Solutions chapitre 4
Créer une classe simple

Exercice 1

Voici un code possible pour la classe `Article`.

```
class Article
{
    //- Variables d'instances ----------------------------------------
        private string  description;
        private decimal prix;
        private decimal txTva;
        private decimal txRemise;

    //- Constructeurs -------------------------------------------------
    public Article(string description, decimal prix, decimal txTva)
    {
        //- Validations locales ---------------------------------------
        if (   Regex.Match(description, "^[0-9A-Z]{1}").Success == false
            || description.Length < 10
            || description.Length > 100)
            throw new Exception("Description de structure incorrecte.");
        if (txTva < 0.1m || txTva > 0.3m)
            throw new Exception(
                    "Le taux de TVA doit être compris entre 10% et 30%.");

        //- Initialisations -------------------------------------------
        this.description = description;
        this.SetPrix(prix);
        this.txTva       = txTva;
        this.txRemise    = 0;
    }

    public Article(string description, decimal prix):
                                    this(description, prix, 0.205m)
    {
    }

    //- Méthodes ------------------------------------------------------
    public void Solder(decimal txRemise)
    {
        //- Validation ------------------------------------------------
        if (txRemise < 0.1m || txRemise > 0.9m)
            throw new Exception(
                "Le taux de remise doit être compris entre 10% et 90%.");
        //- Initialisation --------------------------------------------
        this.txRemise = txRemise;
    }

    public void Solder()
    {
        this.Solder(0.1m);
    }

    public void DéSolder()
    {
        this.txRemise = 0m;
        //ATTENTION : this.Solder(0m) => exception
    }
```

```
    public decimal GetPrixVente()
    {
        return this.prix * (1 + this.txTva) * (1 - this.txRemise);
    }

    public decimal GetPrix()
    {
        return this.prix;
    }

    public void SetPrix(decimal prix)
    {
        if (prix < 0.1m || prix > 2000m)
             throw new Exception(
               "Le prix doit être compris entre 0.1 Euro et 2000 Euros.");
        this.prix = prix;
    }

    public string GetChaîne()
    {
        return this.description +
               " - Prix : " + this.GetPrixVente() + " Euros" +
               ((this.txRemise > 0)?    (" (soldé à " +
                                          (int)((1 - this.txRemise) * 100) +
                                          "%)")
                                          :"") + ".";
    }
}
```

Un code de test possible est le suivant.

```
static void Main(string[] args)
{
    //- Déclarations ----------------------------------------------
    Article article1 = null;

   //- Constructions ----------------------------------------------
    try
    {
        Console.WriteLine("Test des constructeurs");
        //= Enlever commentaire d'une des lignes ====================

        //- Corrects ---------------------------------------------
        article1 = new Article("3 tapettes à mouches", 100m, 0.2m);
        article1 = new Article("Tapettes à mouches"  , 100m, 0.3m);
        article1 = new Article("3 tapettes à mouches", 120m);

        //- Incorrects -------------------------------------------
        //article1 = new Article("tapettes à mouches", 100m);
        //article1 = new Article("Tap", 100m);
        //article1 = new Article(
             "Tapettes à mouches électronique à capteur de " +
              "pression différentiel avec viseur laser intégré " +
             "et périscope de quatrième génération ", 100m);
        //article1 = new Article("3 tapettes à mouches", 0.099m);
        //article1 = new Article("3 tapettes à mouches", 2001m);
        //article1 = new Article("3 tapettes à mouches", 100m, 0.4m);
        //article1 = new Article("3 tapettes à mouches", 100m,
        //                                                0.02m);

       //= Fin enlever commentaire d'une ligne ====================
       if (article1 != null)
       {
            Console.WriteLine(article1.GetChaîne());
            Console.WriteLine("Test de mise en solde.");
            //= Enlever commentaire d'une des lignes ===============

            //- Corrects ---------------------------------------
```

```
            article1.Solder(0.1m);

            //- Incorrects ----------------------------------------
            //article1.Solder(.);
            //article1.Solder(100);
            //= Fin enlever commentaire d'une ligne ================
            Console.WriteLine(article1.GetChaîne());
            article1.DéSolder();
            Console.WriteLine(article1.GetChaîne());
        }
        else
            Console.WriteLine("Aucun objet n'a été construit.");

        Console.WriteLine("Aucune erreur détectée");
    }
    catch (Exception ex)
    {
        Console.WriteLine("Erreur : " + ex.Message);
    }
    Console.ReadLine();
}
```

