

# Chapitre 16 - Modélisation UML pour le cas GPAO

## Analyse du cas GPAO

L'exercice demande de créer un modèle entité/relation selon la méthode UML pour une entreprise de fabrication de camions jouets en plastique (GPAO). Les données fournies dans le tableau 2.3 présentent les articles gérés par l'entreprise.

### Données du tableau

- **Réf:** CD100, CC201, CA000, CA001
- **Désignation:** Camion déménagement bleu, Camion citerne rouge, Cabine montée bleue, Cabine montée rouge
- **Type fabrication ou achat:** Fab. par lot / Fab. à la commande
- **Unité achat ou stock:** À l'unité
- **Délai (sem.):** 2 (pour tous les articles)
- **Prix standard:** 200, 150, 250, 150
- **Lot de réappro.:** 200, 150, 250, 150
- **Stock mini.:** vide (sauf 600, 600, 750 pour les trois premiers)
- **Stock maxi.:** 600, 600, 750, vide

## Modèle UML proposé

Voici le diagramme de classes UML pour modéliser cette gestion de production:

```

+-----+ . . . +-----+
| . . . Article . . . | . . . Stock . . . |
+-----+ . . . +-----+
| -reference: String|1| -quantite: int . . |
| -designation: String|<>----| -stockMin: int . . |
| -prixStandard: float| . . . | -stockMax: int . . |
| -delai: int . . . | . . . | -lotReappro: int |
+-----+ . . . +-----+
          ^
          |
+-----+ . . . |
| . . . . . . . |
+-----+ . . . +-----+
| .. Camion | . . . | .. Cabine . . |
+-----+ . . . +-----+
| -couleur: String| | -couleur: String|
+-----+ . . . +-----+

+-----+
| TypeFabrication . . |
+-----+
| +PAR_LOT . . . |<---+
| +A_LA_COMMANDE . . . |
+-----+ . . . |
          ^
          |
+-----+ . . . |
| ModeProduction |-----+
+-----+
| -type: TypeFabrication|
| -unite: String . . |
+-----+
          ^
          |
          +
          / \
          ---+
          |
+-----+
| . . . Article . . . |

```

## Code Java correspondant

java

```
// Énumération pour Le type de fabrication
enum TypeFabrication {
    ... PAR_LOT,
    ... A_LA_COMMANDE
}

// Classe pour gérer Le mode de production
class ModeProduction {
    ... private TypeFabrication type;
    ... private String unite;

    ...
    ... public ModeProduction(TypeFabrication type, String unite) {
        this.type = type;
        this.unite = unite;
    }

    ... // Getters et setters
    ... public TypeFabrication getType() { return type; }
    ... public String getUnite() { return unite; }
}

// Classe pour gérer Le stock
class Stock {
    ... private int quantite;
    ... private int stockMin;
    ... private int stockMax;
    ... private int lotReappro;

    ...
    ... public Stock(int quantite, int stockMin, int stockMax, int lotReappro) {
        this.quantite = quantite;
        this.stockMin = stockMin;
        this.stockMax = stockMax;
        this.lotReappro = lotReappro;
    }

    ...
    ... // Getters et setters
    ... public int getQuantite() { return quantite; }
    ... public int getStockMin() { return stockMin; }
    ... public int getStockMax() { return stockMax; }
    ... public int getLotReappro() { return lotReappro; }

    ...
    ... // Méthodes métier
    ... public boolean besoinReapprovisionnement() {
        ... return quantite < stockMin;
    }
}
```

```

// Classe abstraite Article
abstract class Article {
    private String reference;
    private String designation;
    private float prixStandard;
    private int delai;
    private Stock stock;
    private ModeProduction modeProduction;

    public Article(String reference, String designation, float prixStandard,
                  int delai, Stock stock, ModeProduction modeProduction) {
        this.reference = reference;
        this.designation = designation;
        this.prixStandard = prixStandard;
        this.delai = delai;
        this.stock = stock;
        this.modeProduction = modeProduction;
    }

    // Getters et setters
    public String getReference() { return reference; }
    public String getDesignation() { return designation; }
    public float getPrixStandard() { return prixStandard; }
    public int getDelai() { return delai; }
    public Stock getStock() { return stock; }
    public ModeProduction getModeProduction() { return modeProduction; }
}

// Sous-classe Camion
class Camion extends Article {
    private String couleur;

    public Camion(String reference, String designation, float prixStandard,
                 int delai, Stock stock, ModeProduction modeProduction, String couleur) {
        super(reference, designation, prixStandard, delai, stock, modeProduction);
        this.couleur = couleur;
    }

    // Getters et setters spécifiques
    public String getCouleur() { return couleur; }
}

// Sous-classe Cabine
class Cabine extends Article {
    private String couleur;
}

