Bibiane Gagné-Caron  
Rita Merveille Matsinkou

Programmation Orientée Objets

Groupe : 01  
420-N26

Travail Pratique 2

Travail présenté à :  
Charles Jacob

Département d’informatique  
CEGEP Régional de Lanaudière à Joliette

28/05/2025

# Attributs et Méthodes du Système de Simulation Aéroportuaire

## **Classe Aeronef (abstraite)**

### Description Classe de base représentant tous les aéronefs. Elle définit les propriétés essentielles (nom, vitesse, état) et des méthodes générales comme le déplacement et le changement d’état.

### Attributs

| **Nom** | **Type** | **Description** |
| --- | --- | --- |
| Nom | string | Nom de l'aéronef |
| Vitesse | double | Vitesse de déplacement de l'aéronef |
| TempsEntretien | double | Temps nécessaire pour l'entretien |
| type | TypeAeronef | Type d'aéronef (Passager, Cargo, Secours, etc.) |
| typeEtat | TypeEtat | Type d'état actuel |
| PositionActuelle | Position | Position géographique actuelle |
| PositionDestination | Position | Position de destination |
| EtatActuel | EtatAeronef | État comportemental actuel |

### Méthodes

| **Nom** | **Type de retour** | **Description** |
| --- | --- | --- |
| Aeronef() | constructeur | Constructeur par défaut |
| Aeronef(string, double, double, TypeEtat) | constructeur | Constructeur avec paramètres (protégé) |
| CreerEtatDepuisType(TypeEtat) | EtatAeronef | Crée un état à partir d'un type (privé) |
| Avancer(Position) | void | Fait avancer l'aéronef vers une destination |
| MettreAJourPosition(double) | void | Met à jour la position selon un ratio |
| ChangerEtat(TypeEtat) | void | Change l'état de l'aéronef |
| ToString() | string | Représentation textuelle |
| Serialiser() | string | Sérialise l'objet |

Classe AeronefTransport (abstraite)

### **Description**

Spécialise Aeronef pour les aéronefs **de transport**, ajoutant des attributs liés à l’embarquement et au débarquement.

### Attributs

| **Nom** | **Type** | **Description** |
| --- | --- | --- |
| TempsEmbarquement | double | Temps nécessaire pour l'embarquement |
| TempsDebarquement | double | Temps nécessaire pour le débarquement |

### Méthodes

| **Nom** | **Type de retour** | **Description** |
| --- | --- | --- |
| AeronefTransport() | constructeur | Constructeur par défaut |
| AeronefTransport(string, double, double, double, double, TypeEtat) | constructeur | Constructeur avec paramètres (protégé) |
| ToString() | string | Représentation textuelle |
| Serialiser() | string | Sérialise l'objet |

## Classe AvionPassager

### Description

Modèle d’avion conçu pour **transporter des passagers**, avec une capacité en nombre de sièges.

### Attributs

| **Nom** | **Type** | **Description** |
| --- | --- | --- |
| Capacite | int | Nombre maximum de passagers |

### Méthodes

| **Nom** | **Type de retour** | **Description** |
| --- | --- | --- |
| AvionPassager() | constructeur | Constructeur par défaut |
| AvionPassager(string, double, double, int, double, double, TypeEtat) | constructeur | Constructeur complet |
| ToString() | string | Représentation textuelle |
| Serialiser() | string | Sérialise l'objet |

## Classe AvionCargaison

### Description

Conçu pour transporter des marchandises, avec une charge maximale exprimée en poids.

### Attributs

| **Nom** | **Type** | **Description** |
| --- | --- | --- |
| Capacite | double | Capacité de chargement en poids |

### Méthodes

| **Nom** | **Type de retour** | **Description** |
| --- | --- | --- |
| AvionCargaison() | constructeur | Constructeur par défaut |
| AvionCargaison(string, double, double, double, double, double, TypeEtat) | constructeur | Constructeur complet |
| ToString() | string | Représentation textuelle |
| Serialiser() | string | Sérialise l'objet |

## Classe AvionSecours

### Description

Aéronef d’urgence utilisé pour les missions de sauvetage et d’assistance.

### Méthodes

| **Nom** | **Type de retour** | **Description** |
| --- | --- | --- |
| AvionSecours() | constructeur | Constructeur par défaut |
| AvionSecours(string, double, double, TypeEtat) | constructeur | Constructeur avec paramètres |
| ToString() | string | Représentation textuelle |
| Serialiser() | string | Sérialise l'objet |

## Classe AvionCiterne

Aéronef capable de transporter et larguer des liquides, souvent utilisé pour la lutte contre les incendies.

