nom;
    public int Population;

    public Circonscription(string nom, int population)
    {
        Nom = nom;
        Population = population;
    }

    public override string ToString()
    {
        return Nom + " (" + Population.ToString() + " habitants)";
    }
}
```

Nous souhaitons créer la classe **Régions** qui se caractérise par son nom, la liste des circonscriptions et la circonscription principale.

1. Définir les attributs de la classe ainsi que leurs propriétés sachant que : **4.5 pts**
   a. Le nom **ne peut pas contenir des chiffres** et doit comporter au **moins 4 caractères**, sinon **on génère une exception**. **3 pts**
   b. La circonscription principale **doit faire partie des circonscriptions**, sinon **elle faut l'ajouter**. **3 pts**
2. Ajouter les constructeurs suivants :
   a. Un constructeur permettant d'initialiser le nom. **1.5 pt**
   b. Un constructeur permettant d'initialiser le nom et la circonscription principale. **2 pts**
3. Ajouter la propriété **NombreCirconscriptions** permettant de retourner le nombre de circonscriptions de la région en cours. **2 pts**
4. Ajouter la méthode **Rechercher(String nomCirconscription)** renvoyant l'indice de la circonscription dont le nom est passé en paramètre s'elle appartient à la région en cours et -1 sinon. **3 pts**
5. Ajouter la méthode **Ajouter(Circonscription circonscription)** permettant d'ajouter la circonscription en paramètre à la liste s'elle n'existe pas auparavant. **3 pts**
6. Ajouter la méthode **Supprimer(String nomCirconscription)** permettant de supprimer la circonscription dont le nom est passé en paramètre. **S'elle s'agit de la circonscription principale, Il faut remettre à null le champ CirconscriptionPrincipal.** **4 pts**
7. Ajouter la méthode **TrierParPopulation(String ordre) retournant la liste des circonscriptions** triée par population dans l'ordre spécifiée en argument (croissant ou décroissant). **5 pts**
8. Ajouter la méthode **PopulationTotale()** retournant le nombre d'habitants **3 pts**

de la région.

