ENICARTHAGE
المدرسة الوطنية للمهندسين بقرطاج
Ecole Nationale d'Ingénieurs de Carthage

جامعة قرطاج
Université de Carthage

Programmation

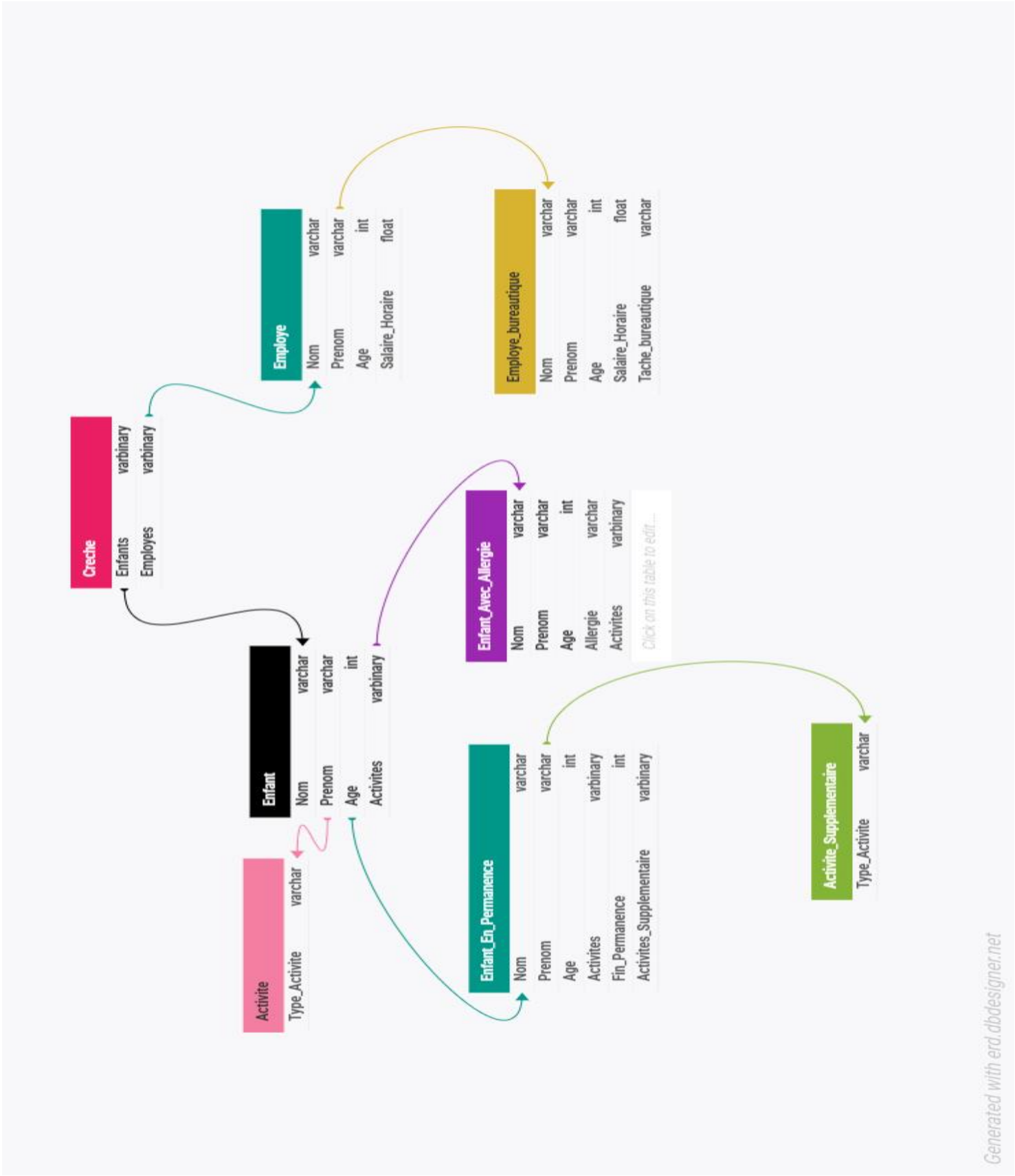
Oriente objet

Projet 4
Trinome 8

Encadre par :
Mrs. Wiem Yaiche

Presente par :
Selmi Wael
Ncir Alaeddine
Mhamdi Becem

Schema:


