|  |  |  |
| --- | --- | --- |
| C:\Users\Administrateur\Downloads\logo fr .jpg |  | C:\Users\Administrateur\Downloads\logo ar.jpg |
| **Office de la Formation Professionnelle et de la Promotion du Travail** |
| **Direction de la Recherche et de l’Ingénierie de la Formation : Division Examen** |

**Examen National de Fin d’année**

**Session de juin 2019**

**Examen de Passage (Epreuve de Synthèse)**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Eléments de correction** | | | | | | | | |
| **Filière** | ***Techniques de Développement Informatique*** | | | **Variante** | | | | **V2** |
|  |  | | |  | | | |  |
| **Niveau** | ***TS*** | **Durée** | ***4 Heures*** |  |  | **Barème** | ***/100*** | |
|  |  | | |  | | | |  |

**Consignes et Précisions aux correcteurs :**

Chers Mesdames et messieurs les correcteurs, veuillez SVP respecter impérativement les consignes suivantes :

* **Tenir compte de la plate-forme utilisée (.Net, Java, ...)**
* **Plusieurs solutions peuvent être envisagées pour une même question.**

|  |  |
| --- | --- |
| **Partie I : Théorie** | **40 pts** |
|  | |
| **Dossier 1 : L’essentiel en technologies de l’information** | **12 pts** |
|  | |
| 1. **0.5 pt pour chaque conversion** | **6 pts** |
| |  |  |  |  | | --- | --- | --- | --- | | **Binaire** | **Octal** | **Décimal** | **Hexadécimal** | | 10101010 |  |  |  | |  | 63 |  |  | |  |  | 17 |  | |  |  |  | 2CD | | |
|  | |
|  | **1.5 pts** |
| **Ou**  **Ou**  **Ou** | **1.5 pts** |
| 1. Simplification moyennant le tableau de Karnaugh :  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | |  |  |  |  |  | **Ou :** | |  | **1** | **1** | **1** | 0 | |  | **1** | 0 | **1** | 0 | | **3 pts** |

|  |  |
| --- | --- |
| **Dossier 2 : Analyse et conception orientée objet** | **13 pts** |
|  | |
| 1. Diagramme de cas d’utilisation : **0.25 pt pour chaque élément (acteur, Cas d’utilisation et association)** | **5 pts** |
|  | |
| 1. Diagramme de classes : **0.5 pt pour chaque élément (classe et association)** | **8 pts** |
|  | |

|  |  |
| --- | --- |
| **Dossier 3 : Programmation structurée** | **15 pts** |
|  | |
| Algorithme Ex01  Variables choix : Caractère  flag : Booléen  x, y, somme : Entier  Début  Flag <- FAUX  Répéter  Ecrire("1. Donner x et y\n")  Ecrire("2. Afficher la somme des nombres entre x et y\n")  Ecrire("3. Quitter\n")  Ecrire("\nVotre choix\n")  Lire(choix)  Selon (choix) Faire  1 :  flag <- VRAI    Ecrire("Donner x : ")  Lire(y)  Répéter  Ecrire("Donner y : ")  Lire(y)  TantQue(Non(y > x))  2 :  Si(flag = FAUX)  Ecrire("Vous devez donner x et y")  Sinon  somme <- 1  Pour i <- x à y Faire  somme <- somme + i  FinPour  Ecrire("La somme est : ", somme)  FinSi  3 :  Ecrire("Merci")  Sinon :  Ecrire("Choix invalide !!!")  FinSelon  TantQue(choix != '3')  Fin | |

|  |  |
| --- | --- |
| **Partie II : Pratique** | **60 pts** |
|  | |
| **Dossier 1 : Programmation structurée** | **12 pts** |
|  |  |
|  | **3 pts** |
| |  | | --- | | **A screenshot of a cell phone  Description automatically generated** | | |
|  | **2 pts** |
| |  | | --- | |  | | |
|  | **7 pts** |
|  | |

