Projet d'analyse Orientées Objets

Réalisé par

KALAI Mohamed Hedi

SATOURI Khaireddine

L3 Informatique

Année Universitaire : 2021-2022

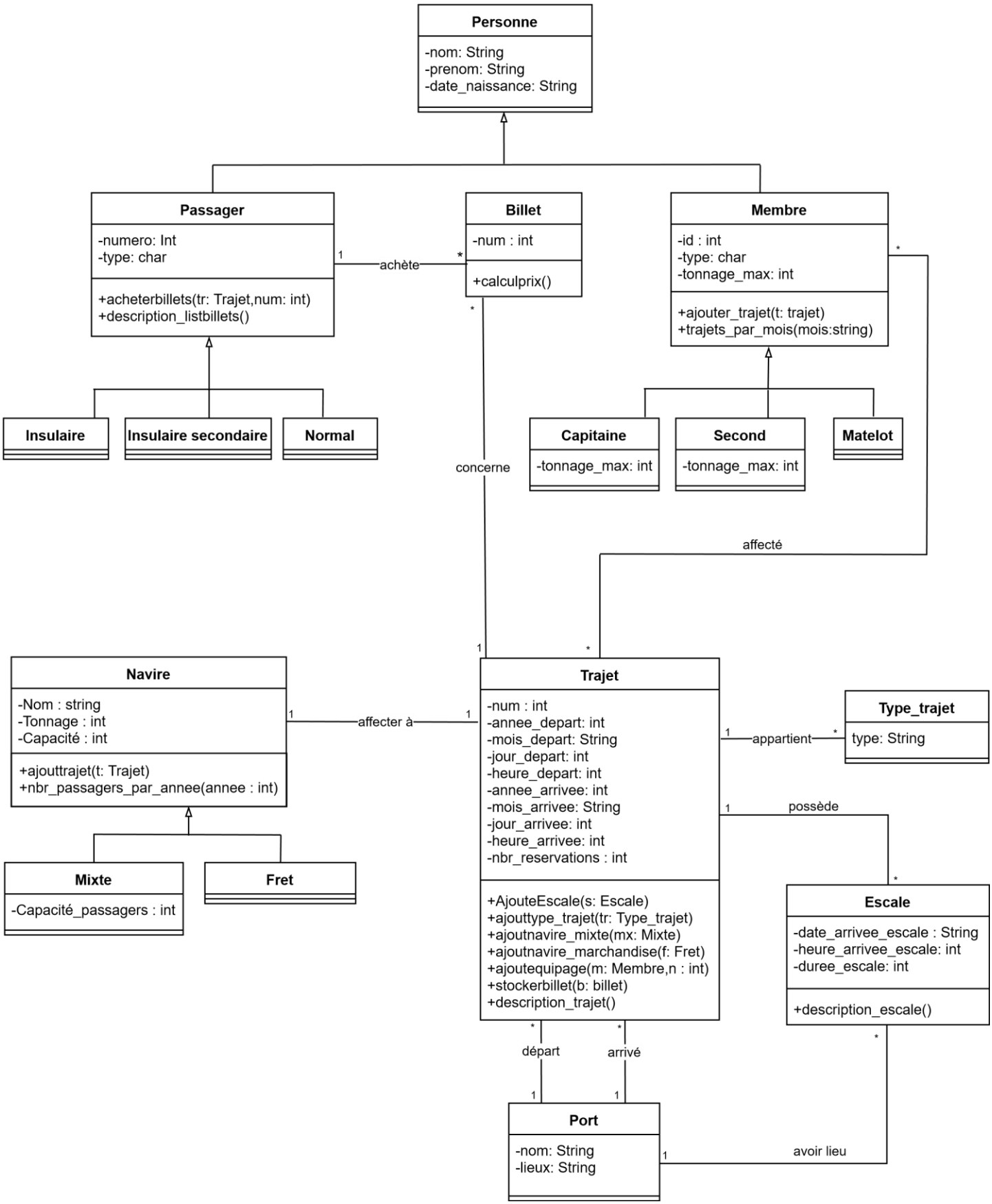
1.Introduction générale

Ce rapport résume toute la démarche de conception et implémentation de projet qui nous a permis de réaliser un logicielle permettant la **gestion simplifiée d'une compagnie maritime**.