9. Soit le formulaire suivant :



Avec :

Contrôle	Type	Libellé
Casablanca-Settat	ComboBox	comboBox_Régions
Préfecture de Mohammédia (322286 habitants) -> Préfecture de Casablanca (2949805 habitants) Province de Nouaceur (236119 habitants) Province de Médiouna (122851 habitants) Marrakech (928850 habitants)	ListBox	listBox_Circonscriptions
Croissant	RadioButton	radioButton_Croissant
Décroissant	RadioButton	radioButton_Décroissant
Trier	Button	button_Trier
Ordre initial	Button	button_OrdreInitial

a. Déclarer une collection des objets **Région**. **1 pt**

b. Ecrire le code s'exécutant au chargement qui permet de : **1 pt**

i. Remplir la collection par les pays suivants : **6 pts**

Région	Circonscriptions		Circonscription principale
	Nom	Population	
Casablanca-Settat	Préfecture de Mohammédia	322286	Préfecture de Casablanca
	Préfecture de Casablanca	2949805	
	Province de Nouaceur	236119	
	Province de Médiouna	122851	

Marrakech-Safi	-	-	-
Laâyoune-Sakia El Hamra	Province de Laâyoune	210023	Province de Laâyoune
	Province de Boujdour	46129	
	Province de Tarfaya	10410	

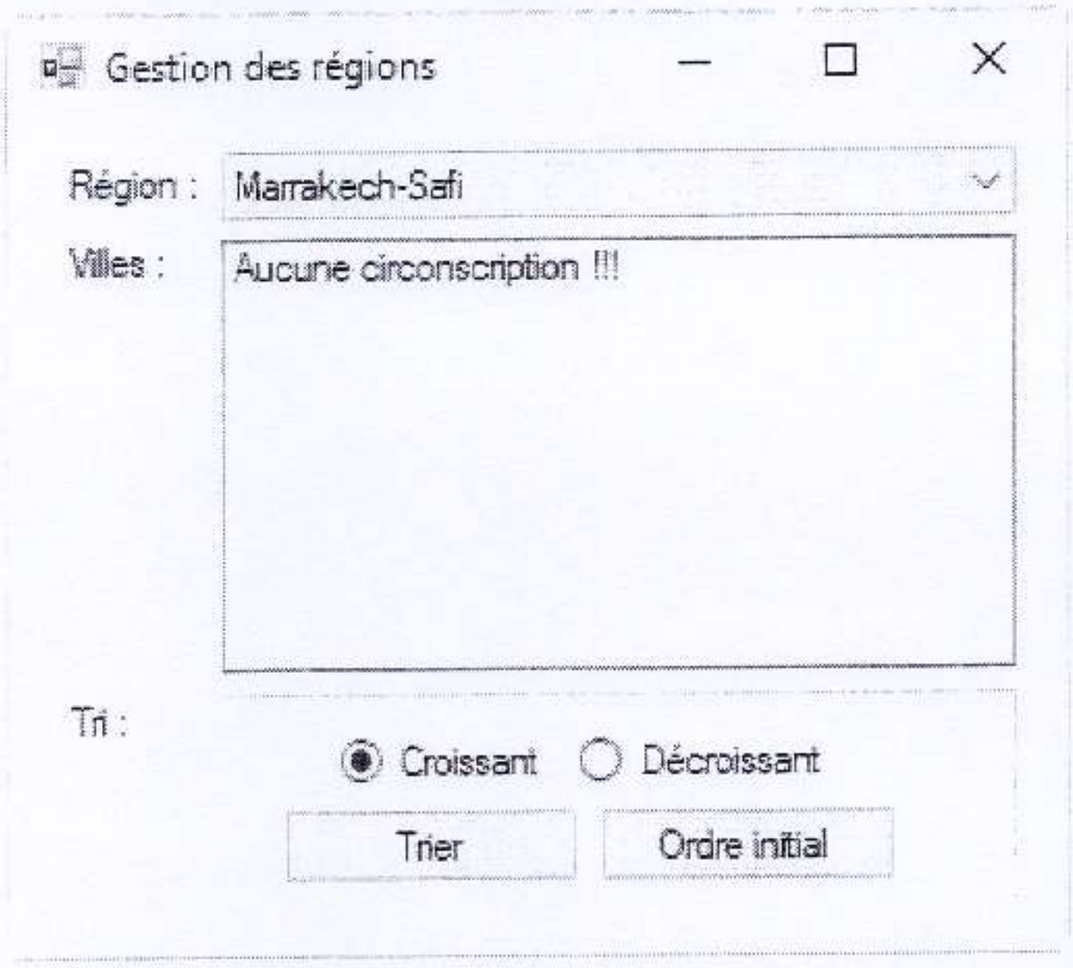
ii. Charger, **à partir de la collection**, le ComboBox par les noms des régions. **3 pts**

c. Ecrire le code permettant d'afficher les circonscriptions de la région sélectionnée. **4 pts**

i. Remarquez que la circonscription principale est affichée différemment des autres circonscriptions : **2 pts**

Préfecture de Mohammédia (322286 habitants)
-> Préfecture de Casablanca (2949805 habitants)
Province de Nouaceur (236119 habitants)
Province de Médiouna (122851 habitants)
Marrakech (928850 habitants)

ii. Si la région ne contient aucune circonscription, on affiche le message « Aucune circonscription !!! ». **2 pts**



d. Programmer le bouton **Trier** permettant de trier les circonscriptions dans l'ordre choisi. **4 pts**

e. Programmer le bouton **Ordre initial** permettant de rétablir l'ordre initiale des circonscriptions. **4 pts**

f. Ecrire le code permettant la **confirmation** de la fermeture du formulaire. **3 pts**