Exercice 2

Voici un exemple de code possible pour la classe `Dé`.

```
class Dé
{
    /*- Variables d'instance ----------------------------------------*/
    private int    valeur;
    private Random générateur;

    /*- Constructeurs et méthodes -----------------------------------*/
    public Dé()
    {
        this.ConstruireGénérateur();
        this.ChangerAléatoirement();
    }

    public Dé(int valeur)
    {
        if (valeur < 1 || valeur > 6)
            throw new Exception(
                        "La valeur doit être un comprise entre 1 et 6.");
        this.ConstruireGénérateur();
        this.valeur = valeur;
    }

    public void Lancer()
    {
        this.ChangerAléatoirement();
    }

    public int GetValeur()
    {
        return this.valeur;
    }

    private void ChangerAléatoirement() //Méthode factorisation
    {
       this.valeur = this.générateur.Next() % 6 + 1;
       //ou bien this.valeur = this.générateur.Next(1, 7);
    }

    private void ConstruireGénérateur() //Méthode de factorisation
    {
```

```
        this.générateur = new Random(DateTime.Now.Millisecond);
        Thread.Sleep(TimeSpan.FromSeconds(0.01));

    }
}
```

Voici un exemple de code possible pour la méthode principale.

```
static void Main(string[] args)
{
    int                 nbDés = 0,
                        somme;
    ArrayList           lesDés       = new ArrayList();

    //- Tableau de compteurs. statistiques[n-1] sera un compteur
    //  du nombre de fois que la somme n a été obtenue.

    int[]               statistiques;

    Console.Write("Nombre de dés entre 2 et 10 : ");
    while(   Int32.TryParse(Console.ReadLine(), out nbDés) == false
          || nbDés < 2
          || nbDés > 10)
        Console.Write("Un ENTIER ENTRE 2 ET 10, pignouf : ");

    statistiques = new int[nbDés * 6];

    for (int cptrDé = 0; cptrDé < nbDés; cptrDé++)
        lesDés.Add(new Dé());


    //- Un tour par simulation ----------------------------------
    for (int cptSimu = 0; cptSimu < 10000000; cptSimu++)
    {
         //- Jet des dés et calcul de la somme obtenue -----------
        somme = 0;
        foreach (Dé dé in lesDés)
        {
           dé.Lancer();
           somme += dé.GetValeur();
        }
        //- On augmente le compteur qui correspond à cette somme -
        statistiques[somme-1]++;
    }


    for (somme = 0; somme < statistiques.Length; somme++)
        if(statistiques[somme] > 0)
            Console.WriteLine("Total de " + (somme + 1) +
                              " observé avec une probabilité de " +
                              statistiques[somme] / 100000.0 + " %.");

     Console.ReadLine();

}
```

