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

# Table des matières

[Attributs et Méthodes du Système de Simulation Aéroportuaire 6](#_Toc199329860)

[Classe Aeronef (abstraite) 6](#_Toc199329861)

[Description 6](#_Toc199329862)

[Attributs 6](#_Toc199329863)

[Méthodes 6](#_Toc199329864)

[Classe AeronefTransport (abstraite) 7](#_Toc199329865)

[**Description** 7](#_Toc199329866)

[Attributs 7](#_Toc199329867)

[Méthodes 7](#_Toc199329868)

[Classe AvionPassager 8](#_Toc199329869)

[Description 8](#_Toc199329870)

[Attributs 8](#_Toc199329871)

[Méthodes 8](#_Toc199329872)

[Classe AvionCargaison 8](#_Toc199329873)

[Description 8](#_Toc199329874)

[Attributs 8](#_Toc199329875)

[Méthodes 9](#_Toc199329876)

[Classe AvionSecours 9](#_Toc199329877)

[Description 9](#_Toc199329878)

[Méthodes 9](#_Toc199329879)

[Classe AvionCiterne 9](#_Toc199329880)

[Description 9](#_Toc199329881)

[Méthodes 10](#_Toc199329882)

[Classe Helicoptere 10](#_Toc199329883)

[Description 10](#_Toc199329884)

[Méthodes 10](#_Toc199329885)

[Classe Client (abstraite) 10](#_Toc199329886)

[Description 10](#_Toc199329887)

[Attributs 11](#_Toc199329888)

[Méthodes 11](#_Toc199329889)

[Classe ClientTransport (abstraite) 11](#_Toc199329890)

[Description 11](#_Toc199329891)

[Attributs 11](#_Toc199329892)

[Méthodes 11](#_Toc199329893)

[Classe Passager 12](#_Toc199329894)

[Description 12](#_Toc199329895)

[Attributs 12](#_Toc199329896)

[Méthodes 12](#_Toc199329897)

[Classe Cargo 12](#_Toc199329898)

[Description 12](#_Toc199329899)

[Attributs 12](#_Toc199329900)

[Méthodes 13](#_Toc199329901)

[Classe Secours 13](#_Toc199329902)

[Description 13](#_Toc199329903)

[Attributs 13](#_Toc199329904)

[Méthodes 13](#_Toc199329905)

[Classe Incendie 13](#_Toc199329906)

[Description 13](#_Toc199329907)

[Attributs 14](#_Toc199329908)

[Méthodes 14](#_Toc199329909)

[Classe Observation 14](#_Toc199329910)

[Description 14](#_Toc199329911)

[Attributs 14](#_Toc199329912)

[Méthodes 14](#_Toc199329913)

[Classe FabriqueAeronef (Singleton) 15](#_Toc199329914)

[Description 15](#_Toc199329915)

[Attributs 15](#_Toc199329916)

[Méthodes 15](#_Toc199329917)

[Classe FabriqueClient (Singleton) 15](#_Toc199329918)

[Description 15](#_Toc199329919)

[Attributs 15](#_Toc199329920)

[Méthodes 16](#_Toc199329921)

[Classe Aeroport 16](#_Toc199329922)

[Description 16](#_Toc199329923)

[Attributs 16](#_Toc199329924)

[Méthodes 17](#_Toc199329925)

[Classe Scenario 17](#_Toc199329926)

[Description 17](#_Toc199329927)

[Attributs 17](#_Toc199329928)

[Méthodes 18](#_Toc199329929)

[Classe ScenarioMemento 19](#_Toc199329930)

[Description 19](#_Toc199329931)

[Attributs 19](#_Toc199329932)

[Méthodes 19](#_Toc199329933)

[Classe CaretakerScenario 20](#_Toc199329934)

[Description 20](#_Toc199329935)

[Attributs 20](#_Toc199329936)

[Méthodes 20](#_Toc199329937)

[Classe Evenement 20](#_Toc199329938)

[Descritpion 20](#_Toc199329939)

[Attributs 20](#_Toc199329940)

[Méthodes 21](#_Toc199329941)

[Classes d'État (EtatAeronef et ses implémentations) 21](#_Toc199329942)

[Description 21](#_Toc199329943)

[EtatAeronef (Interface) 21](#_Toc199329944)

[Classe Simulateur 21](#_Toc199329945)

[Description 21](#_Toc199329946)

[Attributs 21](#_Toc199329947)

[Méthodes 22](#_Toc199329948)

[Événements 23](#_Toc199329949)

[Classe FacadeSimulateur 23](#_Toc199329950)

[Description 23](#_Toc199329951)

