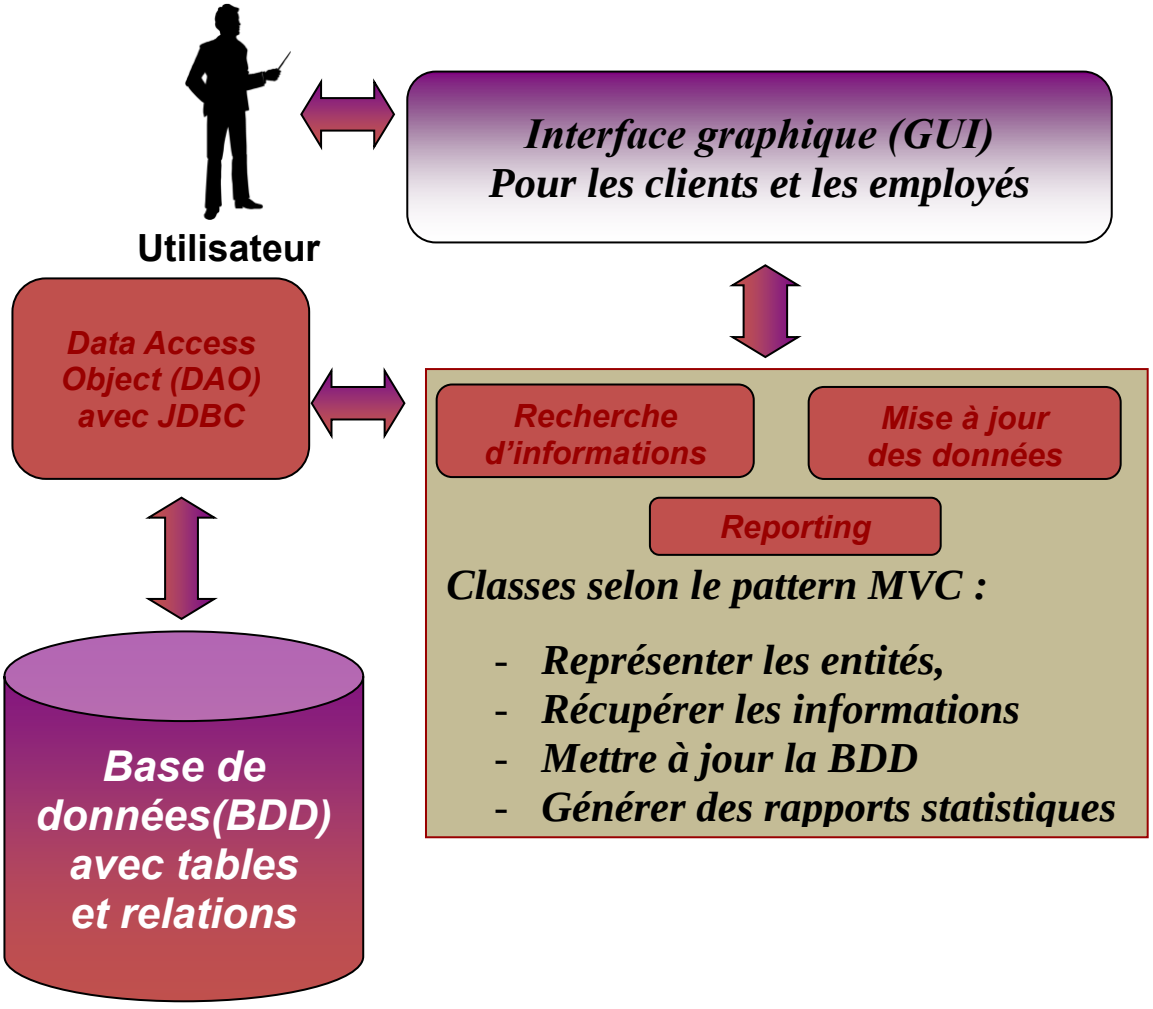
# Projet Java ING3 2025 : Thème Attractions