Exercice 3

(a) Voici un exemple de code possible pour la classe `CompteAVue`.

```
class CompteAVue
{
    //- Variables d'instance ------------------------------------------
    private string   nomTitulaire;
    private string   numéro;
    private decimal  solde;
```

```
    private decimal  ligneCrédit;    // Valeur 1000 signifie ligne à -1000
    private decimal  débitJournalierMax;
    private decimal  cumulDébitsJournéeCourante;
    private DateTime dateDernierDébit;

    //- Constructeurs --------------------------------------------------------
    public CompteAVue(string numero, string nomTitulaire) :
                    this(numero, nomTitulaire, 0m, 0m, 1000m)
    {
    }

    public CompteAVue(string numero, string nomTitulaire,
                      decimal solde, decimal ligneCrédit,
                      decimal débitJournalierMax)
    {
        this.SetNuméro(numero);
        this.SetNomTitulaire(nomTitulaire);
        this.SetSoldeEtLigne(solde, ligneCrédit);
        this.débitJournalierMax = débitJournalierMax;
        this.cumulDébitsJournéeCourante = 0;
        this.dateDernierDébit = DateTime.Today;
    }

    //- Méthode publiques ----------------------------------------------------
    public bool Créditer(decimal montant)
    {
        bool retVal = false;

        if (montant >= 0)
        {
            this.solde += montant;
            retVal = true;
        }
        return retVal;
    }

    public bool Débiter(decimal montant)
    {
        bool retVal = false;

        if(this.dateDernierDébit < DateTime.Today)
        {
            this.dateDernierDébit = DateTime.Today;
            this.cumulDébitsJournéeCourante = 0m;
        }

        if (   montant > 0
            && this.cumulDébitsJournéeCourante + montant
                                      <= this.débitJournalierMax
            && this.solde - montant >= -this.ligneCrédit)
        {
            this.solde                       -= montant;
            this.cumulDébitsJournéeCourante += montant;
            retVal = true;
        }

        return retVal;
    }


    public decimal GetSolde()
    {
        return this.solde;
    }

    public decimal GetLigneRestanteJournée()
    {
        return this.débitJournalierMax - this.cumulDébitsJournéeCourante;
```

```
    }

    public string GetChaîne()
    {
        return "Compte                  : " +
               this.numéro + "\n" +
               "Titulaire               : " +
               this.nomTitulaire + "\n" +
               "Solde                   : " +
               this.solde + " Euros\n" +
               "Ligne de crédit         : " +
               this.ligneCrédit + " Euros\n" +
               "Maximum journalier      : " +
               this.débitJournalierMax + " Euros\n" +
               "Ligne restante          : " +
               this.GetLigneRestanteJournée() + " Euros\n" +
               "Journée courante        : " +
               this.dateDernierDébit.ToString("dd-MM-yy") + "\n" +
               "Débits journée courante : " +
               this.cumulDébitsJournéeCourante + " Euros.";
    }

    //- Méthode privée ----------------------------------------------------
    private void SetNuméro(string numéro)
    {
        if (numéro.Length != 16
            || Regex.Match(numéro, "^BE[0-9]{14}$").Success == false)
            throw new Exception(
                      "Un compte doit débuter pas BE suivi de 14 chiffres.");

        if (Convert.ToInt64(numéro.Substring(4, 10)) % 97 !=
            Convert.ToInt64(numéro.Substring(14, 2)))
            throw new Exception("Erreur de check digit.");

        this.numéro = numéro;
    }

    private void SetNomTitulaire(string nomTitulaire)
    {
        for (int cpt = 0; cpt < nomTitulaire.Length; cpt++)
            if(Char.IsDigit(nomTitulaire[cpt]))
                throw new Exception(
                  "Le nom du titulaire contient un (des) chiffre(s).");
        this.nomTitulaire = nomTitulaire;
    }

    private void SetSoldeEtLigne(decimal solde, decimal ligneCrédit)
    {
        if (solde < -ligneCrédit)
           throw new Exception(
                      "La ligne de crédit est dépassée.");
        this.solde = solde;
        this.ligneCrédit = ligneCrédit;
    }
}
```

(b) Voici un exemple de programme de test.

```
static void Main(string[] args)
{
     CompteAVue compte = null;
     string choix;

     try
     {
         //= Décommenter une seule instruction ========================
         compte = new CompteAVue("BE84063178154679", "Jule César");
         //compte = new CompteAVue("BE84063178154679", "Jule César",
```

```
            //                          -100, 10000, 1000);
            //compte = new CompteAVue("BE84063178154678", "Alexandre le grand");
            //compte = new CompteAVue("BE84 0631 7815 4678",
            //                                    "Alexandre le grand");
            //compte = new CompteAVue("DE84063178154678", "Alexandre le grand");
            //compte = new CompteAVue("BE840BE178154678", "Alexandre le grand");
            //compte = new CompteAVue("BE84063178154679", "3 Suisses");
            //compte = new CompteAVue("BE84063178154679", "Jule César", -10001,
            //                                                  10000, 1000);
            Console.WriteLine(compte.GetChaîne());

            do
            {
                Console.WriteLine("\n<C>réditer");
                Console.WriteLine("<D>ébiter");
                Console.WriteLine("<Q>uitter");

                do
                {
                    Console.Write("\nVotre choix : ");
                    choix = Console.ReadLine().ToUpper().Trim();
                }
                while (   choix.Length != 1
                       || "CDQ".Contains(choix.ToUpper()) == false);

                switch (choix.ToUpper())
                {
                    case "C":
                        Console.Write("Montant à créditer : ");
                        if (compte.Créditer(
                                Convert.ToDecimal(Console.ReadLine())) == false)
                        {
                            Console.ForegroundColor = ConsoleColor.Red;
                            Console.WriteLine("REFUS.");
                            Console.ForegroundColor = ConsoleColor.White;
                        }
                        break;
                    case "D":
                        Console.Write("Montant : ");
                        if (compte.Débiter(
                                Convert.ToDecimal(Console.ReadLine())) == false)
                        {
                            Console.ForegroundColor = ConsoleColor.Red;
                            Console.WriteLine("REFUS.");
                            Console.ForegroundColor = ConsoleColor.White;
                        }
                        break;
                }
                Console.WriteLine("\n" + compte.GetChaîne());
            }
            while (choix != "Q");
        }
        catch (Exception ex)
        {
            Console.ForegroundColor = ConsoleColor.Red;
            Console.WriteLine("ERREUR : " + ex.Message);
            Console.ForegroundColor = ConsoleColor.White;
        }
}
```

