SAE S2.01 – Développement d'une application avec une IHM

Génération des articles – Début du modèle métier

1. Présentation du modèle métier partiel

Ci-dessous, le diagramme UML de Classes de conception détaillée partielle que nous avons obtenu en programmant la SAE.

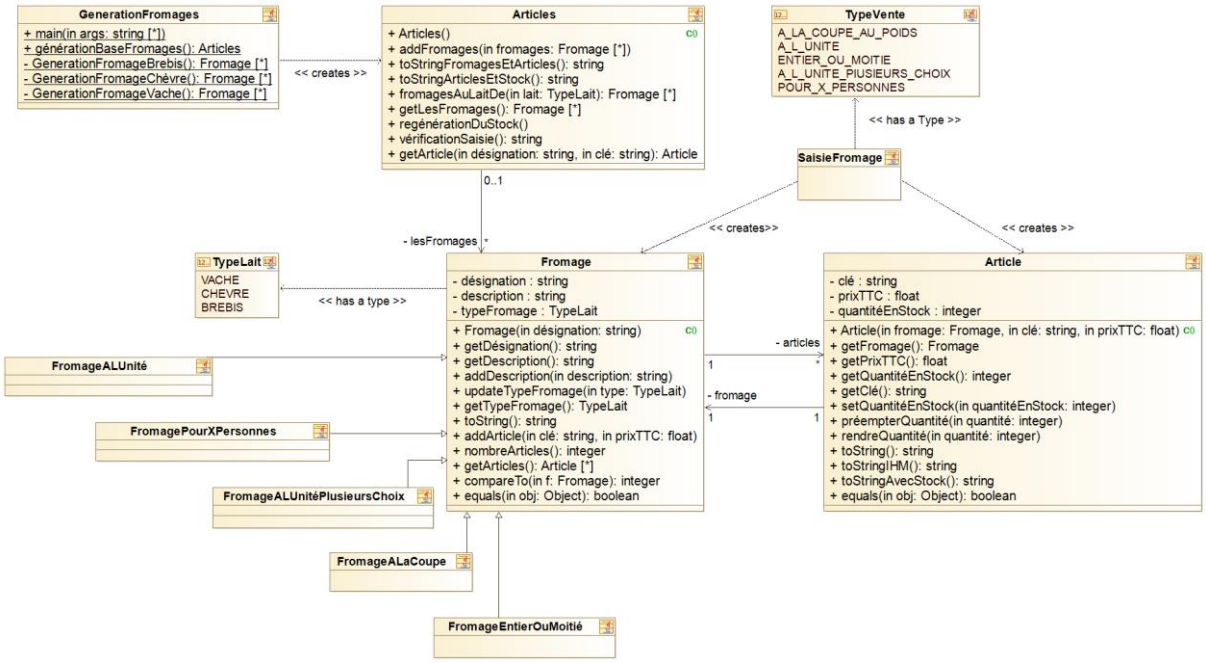

Figure 1 : notre modèle métier "partiel"

Le cœur de la base à objets est composé des classes Articles, Fromage et Article. La description et la désignation d'un fromage sont factorisés pour tous les articles correspondant à un même fromage. La classe Articles a la responsabilité de gérer l'ensemble des fromages et articles que l'entreprise propose.

La génération de la base de fromages et articles est assurée par la classe GenerationFromages qui utilise la classe SaisieFromage pour générer la base à partir d'une donnée semi-structurée à base de tableaux de chaînes de caractères et de réels.

2. Explications sur les types énumérés

2.1 Qu'est-ce qu'un type énuméré ?

Un type énuméré permet de définir un ensemble d'identificateurs pour des constantes ; il peut permettre par exemple de définir l'ensemble : LUNDI, MARDI, ... pour les jours de la semaine. A partir du JDK 5.0, on peut définir des types énumérés à l'aide de classes. Ces classes diffèrent des classes traditionnelles ; en particulier, leur en-tête commence par le mot `enum` à la place du mot `class`. L'ensemble des valeurs possibles qu'un type énuméré peut prendre est l'ensemble des constantes déclarées dans la clause `enum`. A noter que l'on peut ajouter de l'information à ces constantes en ajoutant des attributs privés, un constructeur privé et des accesseurs publics.