Creche
Enfants varbinary
Employes varbinary
Employe
Nom varchar
Prenom varchar
Age int
Salaire_Horaire float
Employe_bureautique
Nom varchar
Prenom varchar
Age int
Salaire_Horaire float
Tache_bureautique varchar
Enfant
Nom varchar
Prenom varchar
Age int
Activites varbinary
Activite
Type_Activite varchar
Enfant_Avec_Allergie
Nom varchar
Prenom varchar
Age int
Allergie varchar
Activites varbinary
Click on this table to edit...
Enfant_En_Permanence
Nom varchar
Prenom varchar
Age int
Activites varbinary
Fin_Permanence int
Activites_Supplementaire varbinary
Activite_Supplementaire
Type_Activite varchar
Generated with erd.dbdesigner.net

Code:

Main.cpp:

```
#include "Creche.cpp"
int main() {
    int ChoixEnfantOuEmployee;
    Creche c;
    do {
        cout << "Vous-voulez ajouter un Enfant ou un Employee (Enfant = 1 , Employee = 2 , Tapez 9 pour acceder au menu principal) : ";
        cin >> ChoixEnfantOuEmployee;
        if (ChoixEnfantOuEmployee == 1) {
            int ChoixEnfant;
            do {
                cout << "Vous-voulez ajouter un Enfant Normal, En Permenance ou avec une Allergie (1 = Normal, 2 = Permenance, 3 = Allergie) : ";
                cin >> ChoixEnfant;
                if (ChoixEnfant == 1) {
                    Enfant* e = new Enfant;
                    e->saisirEnfant();
                    c.AjouterEnfant(e);
                    ChoixEnfantOuEmployee = -1;
                } else if (ChoixEnfant == 2) {
                    EnfantPermanence* e = new EnfantPermanence();
                    e->saisirEnfant();
                    c.AjouterEnfant(e);
                    ChoixEnfantOuEmployee = -1;
                } else if (ChoixEnfant == 3) {
                    EnfantAvecAllergie* e = new EnfantAvecAllergie();
                    e->saisirEnfant();
                    c.AjouterEnfant(e);
                    ChoixEnfantOuEmployee = -1;
                }
            } while (ChoixEnfant != 1 && ChoixEnfant != 2 && ChoixEnfant != 3);
        } else if (ChoixEnfantOuEmployee == 2) {
            int ChoixEmployee;
            do {
                cout << "Vous-voulez ajouter un Employee Normal ou Bureautique (1 = Normal, 2 = Bureautique) : ";
                cin >> ChoixEmployee;
                if (ChoixEmployee == 1) {
                    Employee* e = new Employee();
                    e->saisirEmployee();
                    c.AjouterEmployee(e);
                    ChoixEnfantOuEmployee = -1;
                } else if (ChoixEmployee == 2) {
                    EmployeeBureautique* e = new EmployeeBureautique();
                    e->saisirEmployee();
                    c.AjouterEmployee(e);
                    ChoixEnfantOuEmployee = -1;
                }
```

```
        } while (ChoixEmployee != 1 && ChoixEmployee != 2);
    }
} while (ChoixEnfantOuEmployee != 9);
int choix;
cout << "=== BIENVENUE DANS LE SYSTEME DE GESTION DE CRECHE ===" << endl;
do {
    cout << "\n--- MENU PRINCIPAL ---" << endl;
    cout << "1. Ajouter" << endl;
    cout << "2. Supprimer" << endl;
    cout << "3. Afficher" << endl;
    cout << "4. Traitement" << endl;
    cout << "9. Quitter" << endl;
    cout << "Votre choix: ";
    cin >> choix;
    switch (choix) {
        case 1: {
            int ajoutChoix;
            cout << "\n--- MENU AJOUT ---" << endl;
            cout << "1. Ajouter un enfant normal" << endl;
            cout << "2. Ajouter un enfant en permanence" << endl;
            cout << "3. Ajouter un enfant avec allergie" << endl;
            cout << "4. Ajouter un employe normal" << endl;
            cout << "5. Ajouter un employe bureautique" << endl;
            cout << "Choix: ";
            cin >> ajoutChoix;
            switch (ajoutChoix){
                case 1:{
                    Enfant* en = new Enfant;
                    en->saisirEnfant();
                    c.AjouterEnfant(en);
                    break;}
                case 2 :{