(c) Evident.

Exercice 4

(a) Voici un exemple de code possible pour la classe `Chrono`. Elle utilise une énumération `EtatChrono`, mais n'importe quel type énuméré (`int`, `char`, `string`, etc.) pourrait aussi être utilisé.

```
enum EtatChrono { Initialisé, Démarré, Arrêté };

class Chrono
{
    //- Variables d'instance ----------------------------------------------
    private DateTime dernierDémarrage;  //Heure du dernier (re)démarrage
    //(si démarré).
    private TimeSpan accuMS;            //Mémorise le temps total de
    //déjà mesuré, en millièmes de
    //secondes, lors du dernier arrêt.
    private EtatChrono état;   //Etat actuel du chronomètre

    //- Constructeur ------------------------------------------------------
    public Chrono()
    {
        this.accuMS = new TimeSpan(0);
        this.état = EtatChrono.Initialisé;
        this.dernierDémarrage = DateTime.Now;
        //Pour éviter tout problème lors d'une invocation
        //prématurée à getSeconds
    }

    //- Méthodes publiques ------------------------------------------------
    public void Initialiser()
    {
        if (this.état != EtatChrono.Démarré)
        {
            this.accuMS = new TimeSpan(0);
            this.état = EtatChrono.Initialisé;
        }
    }

    public void Démarrer()
    {
        if (this.état != EtatChrono.Démarré)
        {
            this.état = EtatChrono.Démarré;
            this.dernierDémarrage = DateTime.Now;
        }
    }

    public void Arrêter()
    {
        if (this.état == EtatChrono.Démarré)
        {
            this.état = EtatChrono.Arrêté;
            this.accuMS += this.GetEcartDepuisDernierDémarrage();
        }
    }

    private TimeSpan GetEcartDepuisDernierDémarrage()
    {
        return (DateTime.Now - this.dernierDémarrage);
    }

    public string EnChaîne()
    {
        TimeSpan total = this.accuMS;
        total += this.GetEcartDepuisDernierDémarrage();

        return String.Format("{0:00}:{1:00}:{2:00}:{3:00}",
                             total.Hours, total.Minutes,
                             total.Seconds, total.Milliseconds / 10);
    }
}
```

(b) Voici un exemple de code de test.

```
static void Main(string[] args)
{
    Chrono chrono = new Chrono();
    string choix;

    do
    {
        Console.WriteLine();
        Console.WriteLine("<1> Afficher");
        Console.WriteLine("<2> Démarrer");
        Console.WriteLine("<3> Arrêter");
        Console.WriteLine("<4> Réinitialiser");
        Console.WriteLine("<5> Quitter");

        do
        {
            Console.Write("\nVotre choix : ");
            choix = Console.ReadLine().ToUpper().Trim();
        }
        while (Regex.Match(choix, "^[1-5]$").Success == false);

        switch (choix.ToUpper())
        {
            case "1":
                Console.WriteLine(chrono.EnChaîne());
                break;
            case "2":
                chrono.Démarrer();
                break;
            case "3":
                chrono.Arrêter();
                break;
            case "4":
                chrono.Initialiser();
                break;
        }
    }
    while (choix != "5");
}
```