2.2 Comment nous en servons-nous dans l'application ?

Le type énuméré `TypeLait` permet de qualifier le type de lait d'un fromage. Nous nous en servons comme d'un filtre pour présenter les fromages au lait de vache, au lait de chèvre ou au lait de brebis, lorsque l'utilisateur le demande.

```
package modèle;

public enum TypeLait {
	VACHE("Vache"), CHEVRE("Chèvre"), BREBIS("Brebis");

	private String typeDeLait;

	private TypeLait(String typeDeLait) {
		this.typeDeLait = typeDeLait;
	}

	public String getTypeDeLait() {
		return this.typeDeLait;
	}
}
```

Le type énuméré `TypeVente` permet de qualifier la façon de construire des articles d'un fromage en fonction de son type de vente. Ce type énuméré nous a servi uniquement à construire « proprement » une base objets de fromages et articles. Des explications suivent à la section suivante.

```
package modèle;

public enum TypeVente {
A_LA_COUPE_AU_POIDS, A_L_UNITE, ENTIER_OU_MOITIE, A_L_UNITE_PlUSIEURS_CHOIX,
POUR_X_PERSONNES
}
```

3. Explications sur la génération des fromages

A partir des données trouvées sur le site de vente de fromages que nous avons « rewampé », il a fallu les organiser afin d'avoir un traitement uniforme pour en faire des objets java.

La stratégie envisagée a été d'utiliser un format de données en entrée assez générique pour prendre en compte tous les cas de figures dans un même formatage des données en entrée.

Donc à partir du type semi-structuré défini dans la classe `SaisieFromage` et du type énuméré `TypeVente`, nous avons construit un *builder* de fromage et articles par type de vente envisagé.

3.1 Type semi-structuré dans SaisieFromage

Le début du code de la classe `SaisieFromage` est donné ci-dessous :

```
private String désignation;
private String description;
private TypeVente vente;
private String[] cléArticle;
private float[] prixArticle;

public SaisieFromage(String désignation, String description, TypeVente vente) {
	this.désignation = désignation;
	this.description = description;
	this.vente = vente;
}

public SaisieFromage(String désignation, String description, TypeVente vente,
String[] cléArticle, float[] prixArticle) {
	this(désignation, description, vente);
	this.cléArticle = cléArticle;
	this.prixArticle = prixArticle;
}
```

La désignation, la description et le type de vente du fromage correspondent à la partie fixe d'un objet `SaisieFromage`. L'ensemble des clés et des prix correspondent à la partie variable de la donnée en entrée, elles dépendent du type de vente du fromage. Et donc le tout est un type semi-structuré. A noter la présence d'un chaînage entre constructeurs d'une même classe. Le premier des constructeurs correspond à la partie fixe d'un objet `SaisieFromage`.

3.2 Les formats en entrée diffèrent suivant le type de vente envisagé

Examinons les différents type de vente que nous avons envisagé. Rien ne vous interdit d'en rajouter si vous en avez le temps, l'audace et l'envie…

Le type de vente à l'unité ne nécessite qu'un prix et une clé vide. Une brique de brebis fermière ne se vend qu'à l'unité.

```
@Test
public void testSaisieFromageEtArticleALunité() {
	SaisieFromage uneSaisie  = new SaisieFromage("Brique de Brebis Fermière",
	"Cette brique est fabriquée à base de lait cru de brebis, elle dispose d'une
croûte tendre de couleur ivoire plissée "
	+"et d'une pâte onctueuse et fondante de couleur blanche. "
	+"Lors de sa dégustation vous découvrirez des saveurs fruitées, légèrement
salées avec des arômes de brebis, de noisette et de lait chaud. "
	+"Une authenticité qui révèle la diversité des pâturages du Tarn grâce à son
herbage riche et floral. "
	+ "Un vrai produit qui doit sa beauté simplement au fermier artisan. "
	+ "Elle est fondante, douce et persistante, prend du caractère sans
agressivité en s'affinant avec un petit goût de noisette.",
	TypeVente.A_L_UNITE,new String[]{""} ,new float[]{7.89F});
	Fromage f = uneSaisie.builderFromage();
	assertEquals("Brique de Brebis Fermière", f.getDésignation());
	assertEquals(1, f.nombreArticles());
	assertEquals("", f.getArticles().get(0).getClé());
	assertEquals(7.89F, f.getArticles().get(0).getPrixTTC(), 0F);
}
```

Le type de vente Entier ou Moitié nécessite deux prix référencés par les clés entier ou moitié. Le brin d'amour corse peut se vendre en entier ou par moitié. A noter que pour la gestion des stocks, nous avons simplifié en considérant un stock par article et non un stock unique de brin d'amour corse que l'on découperait à la demande.

```
@Test
public void testSaisieFromageEtArticlesEntierOuMoitié() {
	SaisieFromage uneSaisie = new SaisieFromage("Brin d'Amour Corse",
	"Ce fromage au lait cru de brebis à pâte molle à croûte fleurie, propose un
caractère corse qui s'exprime par sa croûte recouverte d'herbes, "
	+"de sarriettes et de romarin. Le Brin d'Amour offre un goût inimitable et
une saveur parfumée et intense, il reste une merveille pour le palais. "
	+"C'est tout le caractère du fromage corse vous faisant voyager qui
s'exprime dans ce mariage très heureux de lait de brebis et d'aromates. "
	+ "La «Fleur du maquis» dont la pâte est fine et la couleur varie entre le
rouge et le vert.",
	TypeVente.ENTIER_OU_MOITIE, new String[]{""} ,new float[]{8.5F, 16.8F});
	Fromage f = uneSaisie.builderFromage();
	assertEquals("Brin d'Amour Corse", f.getDésignation());
	assertEquals(2, f.nombreArticles());
	assertEquals("moitié", f.getArticles().get(0).getClé());
	assertEquals(8.5F, f.getArticles().get(0).getPrixTTC(), 0F);
	assertEquals("entier", f.getArticles().get(1).getClé());
	assertEquals(16.8F, f.getArticles().get(1).getPrixTTC(), 0F);
}
```