Ce rapport contient l'ensemble des éléments du projet d'un point de vue théorique et les points les plus importants de la conception du programme.

2.Partie analyse

Après étude détaillé du cahier de charge pour le fonctionnement de système , on 'est arrivé à construire un **diagramme de classe UML** qui représente l'interaction entre les différents objets du système avec les différentes opérations qui les constitues.


Personne
-nom: String
-prenom: String
-date_naissance: String
Passager
-numero: Int
-type: char
+acheterbillets(tr: Trajet,num: int)
+description_listbillets()
Billet
-num : int
+calculprix()
Membre
-id : int
-type: char
-tonnage_max: int
+ajouter_trajet(t: trajet)
+trajets_par_mois(mois:string)
achète
1
*
Insulaire
Insulaire secondaire
Normal
Capitaine
-tonnage_max: int
Second
-tonnage_max: int
Matelot
concerne
affecté
Navire
-Nom : string
-Tonnage : int
-Capacité : int
+ajouttrajet(t: Trajet)
+nbr_passagers_par_annee(annee : int)
affecter à
Trajet
-num : int
-annee_depart: int
-mois_depart: String
-jour_depart: int
-heure_depart: int
-annee_arrivee: int
-mois_arrivee: String
-jour_arrivee: int
-heure_arrivee: int
-nbr_reservations : int
+AjouteEscale(s: Escale)
+ajouttype_trajet(tr: Type_trajet)
+ajoutnavire_mixte(mx: Mixte)
+ajoutnavire_marchandise(f: Fret)
+ajoutequipage(m: Membre,n : int)
+stockerbillet(b: billet)
+description_trajet()
Type_trajet
type: String
appartient
possède
Mixte
-Capacité_passagers : int
Fret
Escale
-date_arrivee_escale : String
-heure_arrivee_escale: int
-duree_escale: int
+description_escale()
départ
arrivé
Port
-nom: String
-lieux: String
avoir lieu

- Chaque personne est spécifiée par son nom, prénom et sa date de naissance,
  Une personne peut être un passager ou un membre :
- Chaque **membre** est spécifié par son identifiant et son type (‘c’ : capitaine, ‘s’ : second, ‘m’ :matelot),
  Un membre est affecté à un ou plusieurs trajets et pour chaque trajet on stocke les nombres d’heures de navigation et on appelle la fonction trajets_par_mois() pour afficher le nombre d’heures de navigation avec une liste triée de trajets réalisé pour un mois donnée,
  Le tonnage_max est une valeur d’accréditation qui ne peut être modifier que par les membres de type capitane et second.
- Chaque **passager** est spécifié par son numéro et son type (‘n’ : normal, ‘i’ : insulaire, ‘s’ :insulaire secondaire)
  Un passager peut acheter un ou plusieurs **billets** (acheterbillets())selon la disponibilité pour un telle trajet avec un prix spécial selon son type et pour chaque passager on appelle la fonction description_listbillets() pour afficher une liste triée des billets achetés
- Un **trajet** est spécifié par son date et port de départ , date et port d’arrivée, nombres de réservations (mis à jour à chaque nouvel achat d’un billet) et peut posséder un ou plusieurs escales,
  Chaque trajet possède un navire(ajoutnavire_mixte() ou ajoutenavire_marchandise()) inclut selon l’accord du capitaine et des seconds (si le navire est refusé le voyage n’est pas confirmé et les billets ne peuvent pas être acheté) et ajouter selon le type de trajet, des membres d’équipages (ajoutequipage()), et peut possédé des escales (AjouteEscale()),
  On appelle la fonction description_trajet() pour afficher les informations du trajet
- Chaque **navire** est spécifié par son nom, son tonnage de base et se capacité de base,
  Chaque navire ajouter à un trajet est sauvegarde ce trajet dans un conteneur par l’appelle de la fonction ajouttrajet()
  Les **navires mixtes** sont spécifiés par ses capacité_passagers,
  Pour chaque navire mixte en sauvegarde le nombres de passagers pour les voyages passés par ce navire tout au long de l’année
  La fonction nbr_passagers_par_annee() n’a de sens que pour les navires de type mixte (les fret ont une valeur 0)

3.Partie pratique

Pour faire fonctionner notre application on doit implémenter chacune de nos classes, leurs attributs et leurs méthodes.
On va s'intéressé à certaines parties de codes les plus importants pour illustrer notre choix d'implémentation.

