Contrôle continu HAI717I – CC3 : 9 décembre 2022
Programmation par objets
(50 min)
Eléments de correction et barême très indicatif
En effet, le barême sera étendu sur plus de 20 points
pour prendre en compte la longueur du sujet

Université de Montpellier – Faculté Des Sciences
Master informatique (ICO), géomatique, bioinformatique, Physique numérique

Nous étudions la suite des éléments pour un logiciel de gestion des achats dans une station-service. Cette fois-ci, la station-service s'est modernisée et propose maintenant des bornes de rechargement pour les véhicules électriques. Les éléments vont vous être donnés au fil des questions.
On vous rappelle pour commencer quelques éléments du contrôle précédent. Attention, nous ne vous redonnons pas la classe **AchatCarburant** qui devra être réécrite.

```
public enum TypeCarburant {Gazole, SP98, SP95}

public class CarteConso {
	private String idCarte;
	private boolean professionnelle;
	public CarteConso() {}
	public CarteConso(String idCarte, boolean professionnelle) {
		this.idCarte = idCarte;
		this.professionnelle = professionnelle;
	}
	public String getIdCarte() {return idCarte;}
	public void setIdCarte(String idCarte) {this.idCarte = idCarte;}
	public boolean isProfessionnelle() {return professionnelle;}
	public void setProfessionnelle(boolean professionnelle)
		{this.professionnelle = professionnelle;}
	public String toString() {
		return "CarteConso [idCarte=" + idCarte + ",
			professionnelle=" + professionnelle + "]";
	}
}

public class Carburant {
	private TypeCarburant type;
	private double prixAuL;
	public Carburant() {}
	public Carburant(TypeCarburant type, double prixAuL)
		{this.type = type; this.prixAuL = prixAuL;}
	public String toString() {return "Carburant [type=" + type + ",
		prixAuL=" + prixAuL + "]";}
}
```

Nous ajoutons la description des types de bornes. Un type de borne est décrit de manière simplifiée par une énumération, comportant trois valeurs. L'énumération est de plus dotée d'une méthode qui retourne pour chaque valeur le prix en euro de la minute de rechargement.

```
public enum TypeBorne{
        AC43kw, DC50kw, DC175kw;
        public double prixMn()
        {
                switch (this) {
                case AC43kw: case DC50kw: return 0.30;
                case DC175kw: return 0.55;
                default: return 0;
                }
        }
}
```

Si vous n'avez pas eu l'occasion de manipuler des énumérations dotées de méthodes, voici deux exemples d'application de la méthode. La première application est réalisée directement sur une valeur littérale de l'énumération, la deuxième est réalisée sur une variable.

```
System.out.println(TypeBorne.AC43kw.prixMn());
TypeBorne tb = TypeBorne.AC43kw; System.out.println(tb.prixMn());
```

Question 1.

Écrivez les entêtes et attributs de trois classes représentant les achats d'énergie dans la station-service avec les informations suivantes :

- un achat d'énergie (classe **AchatEnergie**) est associé à une carte consommateur.
- un achat de carburant (classe **AchatCarburant**) est une sorte d'achat d'énergie qui, de plus, comporte un carburant et une quantité achetée en litres.
- un achat d'électricité (classe **AchatElectricite**) est une sorte d'achat d'énergie qui, de plus, comporte un type de borne et une durée en minutes.

Les accesseurs à ces attributs sont supposés exister pour la suite.

Réponse Question 1 (4 points).

```
public class AchatEnergie { //0,25
        private CarteConso carteConso; //0,75
}
public class AchatCarburant extends AchatEnergie{ //0,5
        private Carburant carburantAchete; // 0,5
        private double quantiteEnLitre; //0,5
}
public class AchatElectricite extends AchatEnergie{ //0,5
        private double dureeMn; // 0,5
        private TypeBorne borne; // 0,5
}
```

Question 2.
Écrivez pour chacune des trois classes représentant les achats d'énergie un constructeur permettant d'initialiser tous les attributs.