                    EnfantPermanence* ep = new EnfantPermanence();
                    ep->saisirEnfant();
                    c.AjouterEnfant(ep);
                    break;}
                case 3:{
                    EnfantAvecAllergie* ea = new EnfantAvecAllergie();
                    ea->saisirEnfant();
                    c.AjouterEnfant(ea);
                    break;}
                case 4 :{
                    Employee* em = new Employee();
                    em->saisirEmployee();
                    c.AjouterEmployee(em);
                    break;}
                case 5:{
                    EmployeeBureautique* eb = new EmployeeBureautique();
                    eb->saisirEmployee();
                    c.AjouterEmployee(eb);
                    break;}
                default:
                    cout<<"Tapez un choix correct !!"<<endl;
```

```
            break;
        }
        break;
    }
    case 2: {
        int suppChoix;
        cout << "\n--- MENU SUPPRESSION ---" << endl;
        cout << "1. Supprimer un enfant" << endl;
        cout << "2. Supprimer un employe" << endl;
        cout << "Choix: ";
        cin >> suppChoix;
        switch (suppChoix)
        {
        case 1:{
            string nom;
            cout<<"Tapez le nom de l'enfant que vous voulez supprimez : ";
            cin>>nom;
            c.supprimerEnfant(nom);
            break;}
        case 2:{
            string nom;
            cout<<"Tapez le nom de l'employee que vous voulez supprimez : ";
            cin>>nom;
            c.supprimerEmployee(nom);
            break;}
        default:
            cout<<"Tapez un choix correct !!"<<endl;
            break;
        }
        break;
    }
    case 3: {
        int affChoix;
        cout << "\n--- MENU AFFICHAGE ---" << endl;
        cout << "1. Enfants normaux" << endl;
        cout << "2. Enfants en permanence" << endl;
        cout << "3. Enfants avec allergies" << endl;
        cout << "4. Employes normaux" << endl;
        cout << "5. Employes bureautiques" << endl;
        cout << "Choix: ";
        cin >> affChoix;
        switch (affChoix)
        {
        case 1:
            c.afficherEnfantsNormaux();
            break;
        case 2:
            c.afficherEnfantsPermanence();
            break;
        case 3:
            c.afficherEnfantsAllergies();
            break;
        case 4:
```

```
            c.afficherEmployeesNormaux();
            break;
        case 5:
            c.afficherEmployeesBureautiques();
            break;
        default:
            cout<<"Tapez un choix correct !!"<<endl;
            break;
        }
        break;
    }
    case 4: {
        int trtChoix;
        cout << "\n--- MENU TRAITEMENT ---" << endl;
        cout << "1. Lister enfants avec allergie" << endl;
        cout << "2. Afficher le plus vieux enfant" << endl;
        cout << "3. Enfants de moins de 10 ans" << endl;
        cout << "4. Compter enfants en permanence" << endl;
        cout << "5. Afficher le plus employee salarie" << endl;
        cout << "Choix: ";
        cin >> trtChoix;
        switch (trtChoix)
        {
        case 1:{
            string allergie;
            cout<<"Tapez une allergie : ";
            cin>>allergie;
            c.listerEnfantsAvecAllergie(allergie);
        }
            break;
        case 2:
            c.afficherPlusVieuxEnfant();
            break;
        case 3:
            c.listerEnfantsMoinsDe10Ans();
            break;
        case 4:
            c.compterEnfantsPermanence();
            break;
        case 5:
            c.afficherEmployeePlusHautSalaire();
            break;
        default:
            break;
        }
        break;
    }
    case 9:
        cout << "Merci d'avoir utilise le systeme de gestion de creche." << endl;
        break;
    default:
        cout << "Choix invalide. Veuillez réessayer." << endl;
}
```

```
    } while (choix != 5);
    return 0;
}
```