Classe trajet :

Cette figure montre les différents attributs de la classe trajet :

```
trajet::trajet(int n,port *dep,int ansdep,string mdep,int jdep,int hdep,port *arr,int ansarr,string marr,int jarr,int harr,float prix){
num=n;

depart=dep;
annee_depart=ansdep;
mois_depart=mdep;
jour_depart=jdep;
heure_depart=hdep;

arrive=arr;
annee_arrivee=ansarr;
mois_arrivee=marr;
jour_arrivee=jarr;
heure_arrivee=harr;

prix_base=prix;
nbr_reservations=0;
}
```

La séparation des parties d'une date (année, mois, jour et heure) est nécessaire pour construire la fonction de comparaison et trié une liste selon les trajets comme le montre la figure ci-dessous pour triée la liste des billets selon leurs trajets dans la classe **bille.cpp**:

```cpp
bool comparaison(billet *b1, billet *b2){
    if(b1->gett()->getannee_depart()==b2->gett()->getannee_depart()){
        if(b1->gett()->getmois_depart()==b2->gett()->getmois_depart()){
            if(b1->gett()->getjour_depart()==b2->gett()->getjour_depart()){
                return(b1->gett()->getheure_depart()<b2->gett()->getheure_depart());
            }
            else return (b1->gett()->getjour_depart()<b2->gett()->getjour_depart());
        }
        else return (b1->gett()->getmois_depart()<b2->gett()->getmois_depart());
    }
    else return (b1->gett()->getannee_depart()<b2->gett()->getannee_depart());
}

//list triée (sur la date et l'heure) des trajets de billets acheter par un client
void Passager::description_listbillets(){
billets.sort(comparaison);
list<billet *>::iterator it1;
if(billets.empty()==true){ //si le client n'a acheté aucune billet
    cout<<"pas encore de billets acheter pour le client "<<this->getnom()<<" "<<this->getprenom()<<endl;
    cout<<"-------------------------------------------------"<<endl;
    cout<<"-------------------------------------------------"<<endl;
    }
else{
    //affichage de la liste triée des billets d'un client
cout<<"description detaille des billets acheter pour le client "<<this->getnom()<<" "<<this->getprenom()<<" :"<<endl;
for(it1=billets.begin();it1 !=billets.end();it1++){
cout<<"-------------------------------------------------"<<endl;
cout<<"numero de billet : "<<(*it1)->getnum()<<endl;
cout<<"prix : "<<(*it1)->calculprix()<<" euros"<<endl;
(*it1)->gett()->description_trajet();
cout<<"-------------------------------------------------"<<endl;
}
```

Et la liste des trajets effectués pour chaque membre dans la classe **membre.cpp**:

```cpp
bool comparaisont(trajet *t1, trajet *t2){
    if(t1->getannee_depart()==t2->getannee_depart()){
        if(t1->getmois_depart()==t2->getmois_depart()){
            if(t1->getjour_depart()==t2->getjour_depart()){
                return(t1->getheure_depart()<t2->getheure_depart());
            }
            else return (t1->getjour_depart()<t2->getjour_depart());
        }
        else return (t1->getmois_depart()<t2->getmois_depart());
    }
    else return (t1->getannee_depart()<t2->getannee_depart());
}

//Liste triée (sur la date et l'heure) des trajets effectués par un membre d'équipage pour un mois donné avec total des heures de navigation ;
 void membre::trajets_par_mois(string mois){
    int nbheures=0;
    trajets_effectues.sort(comparaisont);
list<trajet *>::iterator itt;
for(itt=trajets_effectues.begin(); itt!=trajets_effectues.end(); itt++){
    if((*itt)->getmois_depart() == mois){
        if((*itt)->getjour_depart()==(*itt)->getjour_arrivee()){
            //si la date d'arrivée et la date de dépard est dans le même jour
        nbheures+= (*itt)->getheure_arrivee()-(*itt)->getheure_depart();
        }
        else{
            //sinon
           nbheures+= (24-(*itt)->getheure_depart())+((*itt)->getheure_arrivee())+24*((*itt)->getjour_arrivee()-(*itt)->getjour_depart()-1);
        }

    }
}
```

