Rapport sur jalon 1
Module CDAA– C++ & Qt
Licence Sciences et Technologies – 3ème Année
Groupe : BD

Nom : **BAH**
Prénom : **Saikou Oumar**
Groupe : **A-TP1**

Nom : **DIALLO**
Prénom : **Mamoudou**
Groupe : **A-TP1**

Création des classes :

Contact :
Cette classe représente un contact.
Elle possède les attributs **nom, prenom, entreprise, telephone, URIphoto, dateCreation, dateLastModif, mail listInteraction.**
Elle contient 9 **accesseurs**, 7 **mutateurs**, 2 **méthodes** et une surcharge d'**operateur** << pour l'affichage.
Explications méthodes :
addInteraction(const Interaction &) : nous permet d'ajouter des interactions dans le gestionnaire.
removeInteraction(const Interaction &) : connaissant son contenu, cette fonction nous permet de supprimer des interactions.

Email :
La classe Email permet de stocker l'adresse mail d'un contact et manipuler séparément ses différentes parties.
Elle possède les attributs : **identifiant**, **domaine** et **mc**, les accesseurs pour y accéder et les mutateurs pour les modifier
Explications méthodes :
fromString(const string &) : nous permet ici de récupérer une adresse mail sous forme de string et la fonction se charge de la décomposer, la vérifier puis ensuite retourner un message d'erreur s'il y'a une anomalie ou encore de retourner la bonne adresse mail si elle est correcte.
toString() : nous permet d'afficher(retourner) une adresse mail au format « identifiant@domaine.mc ».

Date :
La classe permet de gérer des dates stockées dans des structures tm notamment de récupérer facilement le jour, le mois, l'année et de l'afficher.
Elle possède un seul attribut : **d** de type **tm *.**
Elle contient 4 **accesseurs**, 1 **mutateurs**, 2 **méthodes** et 5 surcharge **d'operateurs de comparaison** == <= >= < > pour effectuer des comparaisons sur des dates.
Explications méthodes :
fromDate(cons tint day, const int month, const int year) : elle nous permet récupérer une structure tm contenant la date définie par le jour, le mois et l'année passés en paramètre.
toString() : nous permet d'afficher (retourner) une date dans une String au format « jj/mm/aaaa ».

Todo :
Cette classe permet de créer des taches dans l'application avec une description et une date.

Elle possède 3 attributs : **lienInteraction** de type **Interaction*** , **contenu** de type **string**, **date** de type **struct tm**.
Elle contient 3 **mutateurs**, 3 **accesseurs** pour accéder ou modifier chaque attributs et une surcharge d'**operateur** << pour l'affichage.