Creche.h:

```
#include "EnfantPermenance.cpp"
#include "EmployeeBureautique.cpp"
class Creche {
public:
    vector<Employee*> employees;
    vector<Enfant*> enfants;
public:
    Creche();
    Creche(Creche&);
    ~Creche(void);
    void AjouterEmployee(Employee* );
    void AjouterEnfant(Enfant* );
    void afficherEmployeesNormaux() ;
    void afficherEmployeesBureautiques() ;
    void afficherEnfantsNormaux() ;
    void afficherEnfantsPermanence() ;
    void afficherEnfantsAllergies() ;
    void supprimerEmployee( string );
    void supprimerEnfant( string );
    void compterEnfantsPermanence() ;
    void listerEnfantsAvecAllergie(string ) ;
    void afficherPlusVieuxEnfant() ;
    void listerEnfantsMoinsDe10Ans() ;
    void afficherEmployeePlusHautSalaire();
    friend ostream& operator<<(ostream& ,  Creche& );
    friend istream& operator>>(istream& , Creche& );
};
```

Creche.cpp:

```
#include "Creche.h"
Creche::Creche() {
}
Creche::Creche(Creche& other){
    for (int i = 0; i<other.enfants.size(); i++){
        if (typeid(other.enfants[i])==typeid(Enfant)){
            Enfant* e = new Enfant(*other.enfants[i]);
            enfants.push_back(e);
        }
        else if (typeid(other.enfants[i])==typeid(EnfantPermanence)){
            EnfantPermanence* e = dynamic_cast<EnfantPermanence*>(other.enfants[i]);
            enfants.push_back(e);
        }
        else {
            EnfantAvecAllergie* e = dynamic_cast<EnfantAvecAllergie*>(other.enfants[i]);
            enfants.push_back(e);
```

```
        }
    }
    for (int i = 0 ; i<other.employees.size();i++){
        if (typeid(other.employees[i])==typeid(Employee)){
            Employee* e = new Employee(*other.employees[i]);
            employees.push_back(e);
        }
        else {
            EmployeeBureautique * e = dynamic_cast<EmployeeBureautique*>(other.employees[i]);
            employees.push_back(e);
        }
    }
}
Creche::~Creche() {
    for (int i =0;i<employees.size();i++) {
        delete employees[i];
    }
    for (int i =0;i<enfants.size();i++) {
        delete enfants[i];
    }
}
void Creche::AjouterEmployee(Employee* e) {
    employees.push_back(e);
}
void Creche::AjouterEnfant(Enfant* e) {
    enfants.push_back(e);
}
void Creche::afficherEmployeesNormaux() {
    int count = 0;
    cout << "\n=== Employés Normaux ===" << endl;

    for (int i = 0; i < employees.size(); i++) {
        if (typeid(*employees[i]) == typeid(Employee)) {
            count++;
            cout << "Employé no " << count << " : " << endl;
            employees[i]->afficher();
        }
    }
    if (count == 0) {
        cout << "Aucun employé normal dans la creche." << endl;
    } else {
        cout << "Total employés normaux: " << count << endl;
    }
}
void Creche::afficherEmployeesBureautiques() {
    int count = 0;
    cout << "\n=== Employés Bureautiques ===" << endl;
    for (int i = 0; i < employees.size(); i++) {
        if (typeid(*employees[i]) == typeid(EmployeeBureautique)) {
            count++;
            cout << "Employé no " << count << " : " << endl;
            employees[i]->afficher();
        }
```

```
    }
    if (count == 0) {
        cout << "Aucun employé bureautique dans la creche." << endl;
    } else {
        cout << "Total employés bureautiques: " << count << endl;
    }
}
void Creche::afficherEnfantsNormaux() {
    int count = 0;
    cout << "\n=== Enfants Normaux ===" << endl;
    for (int i = 0; i < enfants.size(); i++) {
        if (typeid(*enfants[i]) == typeid(Enfant)) {
            count++;
            cout << "Enfant no " << count << " : " << endl;
            enfants[i]->afficher();
        }
    }
    if (count == 0) {
        cout << "Aucun enfant normal dans la creche." << endl;
    } else {
        cout << "Total enfants normaux: " << count << endl;
    }
}
void Creche::afficherEnfantsPermanence() {
    int count = 0;
    cout << "\n=== Enfants en Permanence ===" << endl;

    for (int i = 0; i < enfants.size(); i++) {
        if (typeid(*enfants[i]) == typeid(EnfantPermanence)) {
            count++;
            cout << "Enfant no " << count << " : " << endl;
            enfants[i]->afficher();
        }
    }
    if (count == 0) {
        cout << "Aucun enfant en permanence dans la creche." << endl;
    } else {
        cout << "Total enfants en permanence: " << count << endl;
    }
}
void Creche::afficherEnfantsAllergies() {
    int count = 0;
    cout << "\n=== Enfants avec Allergies ===" << endl;
    for (int i = 0; i < enfants.size(); i++) {
        if (typeid(*enfants[i]) == typeid(EnfantAvecAllergie)) {
            count++;
            cout << "Enfant no "<<count<<" : "<< endl;
            enfants[i]->afficher();
        }
    }
    if (count == 0) {
        cout << "Aucun enfant avec allergie dans la creche." << endl;
    } else {
```

```
        cout << "Total enfants avec allergies: " << count << endl;
    }
}
void Creche::supprimerEmployee(string nom) {
    int i = 0;
    bool found = false;
    while (i < employees.size() && !found) {
        if (employees[i]->getNom() == nom) {
            found = true;
        } else {
            i++;
        }
    }
    if (found) {
        delete employees[i];
        for (int j = i; j < employees.size() - 1; j++) {
            employees[j] = employees[j+1];
        }
        employees.pop_back();
        cout << "Employee supprime avec succes" << endl;
    } else {
        cout << "Aucun employee avec ce nom existe!" << endl;
    }
}
void Creche::supprimerEnfant(string nom) {
    int i = 0;
    bool trouve = false;
    while (i < enfants.size() && !trouve) {