Autre remarque importante dans la classe trajet concerne l'ajout d'un navire ;
Pour ajouter un navire à un trajet il faut prendre en considération l'accréditation du capitaine et des seconds sur le tonnage maximum pour un trajet donc un navire est ajouté que s'il respecte les conditions mentionnées :

```
void trajet::ajoutnavire_mixte(mixte *mx){
//verification de l'accréditation du capitaine et les seconds avant d'ajouter un navire
vector<membre *>::iterator it;
it=equipage.begin();


//ici on suppose que tous les membres de type capitaines et seconds se mettent d'accord sur la meme accréditation du tonnage maximal
//donc si l'itérateur pointe sur un capitaine ou un second il suffit de vérifier son accréditation son vérifier les autres
while(it!=equipage.end()){
    if((*it)->gettype()!='m'){
        if((*it)->gettype()=='c' || (*it)->gettype()=='s'){
            if(((*it)->gettonnage_max()) >= (mx->getcapacite())){
                mxt=mx;
                nav=mxt;
                break;
            }
            else {
                cout<<"tonnage de navire "<<mx->getnom()<<" réfusé !"<<endl;
                break;
            }
        }
    }
    else { //si c'est un matelot en passe au membre suivant de la liste
        it++;
    }
}
}
```

Classe Passager :

Pour acheter un billet il faut faire une instanciation de la classe billet et avant ca il faut vérifier qu'il y'a des billets disponibles :
On a ajouté un attribut dans la classe trajet présente le nombre de passagers pour le trajet concerné et initialisé à 0 et à chaque appelle de la fonction acheterbillet() cette valeur s'incrémente et puis on fait la comparaison entre le nombre de passager actuel et la capacité du navire

```
void Passager::acheterbillet(trajet *tr,int num){
    tr->setnbr_reservations(tr->getnbr_reservations()+1);
    //on vérifie qu'il y a encore des billets disponible pour le voyage
    if((tr->getnbr_reservations()) > ((tr->getmxt())->getcapacite_passagers())){
        cout<<"malheureusement! pas de places diponibles pour le voyage numéro "<<tr->getnum()<<",la billet numero "<<num<<" ne peut pas etre reserv
    }
    else{
    listtrajets.push_back(tr);
    billets.push_back(new billet(num,this,tr));
    (tr->getnav())->ajouttrajer(tr);
    }
}
```

La dernière ligne permet d'ajouter le trajet à la liste des trajets effectués pour un navire

Classe billet :

Une fonction importante concernant une billet et la fonction calculprix() qui permet de calculer le prix d'un billet selon le type de passager

```
float billet::calculprix(){
    if(this->getp()->gettype()=='n'){
        return (this->gett()->getprix_base());
    }
    else if(this->getp()->gettype()=='i'){
        return ((this->gett()->getprix_base())*0.25);
    }
    else if(this->getp()->gettype()=='s'){
        return ((this->gett()->getprix_base())*0.5);
    }
    return 0.0;

}
```

Cette fonction est appelée dans l'affichage de la liste des billets du passager :

```
//list triée (sur la date et l'heure) des trajets de billets acheter par un client
void Passager::description_listbillets(){
billets.sort(comparaison);
list<billet *>::iterator it1;
if(billets.empty()==true){ //si le client n'a acheté aucune billet
    cout<<"pas encore de billets acheter pour le client "<<this->getnom()<<" "<<this->getprenom()<<endl;
    cout<<"-------------------------------------------------"<<endl;
    cout<<"-------------------------------------------------"<<endl;
    }
else{
    //affichage de la liste triée des billets d'un client
cout<<"description detaille des billets acheter pour le client "<<this->getnom()<<" "<<this->getprenom()<<" :"<<endl;
for(it1=billets.begin();it1 !=billets.end();it1++){
cout<<"-------------------------------------------------"<<endl;
cout<<"numero de billet : "<<(*it1)->getnum()<<endl;
cout<<"prix : "<<(*it1)->calculprix()<<" euros"<<endl;
(*it1)->gett()->description_trajet();
cout<<"-------------------------------------------------"<<endl;
}
}
}
```