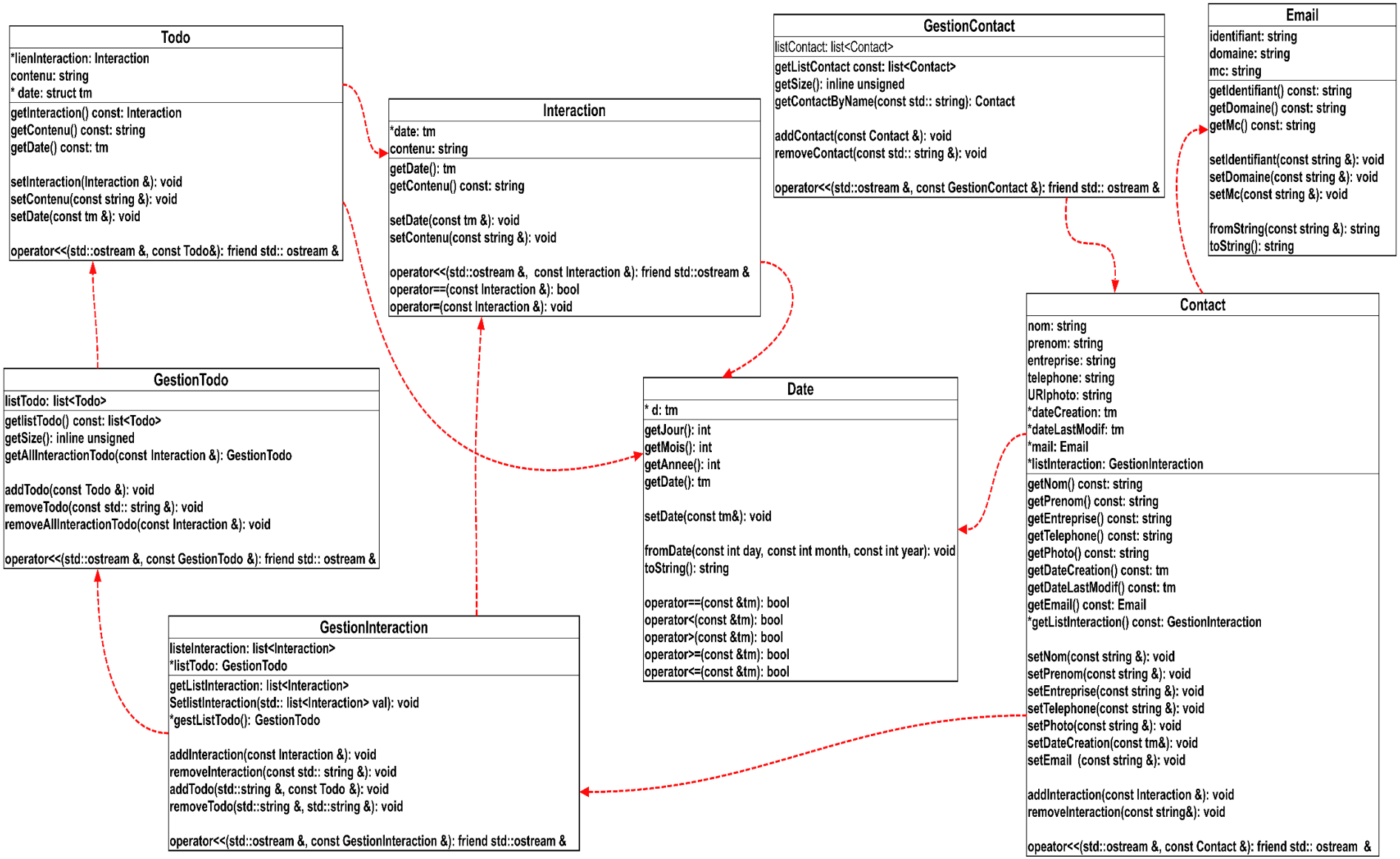
Interaction :
Cette classe décrit des interactions leurs contenus ainsi que leurs dates.
Elle possède 2 attributs : **date** de type **tm** et **contenu** de type **string**.
Elle contient 2 **accesseurs** pour la récupération des attributs, 2 **mutateurs** pour la modification et 3 surcharges d'opérateur << == = pour l'affichage, la comparaison et l'affection.

GestionContact :
Cette classe permet de gérer une liste de contact notamment l'ajout et la suppression et l'affichage de toutes les fiches de contact.
Elle possède un seul attribut : **listContact** de type **list<Contact>**.
Elle contient 3 **accesseurs**, 2 **méthodes** et une surchrage d'**operateur** << pour l'affichage.
Explication méthodes :
addContact(const Contact &): nous permet d'ajouter un nouveau contact dans la liste de contact.
removeContact(const string &nom): nous permet de supprimer un contact en passant en paramètre son nom.

GestionInteraction :
Cette classe permet de gérer une liste d'interaction ainsi que les tâches (Todo) qui lui sont liées. Elle nous permet aussi de gérer l'ajout d'interaction et de tâches, leurs suppressions et affichages.
Elle possède 2 attributs : **listeInteraction** de type **list<Interaction>** et **listTodo** de type **GestionTodo*.**
Elle contient 2 **accesseurs**, 1 **mutateurs**, 4 **méthodes** et une surcharge d'**operateur** << pour l'affichage.
Explications méthodes :
addInteraction(const Interaction &) : elle permet d'ajouter une interaction à la liste des interactions tout en triant celle-ci par ordre croissant des dates. Pour ça, elle parcourt toutes les interactions pour tenter de trouver une interaction dont la date est supérieure à l'interaction passée en paramètre. Si trouvé, elle est insérée avant celle-ci. Si non, elle est rajoutée à la fin de la liste.
removeInteraction(const string &) : nous permet de supprimer une interaction en passant en paramètre son contenu.
addTodo(string &, const Todo &) : nous permet d'ajouter une tâche au gestionnaire de tâche et de la relier à l'interaction dont le contenu est donné en paramètre.
removeTodo(string &, string &) : nous permet de supprimer une tache liée à une interaction