        if (enfants[i]->getNom() == nom) {
            trouve = true;
        } else {
            i++;
        }
    }
    if (trouve) {
        delete enfants[i];
        for (int j = i; j < enfants.size() - 1; j++) {
            enfants[j] = enfants[j+1];
        }
        enfants.pop_back();
        cout << "Enfant supprime avec succes" << endl;
    } else {
        cout << "Aucun enfant avec ce nom existe!" << endl;
    }
}
void Creche::compterEnfantsPermanence()  {
    int count = 0;
    for (int i = 0;i<enfants.size();i++) {
        if(typeid(enfants[i])== typeid(EnfantPermanence)){
            count++;
        }
    }
    cout<<"Il ya "<<count<<" enfant en permenance"<<endl;
```

```
}
void Creche::listerEnfantsAvecAllergie( string typeAllergie)  {
    bool found = false;
    cout << "\n=== ENFANTS AVEC ALLERGIE A " << typeAllergie << " ===" << endl;
    for (int i = 0; i < enfants.size(); ++i) {
        EnfantAvecAllergie* e = dynamic_cast<EnfantAvecAllergie*>(enfants[i]);
        if (e->getTypeAllergie() == typeAllergie) {
            enfants[i]->afficher();
            cout << "Type d'allergie: " << e->getTypeAllergie() << endl;
            cout << "----------------------" << endl;
            found = true;
        }
    }
    if (!found) {
        cout << "Aucun enfant avec cette allergie trouve." << endl;
    }
}
void Creche::afficherPlusVieuxEnfant() {
    if (enfants.empty()) {
        cout << "Aucun enfant dans la creche." << endl;
        return;
    }

    Enfant* plusVieux = enfants[0];
    for (int i = 1; i < enfants.size(); ++i) {
        if (*plusVieux < *enfants[i]) {
            plusVieux = enfants[i];
        }
    }

    cout << "\n=== ENFANT LE PLUS AGE ===" << endl;
    plusVieux->afficher();
}
void Creche::listerEnfantsMoinsDe10Ans()  {
    int ageLimit = 10;
    bool trouve = false;
    cout << "\n=== ENFANTS DE MOINS DE " << ageLimit << " ANS ===" << endl;
    for (int i = 0; i < enfants.size(); ++i) {
        if (enfants[i]->getAge() < ageLimit) {
            enfants[i]->afficher();
            cout << "----------------------" << endl;
            trouve = true;
        }
    }
    if (!trouve) {
        cout << "Aucun enfant de moins de " << ageLimit << " ans trouve." << endl;
    }
}
void Creche::afficherEmployeePlusHautSalaire() {
    Employee* plusHautSalaire = employees[0];
    for (int i = 1; i < employees.size(); ++i) {
        if (employees[i]->getSalaireHoraire() > plusHautSalaire->getSalaireHoraire()) {
            plusHautSalaire = employees[i];
```

```
        }
    }
    cout << "\n=== EMPLOYE AVEC LE SALAIRE LE PLUS ELEVE ===" << endl;
    plusHautSalaire->afficher();
    cout << "Salaire: " << plusHautSalaire->getSalaireHoraire() << endl;
}
ostream& operator<<(ostream& out,Creche& creche) {
    for (int i = 0;i<creche.employees.size();i++) {
        if (typeid(creche.employees[i]) == typeid(Employee)) {
            out<<creche.employees[i];
        } else if (typeid(*creche.employees[i]) == typeid(EmployeeBureautique)) {
            out<<creche.employees[i];
        }
    }
    for (int i = 0;i<creche.enfants.size();i++) {
        if (typeid(creche.enfants[i]) == typeid(Enfant)) {
            out<<creche.enfants[i];
        } else if (typeid(creche.enfants[i]) == typeid(EnfantPermanence)) {
            out<<creche.enfants[i];
        } else if (typeid(creche.enfants[i]) == typeid(EnfantAvecAllergie)) {
            out<<creche.enfants[i];
        }
    }
    return out;
}
istream& operator>>(istream& in, Creche& creche) {
    int choixEnfantOuEmployee;
    do {
        cout << "Vous-voulez ajouter un Enfant ou un Employee (Enfant = 1, Employee = 2, Quitter = 9): ";
        in >> choixEnfantOuEmployee;
        if (choixEnfantOuEmployee == 1) {
            int choixEnfant;
            do {
                cout << "Type d'enfant (1 = Normal, 2 = Permanence, 3 = Allergie): ";
                in >> choixEnfant;
                if (choixEnfant == 1) {
                    Enfant* e = new Enfant();
                    in >> *e;
                    creche.AjouterEnfant(e);
                    choixEnfantOuEmployee = -1;
                }
                else if (choixEnfant == 2) {
                    EnfantPermanence* e = new EnfantPermanence();
                    in >> *e;
                    creche.AjouterEnfant(e);
                    choixEnfantOuEmployee = -1;
                }
                else if (choixEnfant == 3) {
                    EnfantAvecAllergie* e = new EnfantAvecAllergie();
                    in >> *e;
                    creche.AjouterEnfant(e);
                    choixEnfantOuEmployee = -1;
                }
```

```
            } while (choixEnfant != 1 && choixEnfant != 2 && choixEnfant != 3);
        }
        else if (choixEnfantOuEmployee == 2) {
            int choixEmployee;
            do {
                cout << "Type d'employé (1 = Normal, 2 = Bureautique): ";
                in >> choixEmployee;
                if (choixEmployee == 1) {
                    Employee* e = new Employee();
                    in >> *e;
                    creche.AjouterEmployee(e);
                    choixEnfantOuEmployee = -1;
                }
                else if (choixEmployee == 2) {
                    EmployeeBureautique* e = new EmployeeBureautique();
                    in >> *e;
                    creche.AjouterEmployee(e);
                    choixEnfantOuEmployee = -1;
                }
            } while (choixEmployee != 1 && choixEmployee != 2);
        }

    } while (choixEnfantOuEmployee != 9);
    return in;
}
```