|  |  |
| --- | --- |
| **Dossier 2 : Programmation événementielle et orientée objet** | **48 pts** |
|  | |
|  | |
| [Serializable]  class Etudiant  {  private string \_CodeInscription;  public string CodeInscription  {  get { return \_CodeInscription; }  set  {  Match match = Regex.Match(value, @"^\d{12}$");  if (!match.Success)  throw new Exception("Code d'inscription invalide !!!");  \_CodeInscription = value;  }  }  private string \_NomComplet;  public string NomComplet  {  get { return \_NomComplet; }  set { \_NomComplet = value; }  }  public Etudiant()  {  }  public Etudiant(string codeInscription, string nomComplet)  {  CodeInscription = codeInscription;  NomComplet = nomComplet;  }  public override bool Equals(object obj)  {  Etudiant étudiant = obj as Etudiant;  if (étudiant == null)  return false;  return CodeInscription.Equals(étudiant.CodeInscription);  }  public override string ToString()  {  return string.Format("Etudiant {0} : {1}", CodeInscription, NomComplet);  }  } | |
|  | |
|  | |
|  | |
| class Atelier  {  private List<Etudiant> \_Adhérents = new List<Etudiant>();  public List<Etudiant> Adhérents  {  get { return \_Adhérents; }  set { \_Adhérents = value; }  }  public Atelier()  {  }  public int NombreAdhérents  {  get { return Adhérents.Count; }  }  public int IndiceDe(Etudiant adhérent)  {  foreach (Etudiant item in Adhérents)  {  if (item.Equals(adhérent)) return Adhérents.IndexOf(adhérent);  }  return -1;  }  public bool Ajouter(Etudiant adhérent)  {  if (IndiceDe(adhérent) != -1)  return false;  else  {  Adhérents.Add(adhérent);  return true;  }  }  public void Supprimer(string codeInscription)  {  foreach (Etudiant adhérent in Adhérents)  {  if (adhérent.CodeInscription == codeInscription)  {  Adhérents.Remove(adhérent);  return;  }  }  throw new Exception("L'étudiant à supprimer n'est pas adhérent !!!");  }  public void Supprimer(Etudiant adhérent)  {  Supprimer(adhérent.CodeInscription);  }  public void Enregistrer()  {  string chemin = "Atelier de lecture.bin";  FileStream fileStream;  BinaryFormatter binaryFormatter = new BinaryFormatter();  fileStream = new FileStream(chemin, FileMode.OpenOrCreate, FileAccess.Write);  binaryFormatter.Serialize(fileStream, Adhérents);  fileStream.Close();  }  public void Charger()  {  string chemin = "Atelier de lecture.bin";  FileStream fileStream;  BinaryFormatter binaryFormatter = new BinaryFormatter();  if (File.Exists(chemin))  {  try  {  fileStream = new FileStream(chemin, FileMode.Open, FileAccess.Read);  try  {  Adhérents =  binaryFormatter.Deserialize(fileStream) as List<Etudiant>;  }  catch (Exception)  {  throw new Exception("Problème de lecture !!!");  }  fileStream.Close();  }  catch (Exception)  {  throw new Exception("Problème d'ouverture !!!");  }  }  }  public override string ToString()  {  string bilan = "Atelier de lecture :\n";  bilan += string.Format("- Adhérents ({0}) :\n", NombreAdhérents);  foreach (Etudiant adhérent in Adhérents)  {  bilan += string.Format("\t\* {0}\n", adhérent);  }  return bilan;  }  } | |
|  | |
|  | |
|  | |
| public partial class Gestion\_des\_adhérents : Form  {  Atelier \_Atelier = new Atelier();  public Gestion\_des\_adhérents()  {  InitializeComponent();  }  private void Gestion\_des\_adhérents\_Load(object sender, EventArgs e)  {  try  {  \_Atelier.Charger();  dataGridView\_Adhérents.DataSource = \_Atelier.Adhérents;  }  catch (Exception)  {  }  FormClosing += new FormClosingEventHandler(Gestion\_des\_Adhérents\_FormClosing);  }  void Gestion\_des\_Adhérents\_FormClosing(object sender, FormClosingEventArgs e)  {  DialogResult résultat =  MessageBox.Show("Voulez-vous vraiment fermer l'application ?",  "Attention",  MessageBoxButtons.YesNo,  MessageBoxIcon.Question);  if (résultat == DialogResult.Yes)  \_Atelier.Enregistrer();  else  e.Cancel = true;  }  private void button\_Supprimer\_Click(object sender, EventArgs e)  {  string codeInscription = textBox\_CodeInscription.Text;  try  {  \_Atelier.Supprimer(codeInscription);  dataGridView\_Adhérents.DataSource = null;  dataGridView\_Adhérents.DataSource = \_Atelier.Adhérents;  }  catch (Exception exception)  {  MessageBox.Show(exception.Message,  "Attention",  MessageBoxButtons.OK,  MessageBoxIcon.Information);  }  }  private void button\_Ajouter\_Click(object sender, EventArgs e)  {  Etudiant etudiant = new Etudiant(textBox\_CodeInscription.Text,  textBox\_NomComplet.Text);  if (\_Atelier.Ajouter(etudiant))  {  dataGridView\_Adhérents.DataSource = null;  dataGridView\_Adhérents.DataSource = \_Atelier.Adhérents;  }  else  MessageBox.Show("Cet adhérent existe déjà !!!",  "Attention",  MessageBoxButtons.OK, MessageBoxIcon.Information);  }  } | |