Exercice 5

```
class Tondeuse
{
    //- Variables d'instances ------------------------------------------
    private DateTime dteFabrication;
    private double   batMax;
    private double   batRési;
    private double   consoParM;

    //- Constructeurs et méthodes --------------------------------------
    public Tondeuse(double batMax, double consoParM)
    {
        if (batMax <= 0)
                throw new Exception(
        "La charge maximale de la batterie doit être strictement positive.");
        if (consoParM <= 0)
            throw new Exception("
         La consommation par mètre doit être strictement positive.");
        if (this.batMax / this.consoParM < 100)
            throw new Exception("Impossible de tondre 100 mètres.");

        this.dteFabrication = DateTime.Today;
        this.batMax = batMax;
        this.batRési = 0;
        this.consoParM = consoParM;
```

```
    }

    public Tondeuse(double consoParM):this(1000, consoParM)
    {
    }

    public bool HorsService()
    {
        return (this.batRésiduelle / this.consoParM < 5);
    }

    public int Tondre(int demandé)
    {
        int retval = 0;

        while(!this.HorsService() && demandé > 0)
        {
            this.batRésiduelle -= this.consoParM;
            demandé--;
            retval++;
        }

        return retval;
    }

    public void Recharger()
    {
        this.batRésiduelle = this.batMax;
    }

    public string GetChaîne()
    {
        return     this.dteFabrication.ToString("yyyy-MM-dd, ")
               +   ((this.HorsService())?"HORS SERVICE":"EN SERVICE");
    }
}
```

Exercice 6

(a) Voici un exemple de code possible pour les classes `RDV` et `Agenda`.

```
class RDV
{
    //- Variables d'instances ----------------------------------------------
    private DateTime dateDébut;
    private TimeSpan durée;
    private String   objet;

    //- Constructeurs et méthodes ------------------------------------------


    public RDV(string objet, string date,
               string heureDébut, int duréeMinutes)
    {
        try
        {
            if (duréeMinutes <= 0)
                throw new Exception("Durée négative ou nulle interdite.");

            this.dateDébut = Convert.ToDateTime(date + " " + heureDébut);
            this.durée     = new TimeSpan(0, duréeMinutes, 0);

            if ((this.dateDébut + this.durée).ToString("dd/MM/yyyy")
                        != this.dateDébut.ToString("dd/MM/yyyy"))
                throw new Exception(
              "Un rendez-vous ne peut pas chevaucher plusieurs journées.");
```

```
            this.objet = objet;
        }
        catch
        {
            throw new Exception("Paramètres incorrects.");
        }
    }

    public DateTime GetDateDébut()
    {
        return this.dateDébut;
    }

    public DateTime GetDateFin()
    {
        return this.dateDébut + this.durée;
    }

    public TimeSpan GetDurée()
    {
        return this.durée;
    }

    public bool Recouvre(RDV autreRdv)
    {
        return !(   this.dateDébut >= autreRdv.GetDateFin()
                 || this.dateDébut + this.durée
                                   <= autreRdv.GetDateDébut());
    }

    public String DateEnChaîne()
    {
        return this.dateDébut.ToString("dd/MM/yyyy");
    }

    public String EnChaîne()
    {
        return this.dateDébut.ToString("HH:mm")
               + " - "
               + (this.dateDébut + this.durée).ToString("HH:mm")
               + " : "
               + this.objet;
    }
}
```

```
class Agenda
{
    //- Variable d'instance ------------------------------------
    private ArrayList lesRDV;

    //- Méthodes -----------------------------------------------
    public Agenda()
    {
        this.lesRDV = new ArrayList();
    }

    public bool AjouterRDV(string objet, string date,
                           string heureDébut, int duréeMinutes)
    {
        RDV  nouveau = null;
        bool retVal = true;
        int  position;

        try
        {
            nouveau = new RDV(objet, date, heureDébut, duréeMinutes);

            foreach (RDV r in this.lesRDV)
```

```
                if (r.Recouvre(nouveau))
                {
                    retVal = false;
                    break;
                }

            if (retVal == true)
            {
                position = 0;
                while (    position <  this.lesRDV.Count
                       &&  ((RDV)this.lesRDV[position]).GetDateDébut()
                                                < nouveau.GetDateDébut())
                    position++;
                this.lesRDV.Insert(position, nouveau);
                retVal = true;
            }
        }
        catch
        {
            retVal = false;
        }
        return retVal;
    }

    public bool AjouterRDV(string objet, string heureDébut, int duréeMinutes)
    {
        return this.AjouterRDV(objet, DateTime.Now.ToString("dd/mm/yyyy"),
                               heureDébut, duréeMinutes);
    }

    public bool AjouterRDV(string objet, int duréeMinutes)
    {
        return this.AjouterRDV(objet,
                          DateTime.Now.ToString("hh:mm"), duréeMinutes);
    }

    public bool SupprimerRDV(string date, string heureDébut)
    {
        bool retVal = false;

        try
        {
             DateTime demandé = Convert.ToDateTime(date + " " + heureDébut);
             foreach(RDV rdv in this.lesRDV)
                if(rdv.GetDateDébut() == demandé)
                {
                    this.lesRDV.Remove(rdv);
                    retVal = true;
                    break;
                }
        }
        catch
        { }

        return retVal;
    }

    public String EnChaîne(string date)
    {
        String retVal = "Planning du ";
        try
        {
            DateTime trav = Convert.ToDateTime(date);
            retVal += trav.ToString("dd/MM/yyyy") + "\n\n";
            foreach (RDV r in this.lesRDV)
                if (r.DateEnChaîne() == trav.ToString("dd/MM/yyyy"))
                    retVal = retVal + r.EnChaîne() + "\n";
        }
```

```
        catch
        {
            retVal = "";
        }
        return retVal;
    }
}
```