Employee.h:

```
#include <iostream>
#include <string>
#include <vector>
#include <typeinfo>
using namespace std;
class Employee {
protected:
    string nom;
    string prenom;
    int age;
    double salaireHoraire;
public:
    Employee();
    Employee(Employee&);
    virtual ~Employee();
    virtual void afficher()  ;
    virtual void surveillerEnfant() ;
    virtual string getNom() ;
    virtual void setNom(string) ;
    virtual int getAge() ;
    virtual void setAge(int);
    virtual double getSalaireHoraire()  ;
    virtual void setSalaireHoraire(double) ;
    virtual void saisirEmployee();
    virtual double calculerSalaire(double) ;
```

```
    friend istream& operator>>(istream&,Employee&);
    friend ostream& operator<<(ostream&,Employee&);
    Employee operator+(const Employee&);
};
```

Employee.cpp:

```
#include "Employee.h"
Employee::Employee() : nom("Inconnu"),prenom("Inconnu") ,age(0), salaireHoraire(0.0) {
}
void Employee::afficher()  {
    cout << "Nom: " << nom << endl;
    cout << "Prenom: " << prenom << endl;
    cout << "Age: " << age << " ans" << endl;
    cout << "Salaire horaire: " << salaireHoraire << " €" << endl;
}
Employee::Employee(Employee& e){
    nom = e.nom;
    prenom = e.prenom;
    age = e.age;
    salaireHoraire = e.salaireHoraire;
}
string Employee::getNom()  {
    return nom;
}
int Employee::getAge()  {
    return age;
}
double Employee::getSalaireHoraire()  {
    return salaireHoraire;
}
void Employee::setNom(string n)  {
    nom = n;
}
void Employee::setAge(int a) {
    age = a;
}
void Employee::setSalaireHoraire(double salaire)  {
    salaireHoraire = salaire;
}
void Employee::saisirEmployee(){
    cout<<"Tapez Nom : ";
    cin>>nom;
    cout<<"Tapez prenom : ";
    cin>>prenom;
    cout<<"Tapez age : ";
    cin>> age;
    cout<<"Tapez SalaireHoraire : ";
    cin>> salaireHoraire;
}
double Employee::calculerSalaire(double heuresTravaillees)  {
    return salaireHoraire * heuresTravaillees;
```

```
}
Employee::~Employee(void) {
}
istream& operator>>(istream& in,Employee& e){
    cout<<"Tapez Nom : ";
    in>>e.nom;
    cout<<"Tapez prenom : ";
    in>>e.prenom;
    cout<<"Tapez age : ";
    in>> e.age;
    cout<<"Tapez SalaireHoraire : ";
    in>> e.salaireHoraire;
    return in;
}
ostream& operator<<(ostream& out,Employee& e){
    out<<"Nom : "<<e.nom<<endl;
    out<<"Prenom : "<<e.prenom<<endl;
    out<<"Age : "<<e.age<<endl;
    out<<"Salaire horaire : "<<e.salaireHoraire<<endl;
    return out;
}
void Employee::surveillerEnfant() {
}
```