### Méthodes

| **Nom** | **Type de retour** | **Description** |
| --- | --- | --- |
| AvionCiterne() | constructeur | Constructeur par défaut |
| AvionCiterne(string, double, double, TypeEtat) | constructeur | Constructeur avec paramètres |
| ToString() | string | Représentation textuelle |
| Serialiser() | string | Sérialise l'objet |

## Classe Helicoptere

### Description

Aéronef d’intervention **plus flexible** pouvant décoller et atterrir sans infrastructure lourde.

### Méthodes

| **Nom** | **Type de retour** | **Description** |
| --- | --- | --- |
| Helicoptere() | constructeur | Constructeur par défaut |
| Helicoptere(string, double, double, TypeEtat) | constructeur | Constructeur avec paramètres |
| ToString() | string | Représentation textuelle |
| Serialiser() | string | Sérialise l'objet |

## Classe Client (abstraite)

### Description

Classe de base représentant tout **client nécessitant un aéronef** (transport, urgence, observation)

### Attributs

| **Nom** | **Type** | **Description** |
| --- | --- | --- |
| position | Position | Position géographique du client |

### Méthodes

| **Nom** | **Type de retour** | **Description** |
| --- | --- | --- |
| Client() | constructeur | Constructeur par défaut (protégé) |
| Client(Position) | constructeur | Constructeur avec position (protégé) |
| Traiter(Aeronef) | void | Traite l'interaction avec un aéronef |
| estTermine() | bool | Vérifie si le client a terminé |
| Clone() | Client | Crée une copie du client |

## Classe ClientTransport (abstraite)

### Description

Modèle de client nécessitant un **transport vers un aéroport**, souvent un passager ou une cargaison.

### Attributs

| **Nom** | **Type** | **Description** |
| --- | --- | --- |
| Destination | Aeroport | Aéroport de destination |

**Méthodes**

| **Nom** | **Type de retour** | **Description** |
| --- | --- | --- |
| ClientTransport() | constructeur | Constructeur par défaut (protégé) |
| ClientTransport(Position, Aeroport) | constructeur | Constructeur avec paramètres (protégé) |

**10. Classe Passager**

**Attributs**

| **Nom** | **Type** | **Description** |
| --- | --- | --- |
| NbPassagers | int | Nombre de passagers |
| termine | bool | Statut de fin de traitement (privé) |

**Méthodes**

| **Nom** | **Type de retour** | **Description** |
| --- | --- | --- |
| Passager() | constructeur | Constructeur par défaut |
| Passager(Position, Aeroport, int) | constructeur | Constructeur complet |
| Traiter(Aeronef) | void | Traite l'embarquement |
| estTermine() | bool | Vérifie si terminé |
| Clone() | Client | Crée une copie |

## Classe Cargo

**Attributs**

| **Nom** | **Type** | **Description** |
| --- | --- | --- |
| PoidsCargaison | double | Poids de la cargaison |
| termine | bool | Statut de fin de traitement (privé) |

**Méthodes**

| **Nom** | **Type de retour** | **Description** |
| --- | --- | --- |
| Cargo() | constructeur | Constructeur par défaut |
| Cargo(Position, Aeroport, double) | constructeur | Constructeur complet |
| Traiter(Aeronef) | void | Traite le chargement |
| estTermine() | bool | Vérifie si terminé |
| Clone() | Client | Crée une copie |

**12. Classe Secours**

**Attributs**

| **Nom** | **Type** | **Description** |
| --- | --- | --- |
| termine | bool | Statut de fin de mission (privé) |

**Méthodes**

| **Nom** | **Type de retour** | **Description** |
| --- | --- | --- |
| Secours() | constructeur | Constructeur par défaut |
| Secours(Position) | constructeur | Constructeur avec position |
| Traiter(Aeronef) | void | Traite l'intervention |
| estTermine() | bool | Vérifie si mission terminée |
| Clone() | Client | Crée une copie |

**13. Classe Incendie**

**Attributs**

| **Nom** | **Type** | **Description** |
| --- | --- | --- |
| Intensite | int | Intensité de l'incendie |
| eteint | bool | État d'extinction (privé) |

**Méthodes**

| **Nom** | **Type de retour** | **Description** |
| --- | --- | --- |
| Incendie() | constructeur | Constructeur par défaut |
| Incendie(Position, int) | constructeur | Constructeur avec paramètres |
| Traiter(Aeronef) | void | Traite l'extinction |
| estTermine() | bool | Vérifie si éteint |
| Clone() | Client | Crée une copie |

**14. Classe Observation**

**Attributs**

| **Nom** | **Type** | **Description** |
| --- | --- | --- |
| termine | bool | Statut de fin d'observation (privé) |