[Attributs 23](#_Toc199329952)

[Méthodes 23](#_Toc199329953)

[Événements 24](#_Toc199329954)

[Classe ControleurSimulateur 24](#_Toc199329955)

[Description 24](#_Toc199329956)

[Méthodes 24](#_Toc199329957)

[Classe form\_Simulateur 24](#_Toc199329958)

[Description 24](#_Toc199329959)

[Attributs 24](#_Toc199329960)

[Méthodes 25](#_Toc199329961)

[Classes Utilitaires 25](#_Toc199329962)

[FrequenceEvenement 25](#_Toc199329963)

[Position 26](#_Toc199329964)

[GestionnaireFichierXML 26](#_Toc199329965)

[Énumérations 26](#_Toc199329966)

[TypeAeronef 26](#_Toc199329967)

[TypeEtat 26](#_Toc199329968)

[Interfaces 27](#_Toc199329969)

[IObservateur 27](#_Toc199329970)

[ISujet 27](#_Toc199329971)

[IObservateurEvenement 27](#_Toc199329972)

[Relation entre les classes 27](#_Toc199329973)

[Patrons GOF 30](#_Toc199329974)

[Singleton 30](#_Toc199329975)

[Fabrique/Fabrique Abstraite 30](#_Toc199329976)

[Observateur 30](#_Toc199329977)

[État 30](#_Toc199329978)

[Mémento 30](#_Toc199329979)

[Façade 31](#_Toc199329980)

[Composite (Potentiellement) 31](#_Toc199329981)

# Attributs et Méthodes du Système de Simulation Aéroportuaire

## **Classe Aeronef (abstraite)**

### Description

Classe abstraite qui définit les attributs et comportements de base de tous les aéronefs de la simulation. Elle gère l’état, la position et les déplacements des appareils, ainsi que leur cycle de vie dans le scénario. Sert de fondation pour spécialiser différents types d’aéronefs.

### 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

Classe abstraite dérivée d’Aeronef, utilisée comme base pour tous les aéronefs destinés au transport de passagers ou de marchandises. Elle ajoute la gestion des temps d’embarquement et de 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

Classe représentant un avion dédié au transport de passagers. Elle gère la capacité d’accueil, les opérations d’embarquement et de débarquement, et les interactions avec les clients de type passager.

### 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

Classe représentant un avion conçu pour le transport de marchandises. Elle gère la capacité de chargement, les opérations de livraison, et les interactions avec les clients de type cargo.

### 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

Classe représentant un avion affecté aux missions de secours et d’assistance. Elle gère l’exécution rapide de missions urgentes, comme les sauvetages ou l’intervention lors d’incidents.

### 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

### Description

Classe représentant un avion-citerne, généralement utilisé pour la lutte contre les incendies ou le transport de liquides spécialisés. Permet d’intervenir rapidement sur des événements critiques du scénario.

### 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

Classe représentant un hélicoptère, capable d’assurer divers types de missions, telles que le transport, le secours ou la reconnaissance, grâce à sa flexibilité opérationnelle.

### 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 abstraite qui représente une demande ou un client dans le système (passager, cargaison, intervention…). Définit l’interface commune pour le traitement des besoins par les aéronefs.

### 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

Classe abstraite pour les clients ayant besoin d’être transportés d’un aéroport à un autre. Gère la destination et les interactions avec les aéronefs de transport.

### 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) |

## Classe Passager

### Description

Classe représentant un client qui demande le transport d’un nombre déterminé de passagers d’un aéroport à un autre.

### 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

### Description

Classe représentant un client qui demande le transport d’une cargaison de poids défini entre deux aéroports.

### 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 |

## Classe Secours

### Description

Classe représentant une demande de secours émise par un client ou générée par un événement. Déclenche l’envoi d’un aéronef d’urgence pour une intervention spécifique.

### 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 |

## Classe Incendie

### Description

Classe représentant un événement d’incendie à gérer dans la simulation. Gère l’intensité, l’extinction du feu, et l’intervention des aéronefs appropriés.

### 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 |

## Classe Observation

### Description

Classe représentant une demande d’observation ou de reconnaissance aérienne. Peut servir à collecter des données ou surveiller une zone spécifique via un aéronef.

### 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 |

## Classe FabriqueAeronef (Singleton)

### Description

Singleton responsable de la création centralisée des objets Aeronef du système, selon les paramètres et le type désiré. Facilite la gestion et l’extension des différents types d’aéronefs.

### 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 |

## Classe FabriqueClient (Singleton)

### Description

Singleton responsable de la création centralisée des objets Client en fonction des événements survenus dans la simulation. Assure la cohérence et la modularité de l’ajout de nouveaux types de clients.