## Table des matières

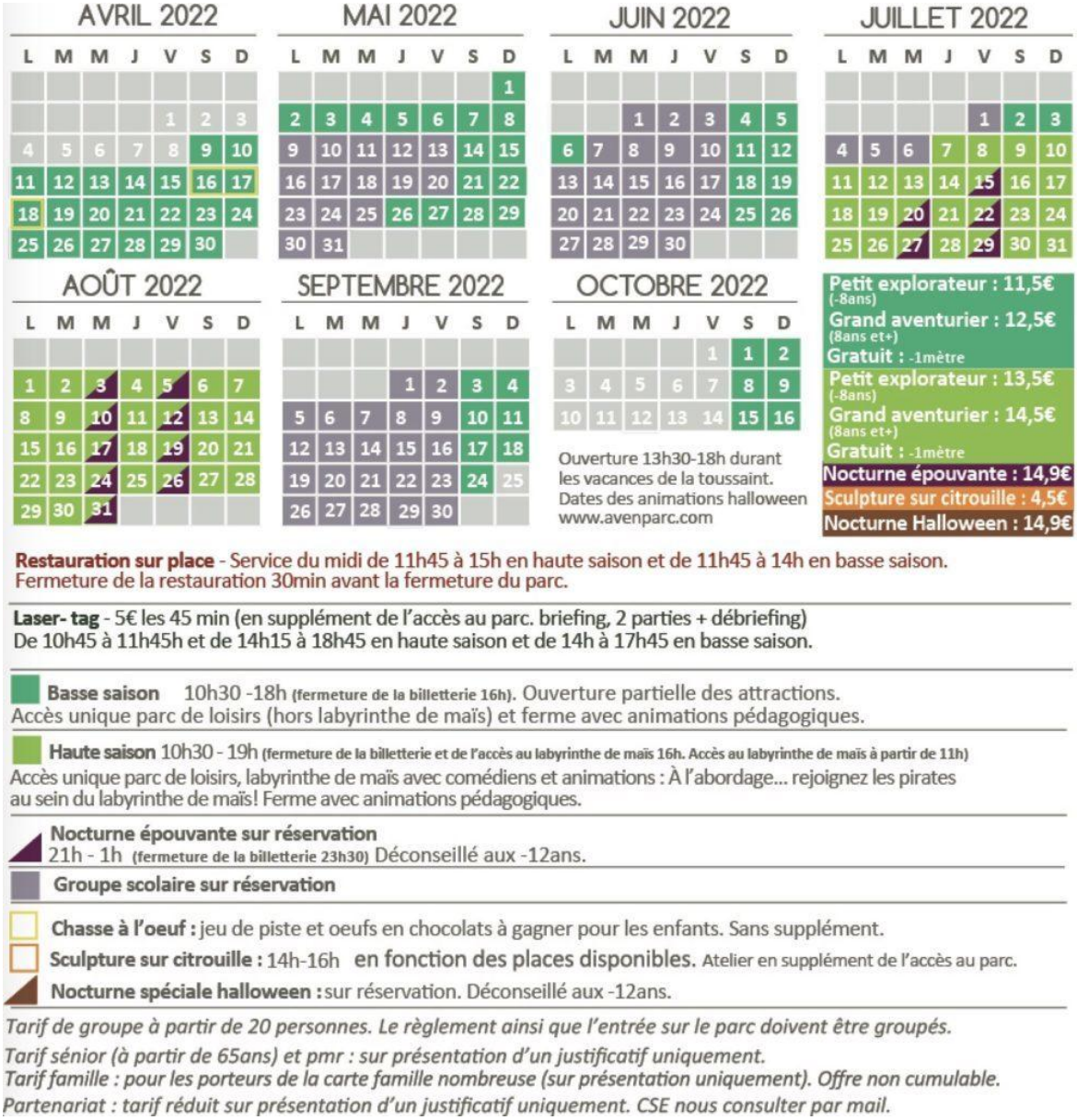
## Objectif du projet

Le but de ce thème "**Attractions**" du projet Java est d'écrire une application qui permettra au client de réserver des attractions, avec remises possibles et facturation. L'application aidera également l'organisation commerciale de conserver et mettre à jour ses archives des ventes et des clients.