Le type de vente à l'unité plusieurs choix, un même fromage, ici le cabris fermier, a plusieurs textures possibles : cendré ou blanc. Dans ce cas un même prix est donné pour l'ensemble des choix possibles.

```
@Test
public void testSaisieFromageEtArticlesALunitéPlusieursChoix() {
	SaisieFromage uneSaisie = new SaisieFromage("Cabris Fermier",
	"Produit par nos soins sous le nom de l'EARL Chemin Fleury, nous respectons
le savoir faire de cette grande région, tout en se distinguant "
	+"par une texture et un goût qui lui sont bien spécifiques. Au lait cru de
chèvre, nos fromages présentent une croûte fine salée au sel blanc "
	+"ou cendré renfermant une pâte souple, tendre et homogène de couleur
blanche. "
	+"Leurs doux goûts de chèvre uniques raviront vos papilles et celles de vos
convives.",
	TypeVente.A_L_UNITE_PLUSIEURS_CHOIX, new String[]{"Moelleux cendré - Jeune
et doux", "Moelleux blanc - Jeune et doux"} ,new float[]{3.39F});
	Fromage f = uneSaisie.builderFromage();
	assertEquals("Cabris Fermier", f.getDésignation());
	assertEquals(2, f.nombreArticles());
	assertEquals("Moelleux cendré - Jeune et doux",
f.getArticles().get(0).getClé());
	assertEquals(3.39F, f.getArticles().get(0).getPrixTTC(), 0F);
	assertEquals("Moelleux blanc - Jeune et doux",
f.getArticles().get(1).getClé());
	assertEquals(3.39F, f.getArticles().get(1).getPrixTTC(), 0F);
}
```

Le type de vente A la coupe ou Au poids, le brie de Melun se vend au poids : 250 grammes, 500 grammes ou un kilogramme. A chaque mesure de poids est affecté un prix. Nous ne connaissons pas le poids d'un brie de Melun complet. Nous avons alors fait la même hypothèse que pour les fromages entier ou moitié. A chaque article correspond un stock distinct.

```
@Test
public void testSaisieFromageEtArticlesALaCoupe() {
	SaisieFromage uneSaisie = new SaisieFromage("Brie de Melun",
	"Plus petit que son grand frère «le Brie de Meaux», il a cependant plus de caractère. Ce fromage au lait cru de vache vous offrira une pâte "
	+"souple de couleur jaune d'or à l'intérieur, et une croûte fleurie blanche parsemée de stries ou de taches rouges ou brunes à l'extérieur. "
	+"Le Brie de Melun vous proposera une saveur très fruitée avec un léger goût de noisette accompagnée d'une odeur du terroir qui le rendra "
	+"indispensable sur un plateau de fromages. Il pourra aussi entrer dans la confection de spécialités régionales dont la plus connue, "
	+"la croûte au brie.",
	TypeVente.A_LA_COUPE_AU_POIDS, new String[]{"250 g", "500 g", "1 Kg"}, new float[]{9.15F, 18.3F, 36.6F});
	Fromage f = uneSaisie.builderFromage();
	assertEquals("Brie de Melun", f.getDésignation());
	assertEquals(3, f.nombreArticles());
	assertEquals("250 g", f.getArticles().get(0).getClé());
	assertEquals(9.15F, f.getArticles().get(0).getPrixTTC(), 0F);
	assertEquals("500 g", f.getArticles().get(1).getClé());
	assertEquals(18.3F, f.getArticles().get(1).getPrixTTC(), 0F);
	assertEquals("1 Kg", f.getArticles().get(2).getClé());
	assertEquals(36.6F, f.getArticles().get(2).getPrixTTC(), 0F);
}
```