Réponse Question 2 (4 points).

```
public AchatEnergie(CarteConso carteConso)  // 0,5 point
        {this.carteConso=carteConso;}  // ou this.setCarteConso(carteConso) //0,5

public AchatCarburant(CarteConso carteConso, Carburant carburantAchete, double qteEnLitre) // 0,5
{
        super(carteConso); // 0,5
        this.carburantAchete = carburantAchete; // ou this.setCarburant(carburantAchete) //0,25
        this.quantiteEnLitre = qteEnLitre; // ou this.setQuantiteEnLitre (qteEnLitre) //0,25
}

public AchatElectricite(CarteConso carteConso, double dureeMn, TypeBorne borne) // 0,5
{
        super(carteConso); // 0,5
        this.dureeMn = dureeMn ; // ou this.setDureeMn(dureeMn) //0,25
        this.borne = borne; // ou this.setBorne(borne) //0,25
}
```

Question 3.
Écrivez pour chacune des trois classes représentant les achats d'énergie une méthode **prixSpecifique** qui retourne le prix de l'énergie achetée :
- Pour les achats de carburant, ce prix spécifique est le produit de la quantité en litres par le prix en litre du carburant.
- Pour les achats d'électricité, ce prix spécifique est le produit de la durée en minutes par le prix en minute prévu pour chaque type de borne.
L'écriture choisie pour la classe **AchatEnergie** vous demande-t-elle de modifier l'entête de la classe ? Si c'est le cas, indiquez comment.

Réponse Question 3 (4 points).

```
// Dans AchatEnergie
        public abstract double prixSpecifique(); // 1,5
// Dans AchatCarburant
        public double prixSpecifique() { // 1,5
                return this.quantiteEnLitre * this.carburantAchete.getPrixAuL();
        }
// Dans AchatElectricite
                public double prixSpecifique() { // 1,5
                return dureeMn * borne.prixMn();
        }
```

Question 4.
Écrivez la ou les méthodes nécessaires pour mettre en place le calcul du prix final (**prixFinal**) pour les achats d'énergie. Ce prix final est obtenu en appliquant au prix spécifique une réduction de 10% pour les cartes consommateur professionnelles. Quel schéma de programmation avez-vous mis en place (voir cours sur l'héritage) ?

Réponse Question 4 (4 points).
On n'écrit **qu'une seule** méthode dans la classe AchatEnergie. // 1

```
public double prixFinal() { //0 ,5
        if (this.carteConso.isProfessionnelle()) //0,5
            return prixSpecifique() * 0.9;  //0,5
        else return prixSpecifique(); //0,5
}
```

Il s'agit du schéma de programmation par généralisation. //1

Question 5.
Écrivez pour chacune des trois classes représentant les achats d'énergie une méthode **toString** retournant une chaîne de caractères comportant tous les attributs ainsi que la valeur du prix final. Quel schéma de programmation avez-vous mis en place (voir cours sur l'héritage) ?

Réponse Question 5 (4 points).

```
// Dans AchatEnergie // 1
    public String toString() {
        return "AchatEnergie [carteConso=" + carteConso + ", prixFinal()=" + prixFinal() + "]";
    }
// Dans AchatCarburant // 1
    public String toString() {
        return "AchatCarburant [carburantAchete=" + carburantAchete +
            ", quantiteEnLitre=" + quantiteEnLitre
            + ", toString()=" + super.toString() + "]";
    }
// Dans AchatElectricite // 1
    public String toString() {
        return "AchatElectricite [dureeMn=" + dureeMn +
            ", borne=" + borne + ", toString()="
            + super.toString() + "]";
    }
```

Il s'agit du schéma de programmation par spécialisation. //1
Nota : l'ordre d'écriture des attributs et de l'appel super.toString() peut être fait différemment.