## Description du programme

Le but de ce thème "**Attractions**" du projet Java est d'écrire une application sur ordinateur pour gérer des rendez-vous chez des spécialistes d'attractions. **Voici ci-dessous un exemple d'interface graphique** :




## Description du programme

Dans ce programme, vous devez gérer un ensemble de classes de support pour une application de demande de réservation de billets de ce parc d'attractions. Vous devez créer un écran factice pour indiquer le traitement des paiements.

L'application doit permettre aux clients de parcourir toutes les attractions disponibles de différents types. Les clients pourront réserver des billets de l'attraction pour la date choisie.

Les clients peuvent être de deux types, soient nouveaux clients ou anciens clients :

- Nouveaux Clients : ils devront s'inscrire avec un identifiant et un mot de passe.
- Anciens clients : ils devront s'inscrire avec un identifiant et un mot de passe membres. Ils pourront accéder à leur historique des attractions passées. Ces clients pourront réserver leurs billets avec réduction offerte en fonction du type de membre (assiduité at âge).

Les clients invités n'auront pas besoin de se connecter et pourront réserver les billets sans aucune réduction offerte.

Tout client pourra être régulier, senior ou enfant, donnant droit à une réduction selon l'âge. Les clients membres auront besoin d'un login et

L'application implique principalement des détails sur les manèges disponibles, leurs caractéristiques et leurs prix ainsi que les commandes clients générées et maintenues.

Cette application doit être développée pour les deux types d'utilisateurs suivants avec un certain nombre de fonctionnalités pour chacun de ces utilisateurs :

**1. Clients** : réserver le trajet choisi, calcul de la facture avec / sans réductions, calculer et visualiser la facture des attractions achetées avec éventuelles remises, naviguer dans la disponibilité des attractions, etc.
**2. Administrateurs de l'application** : mettre à jour les attractions, introduire diverses offres de réduction, maintenir les dossiers des clients, identifier les attractions les plus populaires, etc.

**Vous devez d'abord concevoir et développer la base de données de cette application ainsi que les classes Java nécessaires à la mise en œuvre de l'application.**

## Exigences de mise en œuvre

- Les classes, méthodes et attributs nécessaires doivent être conçus en utilisant la notation de diagramme UML. Toutes les classes, méthodes et attributs doivent être expliqués dans votre documentation. Veuillez discuter de la conception avec moi avant de commencer la mise en œuvre.
- Vous devez être en mesure d'identifier et d'introduire une relation d'héritage et d'agrégation, le cas échéant
- Les écrans GUI nécessaires doivent être ajoutés pour une exécution réussie.
- Les enregistrements doivent être conservés dans la base de données. Votre code Java est censé lire et écrire dans plusieurs tables selon les besoins.
- Chaque table doit contenir au moins 6 enregistrements.

## Architecture générale du système (Modèle MVC + DAO + JDBC)

Dans cette section, l'architecture générale de gestion de ce planning a été décrite. Ce système compte principalement 5 modules :

• Le module **Recherche d'informations** : toutes les demandes possibles dans la base de données, selon plusieurs critères de recherche
• Le module **Mise à jour des Données** : toute opération de modification, d'ajout ou de suppression dans la base de données
• Le module **Reporting** : statistiques sous forme de graphiques (camemberts, histogrammes etc.)
• Le module **Data Access Object (DAO)** interroge ou met à jour les données dans la base de données, via JDBC (Java Data Base Connectivity), et communique avec les 3 modules précédents
• **L'Interface graphique** (GUI) communique avec les 3 premiers modules pour visualiser graphiquement les informations extraites de la base de données
Selon le pattern **MVC**, votre interface graphique constitue la ***Vue*** (uniquement l'affichage) dépendante des actions de l'utilisateur (gestion des événements) au niveau du ***Contrôleur*** (modules Recherche, Mise à jour et reporting). Celui-ci demandera au Modèle de récupérer ou de mettre à jour - via le module d'accès aux données (DAO) - les informations de la base de données, de les organiser et de les assembler (par exemple, en les stockant dans des collections). Ensuite, le contrôleur demandera les données au modèle, les analysera, prendra des décisions et renverra le texte à la vue.
Il vous est conseillé d'adopter le modèle MVC pour le développement d'un projet cohérent. Vous pouvez en savoir plus sur le modèle MVC avec les ressources suivantes :