Classe membre :

Comme on a indiqué que la classe membre permet d'afficher la liste des trajets effectués par un membre pour mois donné et le nombres d'heures de navigation totale

Pour calculer cette valeur il faut ajouter chaque fois la différence entre la date d'arrivée et de départ d'un trajet :

```
//Liste triée (sur la date et l'heure) des trajets effectués par un membre d'équipage pour un mois donné avec total des heures de navigation ;
 void membre::trajets_par_mois(string mois){
    int nbheures=0;
    trajets_effectues.sort(comparaisont);
list<trajet *>::iterator itt;
for(itt=trajets_effectues.begin(); itt!=trajets_effectues.end(); itt++){
    if((*itt)->getmois_depart() == mois){
        if((*itt)->getjour_depart()==(*itt)->getjour_arrivee()){
            //si la date d'arrivée et la date de dépard est dans le même jour
        nbheures+= (*itt)->getheure_arrivee()-(*itt)->getheure_depart();
        }
        else{
            //sinon
           nbheures+= (24-(*itt)->getheure_depart())+((*itt)->getheure_arrivee())+24*((*itt)->getjour_arrivee()-(*itt)->getjour_depart()-1);
        }

    }
}
```

La classe principale :

La classe principale est la classe de vérification de notre programme ; pour cela on a créé un ensemble d'objets et les affectés à un ensemble d'opération pour tester notre logicielle :

```
int main(void){

    capitaine C1("thomas","Bernard","18/06/1990",11,'c',1000);
    capitaine C2("patric","christophe","18/06/1990",13,'c',6000);
    second S1("théo","Durand","19/04/1969",22,'s',1000);
    matelot M1("alexander","Dubois","20/03/1997",55,'m',1000);
    matelot M2("adams","Dubois","20/03/1997",55,'m',1000);
    matelot M3("dali","charni","20/03/1997",55,'m',1000);

    port P1("carthage","tunis");
    port P2("mars","marseille");
    port P3("vens","venise");
    escale ES1(&P3,"05/12/2021",16,2);

    normal N1("khaireddine","satouri","18/05/2000",1200,'n');
    normal N2("Léo","Richard","18/05/2000",1200,'n');
    Insulaire I1("Gabriel","martin","20/06/2000",1201,'i');
    Insulaire_sec IS1("kalai","hedi","20/06/2000",1201,'s');

    trajet T1(123,&P1,2021,"decembre",5,12,&P2,2021,"decembre",7,21,50.);
    trajet T2(124,&P3,2021,"decembre",10,10,&P2,2021,"decembre",10,18,50.);
    trajet T3(125,&P3,2022,"janvier",10,10,&P2,2022,"janvier",10,18,50.);

    mixte NAV1("hannibal",5000,800,3);
    fret NAV2("ellisa",5000,200);
    fret NAV3("soft",10000,8000);

    T1.ajoutequipage(&C1,1);
    T1.ajoutequipage(&S1,10);
    T1.ajoutequipage(&M1,20);
    T1.ajoutequipage(&M3,21);
    T1.ajoutnavire_mixte(&NAV1);
    T1.AjouteEscale(&ES1);
```

```
    T2.ajoutequipage(&C1,1);
    T2.ajoutequipage(&S1,10);
    T2.ajoutequipage(&M1,20);
    T2.ajoutequipage(&M2,21);
    T1.ajoutequipage(&M3,22);
    T2.ajoutnavire_marchandise(&NAV2);

    T3.ajoutequipage(&C2,2);
    T3.ajoutnavire_marchandise(&NAV3);//ici la navire va etre refuse par le capitaine

    N1.acheterbillet(&T1,1);
    N2.acheterbillet(&T1,2);
    IS1.acheterbillet(&T1,3);
    I1.acheterbillet(&T1,4);
    N1.acheterbillet(&T1,5);

    N1.description_listbillets();
    N2.description_listbillets();
    IS1.description_listbillets();
    I1.description_listbillets();
    C1.trajets_par_mois("decembre");
    M1.trajets_par_mois("decembre");
    NAV1.nbr_passagers_par_annee(2021);

}
```

Commande de compilation :

```
g++ principal.cpp
.\a.exe
```