EmployeeBureautique.h:

```
#include "Employee.cpp"
class EmployeeBureautique : public Employee {
    private:
        string tachebureautique;
    public:
        EmployeeBureautique();
        EmployeeBureautique(Employee&, string tache);
        EmployeeBureautique(EmployeeBureautique&);
        string getTache() ;
        void setTache(string) ;
        void afficher() ;
        void saisirEmployee();
        double calculerSalaire(double heuresTravaillees);
        friend istream& operator>>(istream&,EmployeeBureautique&);
        friend ostream& operator<<(ostream&,EmployeeBureautique&);
    };
```

EmployeeBureautique.cpp:

```
#include "EmployeeBureautique.h"
EmployeeBureautique::EmployeeBureautique(){
}
EmployeeBureautique::EmployeeBureautique(EmployeeBureautique& e ){
    nom = e.nom;
    prenom = e.prenom;
```

```
    age = e.age;
    salaireHoraire = e.salaireHoraire;
    tachebureautique = e.tachebureautique;
}
void EmployeeBureautique::afficher() {
    Employee::afficher();
    cout << "Tache bureautique: " << tachebureautique <<endl;
}
string EmployeeBureautique::getTache() {
    return tachebureautique;
}
void EmployeeBureautique::setTache(string t) {
    tachebureautique = t;
}
void EmployeeBureautique::saisirEmployee(){
    Employee::saisirEmployee();
    cout<<"Tapez tachebureautique : ";
    cin>>tachebureautique;
}
double EmployeeBureautique::calculerSalaire(double heuresTravaillees)  {
    const double Bonus = 1.52;
    return salaireHoraire * heuresTravaillees * Bonus;
}
istream& operator>>(istream& in,EmployeeBureautique& e){
    cout<<"Tapez Nom : ";
    in>>e.nom;
    cout<<"Tapez prenom : ";
    in>>e.prenom;
    cout<<"Tapez age : ";
    in>> e.age;
    cout<<"Tapez SalaireHoraire : ";
    in>> e.salaireHoraire;
    cout<<"Tapez Tache bureautique : ";
    return in;
}
ostream& operator<<(ostream& out,EmployeeBureautique& e){
    out<<"Nom : "<<e.nom<<endl;
    out<<"Prenom : "<<e.prenom<<endl;
    out<<"Age : "<<e.age<<endl;
    out<<"Salaire horaire : "<<e.salaireHoraire<<endl;
    out<<"Tache bureautique : "<<e.tachebureautique<<endl;
    return out;
}
```

Enfant.h:

```
#include "Activite.cpp"
class Enfant {
    protected:
        string nom;
        string prenom;
        int age;
```

```
        vector<Activite*> activites;
    public:
        Enfant();
        Enfant(const Enfant& );
        virtual ~Enfant();
        virtual void setNom(string);
        virtual void setPrenom(string);
        virtual void setAge(int);
        virtual string getNom();
        virtual string getPrenom();
        virtual int getAge();
        virtual void saisirEnfant();
        virtual void afficher() ;
        virtual void ajouterActivite(Activite* act);
        bool operator== (const Enfant&);
        bool operator<(const Enfant&);
        friend ostream& operator<<(ostream&, Enfant&);
        friend istream& operator>>(istream&, Enfant& );
    };
```