- Modèle-Vue-Contrôleur **[wikipedia]**
- Structurez une application avec le pattern d'architecture MVC **[openclassrooms]**
- Organisez votre code Java avec l'architecture MVC **[openclassrooms]**
- Écrivez du code Java maintenable avec MVC **[openclassrooms]**
- Implémentez le contrôleur et la vue de votre application **[openclassrooms]**

# Conseils pour le développement structuré du projet

## Étape 1 : Modèle relationnel

Passez en revue toutes les exigences possibles de la base de données et les critères de recherche. Identifiez les entités possibles, les attributs dans la base de données. Il est important de reconnaître soigneusement le rôle de chaque attribut, puis de décider du type de données de l'attribut. Il jouera également un rôle clé dans la détermination des attributs de clé primaire et étrangère. Documenter les relations entre les entités.

## Étape 2 : Création de la base de données

Sur la base du modèle relationnel ci-dessus, créez des tables et des relations à l'aide de MySQL. Insérez les enregistrements dans les tables.

## Étape 3 : Recherche d'informations

Passez en revue les exigences de l'utilisateur pour identifier la gamme possible d'informations que vous devez extraire de la base de données. Plus précisément dans le cas des organisations commerciales, il est important que les employés analysent les ventes. Il est également essentiel pour les clients d'analyser leurs achats passés.

Par exemple :

• Le montant de la vente d'un produit particulier
• Le nombre de commandes passées par le client au cours des trois derniers mois
• Les offres de réduction qui ont été très bien accueillies

Développez les classes nécessaires pour représenter les entités qui permettront à l'utilisateur d'interroger la base de données.

### Étape 4 : Accès aux données

Ce module représente la couche d'accès aux données (DAO) dans la base de données. Via un accès JDBC à la base de données, ce module exécute les requêtes chargées de récupérer ou de mettre à jour les données de la base de données. Il s'agit d'un type d'objet qui se charge pour récupérer les données dans la base de données et qu'un autre type d'objet soit utilisé pour manipuler ces données (couche métier).

### Étape 5 : interface graphique (GUI) et création de rapports (reporting)

Une fenêtre d'accueil permettra à l'utilisateur de se connecter à la base de données en saisissant son EMAIL et son MOT DE PASSE. Ces informations, si elles sont stockées dans une table USER, lui donneront des droits d'accès ou/et de mise à jour sur certaines données du planning.

Votre interface graphique affichera de manière ergonomique, claire et fluide toutes les informations pertinentes. Il vous permettra de naviguer intuitivement d'une page à l'autre. Par exemple, une page de votre graphique d'interface peut contenir des menus avec des éléments de menu ou des onglets si vous préférez.

Ce module permet de générer des statistiques (camemberts, histogrammes, etc.) à l'aide de JFreeChart. Vous pouvez trouver les détails dans la section de la page campus Cours : Projet POO Java, Section : Ressources (campusonline.me).

## Livrables

Vous devez déposer les 2 livrables suivants aux deadlines indiquées sur cette page BoostCamp Section : Livrable à déposer | Projet POO Java S6 2024-2025 | BOOSTCAMP :

**1)** Livrable de conception PowerPoint (format **ppt**, **pptx** ou **pdf**) avec les slides suivants, évalué hors soutenances, dont la deadline est le dimanche 20/04/2025 23h, est à déposer dans le lien Livrable de conception PowerPoint à déposer : deadline le dimanche 20/04/2025 23h en bas de cette page, en respectant le contenu suivant :

- Page de garde avec titre, noms coéquipiers et groupe de TD (1 slide)
- Sommaire (1 slide)
- **Répartition des tâches** par fonctionnalités sous forme de tableau (1 slide)
- **Diagramme de classes** avec l'outil **Draw.io** ou équivalent, selon les patterns **MVC** et **DAO** et présentant les attributs (pas d'attribut objet !), les méthodes, les cardinalités, sans constructeurs ni getters/setters (1 slide ou plus si le diagramme est illisible : par exemple, 1 slide présente l'architecture générale du pattern MVC avec seulement les noms des classes, puis un slide pour chacun des 3 types de modules MVC détaillant le contenu des classes).
- **Design de la maquette de votre interface graphique** principalement composée des 2 éléments suivants :
  - Le *storyboard (voir le lien* **Exemple de storyboard***)* : liens entre les pages, symbolisés par une flèche pour naviguer d'une page à une autre (1 à 2 slides)
  - Des *wireframes (voir le lien* **Exemples de wireframes***)* de certaines de vos pages (par exemple, 1 page pour une recherche, 1 pour une mise à jour et 1 autre pour le Reporting)