**Méthodes**

| **Nom** | **Type de retour** | **Description** |
| --- | --- | --- |
| Observation() | constructeur | Constructeur par défaut |
| Observation(Position) | constructeur | Constructeur avec position |
| Traiter(Aeronef) | void | Traite l'observation |
| estTermine() | bool | Vérifie si terminé |
| Clone() | Client | Crée une copie |

**15. Classe FabriqueAeronef (Singleton)**

**Attributs**

| **Nom** | **Type** | **Description** |
| --- | --- | --- |
| instance | FabriqueAeronef | Instance unique (statique, privé) |
| padlock | object | Verrou pour thread-safety (statique, privé) |
| Instance | FabriqueAeronef | Propriété d'accès à l'instance (statique, public) |

**Méthodes**

| **Nom** | **Type de retour** | **Description** |
| --- | --- | --- |
| FabriqueAeronef() | constructeur | Constructeur privé |
| CreerAeronef(string, TypeAeronef, double, double, double, double, double, TypeEtat) | Aeronef | Crée un aéronef selon le type |

**16. Classe FabriqueClient (Singleton)**

**Attributs**

| **Nom** | **Type** | **Description** |
| --- | --- | --- |
| instance | FabriqueClient | Instance unique (statique, privé) |
| padlock | object | Verrou pour thread-safety (statique, privé) |
| Instance | FabriqueClient | Propriété d'accès à l'instance (statique, public) |

**Méthodes**

| **Nom** | **Type de retour** | **Description** |
| --- | --- | --- |
| FabriqueClient() | constructeur | Constructeur privé |
| CreerClient(Evenement) | Client | Crée un client selon l'événement |

**17. Classe Aeroport**

**Attributs**

| **Nom** | **Type** | **Description** |
| --- | --- | --- |
| m\_dernierAeronefEnvoye | Aeronef | Dernier aéronef envoyé (privé) |
| Nom | string | Nom de l'aéroport |
| Position | Position | Position géographique |
| MinPassagers | int | Nombre minimum de passagers |
| MaxPassagers | int | Nombre maximum de passagers |
| MinCargaisons | double | Poids minimum de cargaison |
| MaxCargaisons | double | Poids maximum de cargaison |
| Aeronefs | List<Aeronef> | Liste des aéronefs |
| Clients | List<Client> | Liste des clients |

**Méthodes**

| **Nom** | **Type de retour** | **Description** |
| --- | --- | --- |
| Aeroport() | constructeur | Constructeur par défaut |
| Aeroport(string, Position, int, int, double, double) | constructeur | Constructeur complet |
| GetAeronefDisponible(TypeEvenement) | Aeronef | Trouve un aéronef disponible |
| EnvoyerAeronef(Aeronef, Position) | void | Envoie un aéronef vers une destination |
| SaveLastAeronef(Aeronef) | void | Sauvegarde le dernier aéronef envoyé |
| GetAeronefs() | List<Aeronef> | Retourne la liste des aéronefs |
| AjouterClient(Client) | void | Ajoute un client |
| GetClients() | List<Client> | Retourne la liste des clients |
| Clone() | Aeroport | Crée une copie de l'aéroport |

**18. Classe Scenario**

**Attributs**

| **Nom** | **Type** | **Description** |
| --- | --- | --- |
| m\_aeroport | List<Aeroport> | Liste des aéroports |
| m\_frequence | List<FrequenceEvenement> | Fréquences des événements |
| m\_observateurs | List<IObservateur> | Liste des observateurs (privé) |
| clientsEvenements | List<Client> | Clients d'événements (privé) |
| HeureActuelle | double | Heure actuelle de simulation |
| dernieresGenerations | Dictionary<TypeEvenement, double> | Dernières générations d'événements (privé) |
| rnd | Random | Générateur aléatoire (statique, privé) |

