Polymorphisme

TP 9

R2.01

Malo Mouron B1

Table des matières

Codez cette classe et fournir son code dans le compte-rendu

```
namespace TP._09;

/// <summary>
/// Represents an article with properties such as description, unit price, and brand.
/// </summary>
public abstract class Article
{
    #region attributs

    // référence de l'article
    private string _reference;

    // Description de l'article
    private string _description;

    // Prix de l'article
    private float _unitPrice;

    // Marque de l'article
    private string _brand;

    #endregion

    #region accesseurs

    /// <summary>
    /// Gets the description of the article.
    /// </summary>
```

```
    public abstract string Description { get; }

    /// <summary>
    /// Gets the unit price of the article.
    /// </summary>
    public abstract float UnitPrice { get; }

    /// <summary>
    /// Gets the brand of the article.
    /// </summary>
    public abstract string Brand { get; }

    #endregion

    #region constructeurs

    /// <summary>
    /// Initializes a new instance of the <see cref="Article"/> class.
    /// </summary>
    /// <param name="reference">The reference of the article.</param>
    /// <param name="description">The description of the article.</param>
    /// <param name="unitPrice">The unit price of the article.</param>
    /// <param name="brand">The brand of the article.</param>
    public Article(string reference, string description, float unitPrice, string brand)
    {
        _reference = reference;
        _description = description;
        _unitPrice = unitPrice;
        _brand = brand;
    }

    #endregion

    #region méthodes

    /// <summary>
    /// Returns a string that represents the current article.
    /// </summary>
    /// <returns>A string that represents the current article.</returns>
    public override string ToString() => $"Référence : {_reference}, Description : {_description}, Prix
unitaire : {_unitPrice}, Marque : {_brand}";

    #endregion
}
```

Pouvez-vous tester cette classe avec un programme (fonction Main) ? Si non, pourquoi ? Si oui, fournir le programme.

Non on ne peut pas tester cette class car c'est une class abstraite et il faut forcément la faire hériter d'une autre class.

Codez cette classe et fournir son code dans le compte-rendu.

```
namespace TP._09;

/// <summary>
/// Represents a unit article that inherits from the Article class.
/// </summary>
public class UnitArticle : Article
{
    #region attributs

    private string _description;
    private float _unitPrice;
    private string _brand;

    #endregion

    #region constructeurs

    /// <summary>
    /// Initializes a new instance of the <see cref="UnitArticle"/> class.
    /// </summary>
    /// <param name="reference">The reference of the unit article.</param>
    /// <param name="description">The description of the unit article.</param>
    /// <param name="unitPrice">The unit price of the unit article.</param>
    /// <param name="brand">The brand of the unit article.</param>
    public UnitArticle(string reference, string description, float unitPrice, string brand)
        : base(reference, description, unitPrice, brand)
    {
        _description = description;
        _unitPrice = unitPrice;
        _brand = brand;
    }

    #endregion

    #region accesseurs

    /// <summary>
    /// Gets the description of the unit article. Inherited from the Article class.
    /// </summary>
    public override string Description => _description;

    /// <summary>
    /// Gets the unit price of the unit article. Inherited from the Article class.
    /// </summary>
    public override float UnitPrice => _unitPrice;

    /// <summary>
    /// Gets the brand of the unit article. Inherited from the Article class.
    /// </summary>
    public override string Brand => _brand;

    #endregion
}
```

Pouvez-vous faire un programme pour tester cette classe ? Si non, pourquoi ? Si oui, fournir le code du test.

Codez cette classe et fournir son code dans le compte-rendu.

```
namespace TP9
{
    /// <summary>
    /// Represents a bundle of articles.
    /// </summary>
    public class Bundle : Article
    {
        #region Fields

        /// <summary>
        /// Gets or sets the number of articles in the bundle.
        /// </summary>
        private int _number;

        /// <summary>
        /// Gets or sets the reduction rate for the bundle.
        /// </summary>
        private int _reductionRate;

        /// <summary>
        /// Represents the unit article that makes up the bundle.
        /// </summary>
        private Article _article;

        #endregion


        #region Constructor

        /// <summary>
        /// Initializes a new instance of the <see cref="Bundle"/> class.
        /// </summary>
        /// <param name="reference">The reference of the bundle.</param>
        /// <param name="number">The number of articles in the bundle.</param>
        /// <param name="article">The unit article that makes up the bundle.</param>
        /// <param name="rate">The reduction rate for the bundle.</param>
        public Bundle(string reference, int number, Article article, int rate) : base(reference)
        {
            this._number = number;
            this._reductionRate = rate;
            this._article = article;
        }

        #endregion

        #region Properties

        /// <summary>
```

```
    /// Gets the description of the bundle.
    /// </summary>
    public override string Description => $"Bundle of {this._number} {_article.Brand} articles";

    /// <summary>
    /// Gets the unit price of the bundle.
    /// </summary>
    public override float UnitPrice => this._number * _article.UnitPrice * (1 - this._reductionRate /
100.0f);

    /// <summary>
    /// Gets the brand of the bundle.
    /// </summary>
    public override string Brand => _article.Brand;

    #endregion

    // Additional methods to calculate total price and manage bundle information
  }
}
```