Le type de vente pour X personnes correspond principalement à des fromages pour de la fondue. La fondue est vendue par portions correspondant à un nombre de personnes. Nous avons alors fait la même hypothèse que pour les fromages entier ou moitié. A chaque article correspond un stock distinct.

```
@Test
public void testSaisieFromageEtArticlesPourXPersonnes() {
	SaisieFromage uneSaisie = new SaisieFromage("Fondue Savoyarde",
	"Idéal pour un repas convivial avec vos amis ou en famille. C'est un mélange de parfum, d'onctuosité, et de raffinement pour "
	+"cette fondue savoyarde. Allez y piquez vos morceaux de pain dans le caquelon au centre de la table. "
	+"La fondue savoyarde est un plat régional de la gastronomie française à base de fromage fondu et de pain, traditionnel des pays de Savoie. "
	+"Ce plat populaire, vous envoûtera de part ses multiples arômes.",
	TypeVente.POUR_X_PERSONNES, new String[]{"3","5", "10"}, new float[]{22.5F, 33.75F, 67.5F});
	Fromage f = uneSaisie.builderFromage();
	assertEquals("Fondue Savoyarde", f.getDésignation());
	assertEquals(3, f.nombreArticles());
	assertEquals("pour 3 personnes", f.getArticles().get(0).getClé());
	assertEquals(22.5F, f.getArticles().get(0).getPrixTTC(), 0F);
	assertEquals("pour 5 personnes", f.getArticles().get(1).getClé());
	assertEquals(33.75F, f.getArticles().get(1).getPrixTTC(), 0F);
	assertEquals("pour 10 personnes", f.getArticles().get(2).getClé());
	assertEquals(67.5F, f.getArticles().get(2).getPrixTTC(), 0F);
}
```

3.3 Dans notre modélisation objet, à un fromage correspond une liste d'articles, peu importe le type de vente

Certes, cela introduit certaines redondances, mais simplifie grandement l'écriture du code java des modèles métiers et IHM. Cette uniformisation de la représentation des données nous a simplifié le

travail. Le code de la transformation d'une donnée en entrée en Fromage et articles associés est contenu par la méthode `buiderFromage` :

```
public Fromage builderFromage() {
	Fromage f = null;
	switch (vente) {
	case A_LA_COUPE_AU_POIDS :
		f = new FromageALaCoupe(désignation);
		if (this.prixArticle != null && this.prixArticle.length > 1
		&& this.cléArticle != null && this.cléArticle.length > 1
		&& this.cléArticle.length == this.prixArticle.length)
			for (int i = 0; i < this.cléArticle.length; i++)
				f.addArticle(this.cléArticle[i], this.prixArticle[i]);
		break;
	case A_L_UNITE :
		f = new FromageALUnité(désignation);
		if (this.prixArticle != null)
			f.addArticle("", this.prixArticle[0]);
		break;
	case ENTIER_OU_MOITIE :
		f = new FromageEntierOuMoitié(désignation);
		if (this.prixArticle != null && this.prixArticle.length == 2) {
			f.addArticle(FromageEntierOuMoitié.MOITIE,this.prixArticle[0]);
			f.addArticle(FromageEntierOuMoitié.ENTIER,this.prixArticle[1]);
		}
		break;
	case A_L_UNITE_PLUSIEURS_CHOIX :
		f = new FromageALUnitéPlusieursChoix(désignation);
		if (this.prixArticle != null && this.prixArticle.length == 1
		&& this.cléArticle != null && this.cléArticle.length > 1)
			for (String clé: this.cléArticle)
				f.addArticle(clé, this.prixArticle[0]);
		break;
	case POUR_X_PERSONNES :
		f = new FromagePourXPersonnes(désignation);
		if (this.prixArticle != null && this.prixArticle.length > 1
		&& this.cléArticle != null && this.cléArticle.length > 1
		&& this.cléArticle.length == this.prixArticle.length)
			for (int i = 0; i < this.cléArticle.length; i++)
				f.addArticle("pour " + this.cléArticle[i] + "
				personnes", this.prixArticle[i]);
		break;
	}
	f.addDescription(description);
	return f;
}
```

Comme vous pouvez le constater, les règles de bonne formation d'une entrée sont dans une clause if présente pour chaque type de vente envisagé.

4. Ce que l'on vous donne

Vous pouvez si vous le voulez ajouter des choses, des nouveaux types de lait, des nouveaux types de vente, des nouveaux articles à générer, des nouveaux attributs, des nouvelles méthodes, mais à priori ces classes sont suffisantes.

4.1 Tel quel

- Classe Fromage

- Classe FromageALaCoupe
- Classe FromageALUnitéPlusieursChoix
- Classe FromageEntierOuMoitié
- Classe FromagePourXPersonnes
- Classe GenerationFromages
- Classe SaisieFromage
- Classe TestSaisieFromage
- Enumération TypeLait
- Enumération TypeVente

Les autres classes sont à compléter. Vous constaterez que l'implémentation donnée ne couvre pas toutes les opérations définies par le diagramme de classes de conception. Vous pouvez les ajouter ou ajouter vos propres opérations.

4.2 A compléter

- Classe Articles
- Classe Article