### 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 |

## Classe Aeroport

### Description

Classe représentant un aéroport dans la simulation. Gère la localisation, la flotte d’aéronefs disponibles, les clients présents et la gestion des mouvements d’aéronefs.

### 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 |

## Classe Scenario

### Description

Classe principale du modèle de simulation. Regroupe l’ensemble des aéroports, gère les événements, les clients, les fréquences de génération d’événements, et la progression temporelle de la simulation.

### 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 |

## Classe ScenarioMemento

### Description

Classe représentant un instantané (snapshot) de l’état d’un scénario à un moment précis. Sert à restaurer le scénario à cet état ultérieurement, selon le pattern Memento.

### 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 |

## Classe CaretakerScenario

### Description

Classe de gestion du mémento du scénario. Permet de sauvegarder l’état initial du scénario et de le restaurer à la demande, facilitant les retours en arrière ou la reprise de la simulation.

### 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 |

## Classe Evenement

### Descritpion

Classe décrivant un événement qui peut se produire dans la simulation (secours, incendie, transport…). Regroupe toutes les informations nécessaires à son traitement par le système.

### 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 |

## Classes d'État (EtatAeronef et ses implémentations)

### Description

Interface ou classe abstraite définissant les opérations possibles pour les différents états d’un aéronef (vol, sol, entretien, embarquement, débarquement, mission). Sert de base au pattern État.

### 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.

## Classe Simulateur

### Description

Classe centrale qui gère le déroulement et la logique de la simulation : progression temporelle, déclenchement et traitement des événements, et gestion des objets principaux.

### 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 |

## Classe FacadeSimulateur

### Description

Classe façade qui centralise et simplifie l’accès aux fonctionnalités principales du simulateur pour l’interface utilisateur et les autres composants.

### 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 |

## Classe ControleurSimulateur

### Description

Classe contrôleur qui orchestre la simulation, relaie les actions entre l’interface et le modèle, et gère la logique métier.

### Méthodes

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

## Classe form\_Simulateur

### Description

Classe représentant la fenêtre graphique principale de la simulation, assurant l’affichage, l’interaction utilisateur et la mise à jour de la vue en fonction des événements.

### 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) |

## Classes Utilitaires

### FrequenceEvenement

**Description**

Classe définissant la fréquence d’apparition des différents types d’événements dans le scénario.

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

### Position

**Description**

Classe représentant une position géographique par latitude et longitude. Fournit aussi des méthodes utilitaires pour la gestion spatiale.

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

### GestionnaireFichierXML

**Description**

Classe utilitaire permettant d’importer et d’exporter les scénarios sous forme de fichiers XML.

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

## Énumérations

### TypeAeronef

* Passager
* Cargo
* Secours
* Citerne
* Helicoptere

### TypeEtat

* Entretien
* Vol
* Sol
* Embarquement
* Debarquement

## Interfaces

### IObservateur

**Description**

Interface pour tout objet souhaitant être notifié lors de la survenue d’un événement dans le scénario.

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

### ISujet

**Description**

Interface pour toute classe pouvant être observée par un ou plusieurs observateurs et pouvant leur notifier les changements ou événements.

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

### IObservateurEvenement

**Description**

Interface dédiée à l’observation et au traitement des événements particuliers.

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

## Relation entre les classes

Aeronef **<|--** AeronefTransport **:** Héritage

Aeronef **<|--** AeronefUrgence **:** Héritage

AeronefTransport **<|--** AvionPassager **:** Héritage

AeronefTransport **<|--** AvionCargaison **:** Héritage

AeronefUrgence **<|--** AvionSecours **:** Héritage

AeronefUrgence **<|--** AvionCiterne **:** Héritage

AeronefUrgence **<|--** Helicoptere **:** Héritage

Client **<|--** ClientTransport **:** Héritage

Client **<|--** ClientEvenement **:** Héritage

ClientTransport **<|--** Passager **:** Héritage

ClientTransport **<|--** Cargo **:** Héritage

ClientEvenement **<|--** Secours **:** Héritage

ClientEvenement **<|--** Incendie **:** Héritage

ClientEvenement **<|--** Observation **:** Héritage

EtatAeronef **<|--** EtatVol **:** Héritage

EtatAeronef **<|--** EtatSol **:** Héritage

EtatAeronef **<|--** EtatEntretien **:** Héritage

EtatAeronef **<|--** EtatEmbarquement **:** Héritage

EtatAeronef **<|--** EtatDebarquement **:** Héritage

EtatAeronef **<|--** EtatEnMission **:** Héritage

IObservateur **<|..** ControleurSimulateur **:** Implémentation

IObservateur **<|..** form\_Simulateur **:** Implémentation

ISujet **<|..** Scenario **:** Implémentation

IObservateurEvenement **<|..** ControleurSimulateur **:** Implémentation

Aeronef **\*--** Position **:** Composition(PositionActuelle)