: composants graphiques Swing légendés avec les conteneurs encadrés, leur mise en page layout (au plus 3 slides)

- **Versioning GIT** : screenshot et lien avec login et passwd, montrant clairement la bonne utilisation et la répartition des tâches sur la (ou les) version(s) du code partagé entre coéquipiers (1 slide)
- **Bilan individuel et collectif** sans blabla (par exemple sous forme de tableau) sur l'état du travail effectué, des compétences acquises et des points d'amélioration. (1 à 2 slides).
- **Sources** : web avec les liens, livres, supports de cours en citant les auteurs. **Toute source non citée est considérée de facto comme un plagiat**. (1 slide)

**2)** Livrable final du code du projet (version soutenance) au format **.zip** ou .**rar** doit contenir les éléments suivants :

- Tous les dossiers et fichiers du projet développé de préférence sur l'IDE **IntelliJ** avec vos fichiers sources **.java.**
- Les librairies **.jar** nécessaires (exemples : accès au serveur de la BDD, JFreeChart, etc.).
- L'exécutable **.jar** en mode graphique de votre programme.
- Tous les fichiers nécessaires au bon fonctionnement de votre projet (exemples : fichier **.sql** de votre base de données, images .au format **.png** ou autres, etc.).
- La documentation **Javadoc** commentée pour les classes et les méthodes (respectez bien le format Javadoc des commentaires /** … */ au-dessus de chaque classe et méthode de votre code).

Ce livrable final, dont la deadline est le dimanche 27/04/2025 23h, est à déposer dans le lien Livrable final du code du projet (version soutenance) à déposer : deadline le dimanche 27/04/2025 23h, en bas de cette page, en respectant son contenu ci-dessus.

## Ressources externes sur campus

Les ressources suivantes se trouvent dans la section de la page campus

Section : Ressources | Projet POO Java S6 2024-2025 | BOOSTCAMP

- Le **pattern MVC** :

- Modèle-Vue-Contrôleur **[wikipedia]**
- Structurez une application avec le pattern d'architecture MVC **[openclassrooms]**
- Organisez votre code Java avec l'architecture MVC **[openclassrooms]**
- Écrivez du code Java maintenable avec MVC **[openclassrooms]**
- Implémentez le contrôleur et la vue de votre application **[openclassrooms]**

- Le **pattern DAO** :

  Le lien ci-dessous vers openclassroom montre une vidéo suivie d'un code source qui charge le DAO sans savoir si celui-ci stocke dans une base de données MySQL ou ailleurs :
  **Utiliser le modèle DAO**

- **Le pattern DAO (Data Access Object) : manipulez vos données grâce aux DAO**

Le lien ci-dessus montre comment le pattern DAO (Data Access Object) fait en sorte que les données de votre base de données collent à vos objets, à l'aide des méthodes de récupération, de création, de mise à jour et (ou) de suppression, et cela de manière stable. Le mappage des données est, en fait, le mécanisme visant à faire correspondre les attributs d'une fiche du système de stockage (BDD) avec les attributs d'un objet (objet Java en ce qui nous concerne).

- **JDBC** (Java Data Base Connectivity) : Développons en Java - JDBC (Java DataBase Connectivity) (jmdoudoux.fr) (Auteur: Jean-Michel Doudoux)

- **JFreeChart:**
  - The JFreeChart Class Library (Auteur : David Gilbert)
  - http://www.jfree.org/jfreechart/api/javadoc/index.html
  - http://www.java2s.com/Code/Java/Chart/CatalogChart.htm
  - http://www.jfree.org/forum/

- **Storyboard** : Exemple de StoryBoard
- **Wireframe:** Exemples de WireFrame