Réalisez un programme qui créée un article unitaire Pen (référence « P005 », marque « Boc », prix unitaire 1,35€, description à votre guise) et un article unitaire Ream (référence « R200 », marque « ClairRuisseau », prix unitaire 6,25€, description à votre guise). Créez un lot de 200 stylos avec la référence « BUN1 » et une réduction de 10%. Faites afficher dans la console le lot et l'article « Ream ». Fournissez la capture d'écran de l'affichage du programme.

```
using TP._09;

public class Program
{
  static void Main(string[] args)
  {

    #region Testing UnitArticle class

    // Create an instance of UnitArticle
    UnitArticle unitArticle = new UnitArticle("A002", "Pen", 1.5f, "Bic");

    // Print the details of the unit article
    Console.WriteLine(unitArticle.ToString());

    #endregion

    #region exercice 6

    // Create an instance of UnitArticle for Pen
    UnitArticle pen = new UnitArticle("P005", "Stylo Boc", 1.35f, "Boc");
```

```
        // Create an instance of UnitArticle for Ream
        UnitArticle ream1 = new UnitArticle("R200", "Rame de papier ClairRuisseau", 6.25f,
"ClairRuisseau");

        // Create a bundle of 200 pens with a 10% reduction
        Bundle penBundle = new Bundle("BUN1", 200, pen, 10);

        // Print the details of the pen bundle and the ream
        Console.WriteLine(penBundle.ToString());
        Console.WriteLine(ream1.ToString());

        #endregion

        #region Programme de test complet

        // Create a Person instance for the client "Paul Rich"
        Person client = new Person { FirstName = "Paul", LastName = "Rich" };

        // Create a Bill instance for the client
        Bill bill = new Bill(client);

        // Create UnitArticle instances for the green pens and the ream of paper
        UnitArticle greenPen = new UnitArticle("GreenPen00", "green pen", 1.05f, "boc");
        UnitArticle ream = new UnitArticle("Ream500", "500 sheets of white paper", 4.45f,
"ClairRuisseau");

        // Create Bundle instances for the bundle of blue pens and the bundle of elastics
        UnitArticle bluePen = new UnitArticle("BluePen00", "blue pen", 1.05f, "boc");
        UnitArticle elastic = new UnitArticle("ELA01", "simple yellow elastic", 0.07f, "Latex");

        //
        Bundle bluePenBundle = new Bundle("BluePen00", 20, bluePen, 6);
        Bundle elasticBundle = new Bundle("ELA01", 1000, elastic, 10);

        // Add the articles and bundles to the bill
        bill.AddArticle(greenPen);
        bill.AddArticle(greenPen);
        bill.AddArticle(ream);
        bill.AddArticle(bluePenBundle);
        bill.AddArticle(elasticBundle);

        // Display the bill in the console
        Console.WriteLine(bill.ToString());
        #endregion
    }
}
```

Codez la classe Person et la classe Bill. Fournir au compte-rendu le code de la classe Bill

```
using System.Text;
using System.Collections.Generic;

namespace TP._09;

/// <summary>
/// Represents a bill with a list of articles and a client.
/// </summary>
public class Bill
{
    #region attributs

    /// <summary>
    /// List of articles in the bill.
    /// </summary>
    private List<Article> _articles;

    /// <summary>
    /// The client associated with the bill.
    /// </summary>
    private Person _client;

    #endregion

    #region accesseurs

     /// <summary>
    /// Calculates the total price of all articles in the bill.
    /// </summary>
    /// <returns>The total price of all articles.</returns>
    public float TotalPrice()
    {
        float total = 0f;
        foreach (var art in _articles) { total += art.UnitPrice; }
        return total;
    }
    #endregion

    #region constructeurs

    public Bill(Person client)
        {
            _client = client;
            _articles = new List<Article>();
        }

    #endregion


    #region methodes

     /// <summary>
    /// Adds an article to the bill.
```

```
    /// </summary>
    /// <param name="article">The article to add.</param>
    public void AddArticle(Article article) => _articles.Add(article);

    /// <summary>
    /// Returns a string that represents the current bill.
    /// </summary>
    /// <returns>A string that represents the current bill.</returns>
    public override string ToString()
    {
        StringBuilder sb = new StringBuilder();
        sb.AppendLine($"Client : {_client.FirstName} {_client.LastName}");
        sb.AppendLine("---------------------------------");
        foreach (var article in _articles)
        {
            sb.AppendLine(article.ToString());
        }
        sb.AppendLine("---------------------------------");
        sb.AppendLine($"Total price : {TotalPrice()}");
        return sb.ToString();
    }

    #endregion

}
```

Fournir l'affichage obtenu sur la console par votre programme.

```
Client : Paul Rich
---------------------------------
Reference: GreenPen00, Description: green pen, Unit Price: 1,05, Brand: Boc
Reference: GreenPen00, Description: green pen, Unit Price: 1,05, Brand: Boc
Reference: BluePen00, Description: Bundle of 20 Boc articles, Unit Price: 15,415999, Brand: Boc
Reference: Ream500, Description: 500 sheets of white paper, Unit Price: 4,45, Brand: ClairRuisseau
Reference: ELA01, Description: Bundle of 1000 Latex articles, Unit Price: 63, Brand: Latex
---------------------------------
Total price : 84,966
```