Aeronef **\*--** Position **:** Composition(PositionDestination)

Aeronef **\*--** EtatAeronef **:** Composition(EtatActuel)

Aeronef **-->** TypeAeronef **:** Association(type)

Aeronef **-->** TypeEtat **:** Association(typeEtat)

Client **\*--** Position **:** Composition(position)

ClientTransport **\*--** Aeroport **:** Composition(Destination)

Aeroport **\*--** Position **:** Composition(Position)

Aeroport o**--** Aeronef **:** Agrégation(Aeronefs)

Aeroport o**--** Client **:** Agrégation(Clients)

Scenario o**--** Aeroport **:** Agrégation(m\_aeroport)

Scenario o**--** FrequenceEvenement **:** Agrégation(m\_frequence)

Scenario o**--** IObservateur **:** Agrégation(m\_observateurs)

Scenario o**--** Client **:** Agrégation(clientsEvenements)

Evenement **\*--** Position **:** Composition(position)

Evenement **-->** Aeroport **:** Association(Destination)

Evenement o**--** Aeroport **:** Agrégation(Aeroports)

Evenement **-->** TypeEvenement **:** Association(typeEvenement)

Simulateur **\*--** Scenario **:** Composition(scenario)

FacadeSimulateur **\*--** Simulateur **:** Composition(simulateur)

form\_Simulateur **\*--** ControleurSimulateur **:** Composition(controleur)

FrequenceEvenement **-->** TypeEvenement **:** Association(Type)

ScenarioMemento o**--** Aeroport **:** Agrégation(Aeroports)

CaretakerScenario **\*--** ScenarioMemento **:** Composition(\_mementoInitial)

FabriqueAeronef **..>** Aeronef **:** Dépendance(crée)

FabriqueClient **..>** Client **:** Dépendance(crée)

FabriqueClient **..>** Evenement **:** Dépendance(utilise)

Client **..>** Aeronef **:** Dépendance(traite)

Aeroport o**--** Aeronef **:** Agrégation

Aeroport o**--** Client **:** Agrégation

Scenario **..>** Evenement **:** Dépendance(traite)

Scenario **--o** Aeroport **:** Agrégation

Scenario **..>** Client **:** Dépendance(utilise)

GestionnaireFichierXML **..>** Scenario **:** Dépendance(imports/exports)

Scenario **..>** ScenarioMemento **:** Composition

CaretakerScenario o**--** Scenario **:** Agrégation

Scenario **..>** IObservateur **:** Dépendance(notifies)

ISujet **..>** IObservateur **:** Dépendance(notifies)

## Patrons GOF

### Singleton

* FabriqueAeronef
* FabriqueClient

Assure qu’il n’existe qu’une seule instance de la fabrique dans tout le système, pour centraliser la création d’aéronefs et de clients.

### Fabrique/Fabrique Abstraite

* FabriqueAeronef
* FabriqueClient

Permet de créer des objets dérivés (différents types d’aéronefs et de clients) sans connaître leur classe concrète lors de la création.

### Observateur

* Scenario
* IObservateur, IObservateurEvenement
* Vue (form\_Simulateur)

Permet aux différentes parties du programme (par exemple l’interface graphique) d’être notifiées automatiquement lorsqu’un événement se produit ou qu’un état change dans le scénario, sans couplage fort.

### État

* EtatAeronef (interface/abstraite)
* EtatVol, EtatSol, EtatEntretien, EtatEmbarquement, EtatDebarquement, EtatEnMission

Permet à un aéronef de changer dynamiquement de comportement selon son état courant (en vol, au sol, en entretien, etc.) sans multiplier les conditions dans le code. Le comportement de l’aéronef est délégué à un objet état spécifique.

### Mémento

* ScenarioMemento
* CaretakerScenario
* Scenario (Utilise)

Permet de sauvegarder un instantané de l’état du scénario (ex : avant le début de la simulation) et de restaurer cet état plus tard, sans exposer les détails internes de l’objet.

### Façade

* FacadeSimulateur

Fournit une interface simplifiée et unifiée pour accéder aux fonctionnalités principales du simulateur. Elle masque la complexité des opérations et interactions internes.

### Composite (Potentiellement)

* Client