GestionTodo :
Cette classe permet de gérer une liste de Todo(tâche) notamment l'ajout, la suppression et l'affichage de tous les Todo.
Elle possède un seul attribut : **listTodo** de type **list<Todo>**.
Elle contient 3 **accesseurs**, 3 **méthodes** et une surcharge d'**operateur** << pour l'affichage.
Explications méthodes :
addTodo(const Todo &) : nous permet d'ajout une tâche à la liste des Todo.
removeTodo(const string &) : nous permet de supprimer une tache en passant en paramètre son contenu de type string.
removeAllInteractionTodo(const Interaction &) : nous permet de supprimer toutes les tâches(Todo) liées à une interaction. Pour cela on parcourt la liste de toutes les taches puis supprimer celles qui correspondent à l'interaction en question.

Lien entre classes :


Todo
*lienInteraction: Interaction
contenu: string
* date: struct tm
getInteraction() const: Interaction
getContenu() const: string
getDate() const: tm
setInteraction(Interaction &): void
setContenu(const string &): void
setDate(const tm &): void
operator<<(std::ostream &, const Todo&): friend std:: ostream &
Interaction
*date: tm
contenu: string
getDate(): tm
getContenu() const: string
setDate(const tm &): void
setContenu(const string &): void
operator<<(std::ostream &, const Interaction &): friend std::ostream &
operator==(const Interaction &): bool
operator=(const Interaction &): void
GestionContact
listContact: list<Contact>
getListContact const: list<Contact>
getSize(): inline unsigned
getContactByName(const std:: string): Contact
addContact(const Contact &): void
removeContact(const std:: string &): void
operator<<(std::ostream &, const GestionContact &): friend std:: ostream &
Email
identifiant: string
domaine: string
mc: string
getIdentifiant() const: string
getDomaine() const: string
getMc() const: string
setIdentifiant(const string &): void
setDomaine(const string &): void
setMc(const string &): void
fromString(const string &): string
toString(): string
GestionTodo
listTodo: list<Todo>
getlistTodo() const: list<Todo>
getSize(): inline unsigned
getAllInteractionTodo(const Interaction &): GestionTodo
addTodo(const Todo &): void
removeTodo(const std:: string &): void
removeAllInteractionTodo(const Interaction &): void
operator<<(std::ostream &, const GestionTodo &): friend std:: ostream &
Date
* d: tm
getJour(): int
getMois(): int
getAnnee(): int
getDate(): tm
setDate(const tm&): void
fromDate(const int day, const int month, const int year): void
toString(): string
operator==(const &tm): bool
operator<(const &tm): bool
operator>(const &tm): bool
operator>=(const &tm): bool
operator<=(const &tm): bool
Contact
nom: string
prenom: string
entreprise: string
telephone: string
URIphoto: string
*dateCreation: tm
*dateLastModif: tm
*mail: Email
*listInteraction: GestionInteraction
getNom() const: string
getPrenom() const: string
getEntreprise() const: string
getTelephone() const: string
getPhoto() const: string
getDateCreation() const: tm
getDateLastModif() const: tm
getEmail() const: Email
*getListInteraction() const: GestionInteraction
setNom(const string &): void
setPrenom(const string &): void
setEntreprise(const string &): void
setTelephone(const string &): void
setPhoto(const string &): void
setDateCreation(const tm&): void
setEmail (const string &): void
addInteraction(const Interaction &): void
removeInteraction(const string&): void
opeator<<(std::ostream &, const Contact &): friend std:: ostream &
GestionInteraction
listeInteraction: list<Interaction>
*listTodo: GestionTodo
getListInteraction: list<Interaction>
SetlistInteraction(std:: list<Interaction> val): void
*gestListTodo(): GestionTodo
addInteraction(const Interaction &): void
removeInteraction(const std:: string &): void
addTodo(std::string &, const Todo &): void
removeTodo(std::string &, std::string &): void
operator<<(std::ostream &, const GestionInteraction &): friend std::ostream &