**Méthodes**

| **Nom** | **Type de retour** | **Description** |
| --- | --- | --- |
| Scenario() | constructeur | Constructeur par défaut |
| Attacher(IObservateur) | void | Attache un observateur |
| Detacher(IObservateur) | void | Détache un observateur |
| NotifierObservateur(Evenement) | void | Notifie les observateurs |
| AjouterEvenementClient(Client) | void | Ajoute un client d'événement |
| DoitGenerer(TypeEvenement, double) | bool | Détermine si un événement doit être généré |
| GetAeroportProche(Position) | Aeroport | Trouve l'aéroport le plus proche |
| TraiterEvenement(Evenement) | Aeronef | Traite un événement |
| GetObservateurs() | List<IObservateur> | Retourne la liste des observateurs |
| GetAeroports() | List<Aeroport> | Retourne la liste des aéroports |
| GetAeroportAleatoire() | Aeroport | Retourne un aéroport aléatoire |
| GetAeroportAleatoireDifferent(Aeroport) | Aeroport | Retourne un aéroport différent |
| GenererPassagers() | void | Génère des événements passagers |
| GenererCargos() | void | Génère des événements cargo |
| GenererEvenementPour(TypeEvenement) | void | Génère un événement spécifique |
| GenererEvenementsSelonFrequence() | void | Génère événements selon fréquences |
| VerifierEtDeclencherEmbarquement() | void | Vérifie et déclenche l'embarquement |
| CreateMemento() | ScenarioMemento | Crée un memento |
| RestoreMemento(ScenarioMemento) | void | Restaure un memento |

**19. Classe ScenarioMemento**

**Attributs**

| **Nom** | **Type** | **Description** |
| --- | --- | --- |
| Aeroports | List<Aeroport> | Liste des aéroports sauvegardés |

**Méthodes**

| **Nom** | **Type de retour** | **Description** |
| --- | --- | --- |
| ScenarioMemento(List<Aeroport>) | constructeur | Constructeur avec liste d'aéroports |

**20. Classe CaretakerScenario**

**Attributs**

| **Nom** | **Type** | **Description** |
| --- | --- | --- |
| \_mementoInitial | ScenarioMemento | Memento initial (privé) |

**Méthodes**

| **Nom** | **Type de retour** | **Description** |
| --- | --- | --- |
| EnregistrerEtatInitial(Scenario) | void | Enregistre l'état initial |
| RestaurerEtatInitial(Scenario) | void | Restaure l'état initial |

**21. Classe Evenement**

**Attributs**

| **Nom** | **Type** | **Description** |
| --- | --- | --- |
| typeEvenement | TypeEvenement | Type d'événement |
| position | Position | Position de l'événement |
| Intensite | int | Intensité (pour incendies) |
| NombrePassagers | int | Nombre de passagers |
| PoidsCargo | double | Poids de la cargaison |
| Vitesse | double | Vitesse associée |
| Destination | Aeroport | Aéroport de destination |
| Aeroports | List<Aeroport> | Liste des aéroports disponibles |

**Méthodes**

| **Nom** | **Type de retour** | **Description** |
| --- | --- | --- |
| NotifierObservateurs() | void | Notifie tous les observateurs |

**22. Classes d'État (EtatAeronef et ses implémentations)**

**EtatAeronef (Interface)**

| **Nom** | **Type de retour** | **Description** |
| --- | --- | --- |
| AvancerPas(double) | void | Avance d'un pas de temps |
| GetTypeEtat() | TypeEtat | Retourne le type d'état |
| GererEtat(Aeronef) | void | Gère l'état de l'aéronef |

**Implémentations (EtatVol, EtatSol, EtatEntretien, EtatEmbarquement, EtatDebarquement, EtatEnMission)**

Chaque classe d'état implémente les méthodes de l'interface avec des comportements spécifiques.

**23. Classe Simulateur**

**Attributs**

| **Nom** | **Type** | **Description** |
| --- | --- | --- |
| scenario | Scenario | Scénario de simulation (privé) |
| timersDeplacements | Dictionary<Aeronef, Timer> | Timers pour les déplacements (privé) |
| simulationTimer | Timer | Timer principal de simulation (privé) |
| simulationEnCours | bool | État de la simulation (privé) |
| pas | double | Pas de temps de simulation |

**Méthodes**

| **Nom** | **Type de retour** | **Description** |
| --- | --- | --- |
| Simulateur() | constructeur | Constructeur par défaut |
| TraiterEvenement(Evenement) | bool | Traite un événement |
| GetScenario() | Scenario | Retourne le scénario |
| DemarrerSimulation() | void | Démarre la simulation |
| LancerSimulationAuto() | void | Lance la simulation automatique |
| ArreterSimulation() | void | Arrête la simulation |
| AvancerUnPas() | void | Avance d'un pas |
| AvancerPlusieursPas(int) | void | Avance de plusieurs pas |
| InitialiserClient() | void | Initialise les clients |
| AfficherVols() | void | Affiche les vols |
| ChargerScenario(string) | void | Charge un scénario |
| GenererEvenementPour(TypeEvenement) | void | Génère un événement |

**Événements**

| **Nom** | **Type** | **Description** |
| --- | --- | --- |
| PositionChanged | Action<Aeronef> | Déclenché lors du changement de position |
| Arrivee | Action<Aeronef> | Déclenché à l'arrivée |
| OnAeronefEnvoye | Action<Aeronef> | Déclenché lors de l'envoi d'aéronef |