Voici un programme de test possible.

```
static void Main(string[] args)
{
    Agenda   agenda = new Agenda();
    string   objet, date, heureDébut, choix;
    int      duréeMinutes;

    do
    {
        Console.WriteLine("\n----- MENU PRINCIPAL ------");
        Console.WriteLine("<1> Ajouter un rendez-vous.");
        Console.WriteLine("<2> Supprimer un rendez-vous.");
        Console.WriteLine("<3> Afficher les rendez-vous d'une journée.");
        Console.WriteLine("<4> Quitter.");

        do
        {
             Console.Write("\nVotre choix : ");
             choix = Console.ReadLine().Trim();
        }
        while (Regex.Match(choix, "^[1-4]$").Success == false);

        try
        {
            switch (choix)
            {
                case "1":
                    Console.WriteLine("Encodage d'un nouveau rendez-vous.");

                    Console.Write    ("   son objet             : ");
                    objet = Console.ReadLine();

                    Console.Write    ("   sa date (dd/mm/yyyy)  : ");
                    date = Console.ReadLine();

                    Console.Write    ("   son heure (hh:mm)     : ");
                    heureDébut = Console.ReadLine();

                    Console.Write    ("   sa durée              : ");
                    duréeMinutes = Convert.ToInt32(Console.ReadLine());

                   Console.WriteLine(!agenda.AjouterRDV(objet,
                                                        date,
                                                        heureDébut,
                                                        duréeMinutes)?
                                     "Echec d'ajout.":
                                     "Rendez-vous ajouté.");
                    break;

                case "2":
                    Console.WriteLine("Suppression d'un rendez-vous.");

                    Console.Write    ("   sa date (dd/mm/yyyy)  : ");
                    date = Console.ReadLine();

                    Console.Write    ("   son heure (hh:mm)     : ");
                    HeureDébut = Console.ReadLine();

                    Console.WriteLine(!agenda.SupprimerRDV(date,
```

```
                                                                heureDébut)?
                                        "Echec de suppression." :
                                        "Rendez-vous supprimé.");
                        break;

                    case "3":
                        Console.Write(
                             "Choisir la journée à afficher (dd/mm/yyyy) : ");
                        date = Console.ReadLine();
                        Console.WriteLine(agenda.EnChaîne(date));
                        break;
                }
            }
            catch (Exception ex)
            {
                Console.WriteLine("ERREUR : " + ex.Message);
            }
        }
        while (choix != "4");
}
```