```

```

    ... public Cabine(String reference, String designation, float prixStandard,
    ...                 int délai, Stock stock, ModeProduction modeProduction, String couleur) {
    ...     super(reference, designation, prixStandard, délai, stock, modeProduction);
    ...     this.couleur = couleur;
    ... }

    ... // Getters et setters spécifiques
    public String getCouleur() { return couleur; }
}

// Classe principale pour démontrer l'utilisation
public class SystemeGPAO {
    public static void main(String[] args) {
        // Création des stocks
        Stock stockCD100 = new Stock(0, 600, 600, 200);
        Stock stockCC201 = new Stock(0, 600, 600, 150);
        Stock stockCA000 = new Stock(0, 750, 750, 250);
        Stock stockCA001 = new Stock(0, 0, 0, 150); // Pas de stock min/max défini

        // Création des modes de production
        ModeProduction parLot = new ModeProduction(TypeFabrication.PAR_LOT, "À l'unité");
        ModeProduction aLaCommande = new ModeProduction(TypeFabrication.A_LA_COMMANDE, "À l'uni

        // Création des articles
        Camion camionDemenagement = new Camion("CD100", "Camion déménagement bleu",
                                                200.0f, 2, stockCD100, parLot, "bleu");

        Camion camionCiterne = new Camion("CC201", "Camion citerne rouge",
                                         150.0f, 2, stockCC201, aLaCommande, "rouge");

        Cabine cabineBleue = new Cabine("CA000", "Cabine montée bleue",
                                         250.0f, 2, stockCA000, parLot, "bleue");

        Cabine cabineRouge = new Cabine("CA001", "Cabine montée rouge",
                                         150.0f, 2, stockCA001, aLaCommande, "rouge");

        // Démonstration
        System.out.println("Articles du système GPAO:");
        System.out.println(camionDemenagement.getReference() + " - " + camionDemenagement.getDe
        System.out.println(camionCiterne.getReference() + " - " + camionCiterne.getDesignation(
        System.out.println(cabineBleue.getReference() + " - " + cabineBleue.getDesignation());
        System.out.println(cabineRouge.getReference() + " - " + cabineRouge.getDesignation());
    }
}

```

## Explication du modèle

**1. Classe Article:** Classe abstraite représentant les articles de l'entreprise, avec leurs attributs communs (référence, désignation, prix, délai).

**2. Classes spécifiques:**

- **Camion:** Hérite d'Article et ajoute la propriété couleur
- **Cabine:** Hérite d'Article et ajoute également la propriété couleur

**3. Classe Stock:** Gère les informations de stock pour chaque article (quantité, stock min/max, lot de réapprovisionnement).

**4. Classe ModeProduction:** Encapsule les informations sur le mode de fabrication (par lot ou à la commande) et l'unité (à l'unité dans ce cas).

**5. Énumération TypeFabrication:** Définit les types de fabrication possibles (PAR\_LOT, A\_LA\_COMMANDE).

## Avantages du modèle proposé

- **Extensibilité:** On peut facilement ajouter de nouveaux types d'articles en créant de nouvelles sous-classes d'Article.
- **Encapsulation:** Chaque classe a une responsabilité bien définie.
- **Relations claires:** Les associations entre classes représentent bien les relations métier.
- **Réutilisabilité:** Les composants comme Stock et ModeProduction peuvent être utilisés par différents types d'articles.

Ce modèle UML reflète fidèlement les données présentées dans le tableau 2.3 tout en offrant une structure orientée objet bien organisée pour gérer la production de jouets.