**24. Classe FacadeSimulateur**

**Attributs**

| **Nom** | **Type** | **Description** |
| --- | --- | --- |
| simulateur | Simulateur | Instance du simulateur (privé) |

**Méthodes**

| **Nom** | **Type de retour** | **Description** |
| --- | --- | --- |
| FacadeSimulateur() | constructeur | Constructeur par défaut |
| TraiterEvenement(Evenement) | bool | Traite un événement |
| AttacherObservateur(IObservateur) | void | Attache un observateur |
| ChargerScenario(string) | void | Charge un scénario |
| DemarrerSimulation() | void | Démarre la simulation |
| LancerSimulationAuto() | void | Lance la simulation automatique |
| ArreterSimulation() | void | Arrête la simulation |
| AvancerUnPas() | void | Avance d'un pas |
| AvancerPlusieursPas(int) | void | Avance de plusieurs pas |
| CreerClient(Evenement) | void | Crée un client |
| AjouterClient() | void | Ajoute un client |
| CreerAeronef() | void | Crée un aéronef |

**Événements**

| **Nom** | **Type** | **Description** |
| --- | --- | --- |
| OnPositionChanged | Action<Aeronef> | Changement de position |
| OnMessage | Action<string> | Message système |
| OnAeronefEnvoye | Action<Aeronef> | Aéronef envoyé |

**25. Classe ControleurSimulateur**

**Méthodes**

| **Nom** | **Type de retour** | **Description** |
| --- | --- | --- |
| Initialiser() | void | Initialise le contrôleur |
| Notifier(Evenement) | void | Notifie d'un événement |

**26. Classe form\_Simulateur**

**Attributs**

| **Nom** | **Type** | **Description** |
| --- | --- | --- |
| controleur | ControleurSimulateur | Contrôleur associé (privé) |
| nomVersAeroport | Dictionary<string, Aeroport> | Mapping nom vers aéroport (privé) |
| marqueurAeronef | Dictionary<Aeronef, PictureBox> | Marqueurs visuels (privé) |
| timerDeplacement | Dictionary<Aeronef, Timer> | Timers de déplacement (privé) |

**Méthodes**

| **Nom** | **Type de retour** | **Description** |
| --- | --- | --- |
| form\_Simulateur() | constructeur | Constructeur par défaut |
| GetMarqueurAeronef(Aeronef) | PictureBox | Retourne le marqueur visuel |
| LancerDeplacement(Aeronef) | void | Lance l'animation de déplacement |
| SetControleur(ControleurSimulateur) | void | Définit le contrôleur |
| Notifier(Evenement) | void | Traite les notifications |
| AfficherAeroportsSurCarte(List<Aeroport>) | void | Affiche les aéroports |
| AfficherEvenementSurCarte(Evenement) | void | Affiche un événement |
| ActiverBoutonsPas(bool) | void | Active/désactive les boutons |
| ExecuterUI(Control, Action) | void | Exécute une action sur l'UI (statique) |

**27. Classes Utilitaires**

**FrequenceEvenement**

| **Nom** | **Type** | **Description** |
| --- | --- | --- |
| Type | TypeEvenement | Type d'événement |
| Frequence | double | Fréquence de génération |

**Position**

| **Nom** | **Type** | **Description** |
| --- | --- | --- |
| Latitude | double | Latitude géographique |
| Longitude | double | Longitude géographique |

**GestionnaireFichierXML**

| **Nom** | **Type de retour** | **Description** |
| --- | --- | --- |
| Importer(string) | Scenario | Importe un scénario (statique) |
| Exporter(Scenario, string) | void | Exporte un scénario (statique) |

**28. Énumérations**

**TypeAeronef**

* Passager
* Cargo
* Secours
* Citerne
* Helicoptere

**TypeEtat**

* Entretien
* Vol
* Sol
* Embarquement
* Debarquement

**29. Interfaces**

**IObservateur**

| **Nom** | **Type de retour** | **Description** |
| --- | --- | --- |
| Notifier(Evenement) | void | Reçoit une notification d'événement |

**30. ISujet**

| **Nom** | **Type de retour** | **Description** |
| --- | --- | --- |
| Attacher(IObservateur) | void | Attache un observateur |
| Detacher(IObservateur) | void | Détache un observateur |
| NotifierObservateur(Evenement) | void | Notifie les observateurs |

**IObservateurEvenement**

| **Nom** | **Type de retour** | **Description** |
| --- | --- | --- |
| Notifier(Evenement) | void | Notifie d'un événement spécifique |