|  |  |  |
| --- | --- | --- |
| 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** | | | | **V1** |
|  |  | | |  | | | |  |
| **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** | | 11011011 | **333** | **219** | **DB** | |  | 71 |  |  | | **1101** | **15** | 13 | **D** | |  |  |  | 1AB | | |
|  | |
|  | **1.5 pts** |
| **Ou**  **Ou**  **Ou** | **1.5 pts** |
| 1. Simplification moyennant le tableau de Karnaugh :  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | |  |  |  |  |  | **Ou :** | |  | **1** | 0 | **1** | 0 | |  | **1** | **1** | **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** |
| **A screenshot of a cell phone  Description automatically generated** | |

|  |  |
| --- | --- |
| **Dossier 3 : Programmation structurée** | **15 pts** |
|  | |
| Algorithme Ex01  Variables choix : Caractère  flag : Booléen  a, b, produit : Entier  Début  Flag <- FAUX  Répéter  Ecrire("1. Entrer a et b\n")  Ecrire("2. Afficher le produit des nombres entre a et b\n")  Ecrire("3. Quitter\n")  Ecrire("\nVotre choix\n")  Lire(choix)  Selon (choix) Faire  1 :  flag <- VRAI    Ecrire("Donner a : ")  Lire(a)  Répéter  Ecrire("Donner b : ")  Lire(b)  TantQue(Non(b > a))  2 :  Si(flag = FAUX)  Ecrire("Vous devez donner a et b")  Sinon  produit <- 1  Pour i <- a à b Faire  produit <- produit \* i  FinPour  Ecrire("Le produit est : ", produit)  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** |
| |  | | --- | | **A screenshot of a cell phone  Description automatically generated** | | |
|  | **7 pts** |
|  | |

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