Enfant.cpp:

```
#include "Enfant.h"
Enfant::Enfant() : nom(""), prenom(""), age(0) {
}
Enfant::Enfant(const Enfant& other){
    nom = other.nom;
    prenom = other.prenom;
    age = other.age;
    for (int i = 0; i <other.activites.size();i++){
        Activite* a = new Activite(*other.activites[i]);
        this->activites.push_back(a);
    }
}
void Enfant::saisirEnfant(){
    cout <<"Tapez le nom du Enfant : ";
    cin >> nom;
    cout <<"Tapez le prenom du Enfant : ";
    cin >> prenom;
    try {
        cout <<"Tapez l'age du Enfant : ";
        cin >> age;
        if (age < 0) throw age;
    }
    catch (int invalidAge) {
        cout << "Erreur : l'age (" << invalidAge << ") doit etre > 0" << endl;
    }
    string choix;
    do{
        Activite* act = new Activite();
        string a;
        cout<< "Saisir activite : ";
```

```
        cin>> a;
        *act = a;
        ajouterActivite(act);
        cout << "Ajoutez activite?";
        cin >> choix;
    }while(choix == "oui");
}
Enfant::~Enfant() {
    for (int i = 0;i<activites.size();i++){
        delete activites[i];
    }
}
void Enfant::afficher()  {
    cout << "Nom: " << nom <<endl ;
    cout<< "Prenom: " << prenom <<endl  ;
    cout<< "age: " << age << endl;
    for (int i = 0;i<activites.size();i++) {
        Activite* a = new Activite(*activites[i]);
        a->afficher();
    }
}
void Enfant::ajouterActivite(Activite* act) {
    activites.push_back(act);
}
string Enfant::getNom(){
    return nom;
}
string Enfant::getPrenom(){
    return prenom;
}
int Enfant::getAge() {
    return age;
}
bool Enfant::operator== (const Enfant& e){
    return (nom == e.nom || prenom == e.prenom || age == e.age);
}
ostream& operator<<(ostream& out, Enfant& e){
    out<<"Nom : " <<e.nom<<endl;
    out<<"Prenom : " <<e.prenom<<endl;
    out<<"Age : " <<e.age<<endl;
    for(int i = 0;i<e.activites.size();i++){
        Activite* a = new Activite(*e.activites[i]);
        out<<"Activite : "<<a<<endl;
    }
    return out;
}
istream& operator>>(istream& in, Enfant& e){
    cout<<"Tapez Nom : ";
    in>>e.nom;
    cout<<"Tapez Prenom : ";
    in>>e.prenom;
    cout<<"Tapez Age : ";
    in>>e.age;
```

```
    string choix;
    do{
        Activite* act = new Activite();
        string a;
        cout<< "Saisir activite : ";
        in>> a;
        *act = a;
        e.ajouterActivite(act);
        cout << "Ajoutez activite?";
        cin >> choix;
    }while(choix == "oui");
    return in;
}
bool Enfant::operator< (const Enfant& e){
    return (age<e.age);
}

void Enfant::setNom(string nom) {
    this->nom = nom;
}

void Enfant::setPrenom(string prenom) {
    this->prenom = prenom;
}
```

EnfantPermenanace.h:

```
#include "EnfantAvecAllergie.cpp"
#include "Activitesupp.cpp"
class EnfantPermanence : public Enfant{
private:
    string finPermanence;
    vector<ActiviteSupplementaire*> activitessupplementaires;
public:
    EnfantPermanence();
    EnfantPermanence(EnfantPermanence&);
    ~EnfantPermanence();
    string getFinPermanence() ;
    void setFinPermanence(string );
    void ajouterActivite(ActiviteSupplementaire* act);
    void saisirEnfant();
    void afficher();
    friend istream& operator>> (istream&,EnfantPermanence&);
    friend ostream& operator<< (ostream&,EnfantPermanence&);
};
```

EnfantPermenance.cpp:

```
#include "EnfantPermenance.h"
EnfantPermanence::EnfantPermanence(){
}
EnfantPermanence::~EnfantPermanence() {
```

```
}
EnfantPermanence::EnfantPermanence(EnfantPermanence& other){
    nom = other.nom;
    prenom = other.prenom;
    age = other.age;
    for (int i = 0 ;i<other.activitessupplementaires.size();i++){
        ActiviteSupplementaire* a = new ActiviteSupplementaire(*other.activitessupplementaires[i]);
        activitessupplementaires.push_back(a);
    }
}
string EnfantPermanence::getFinPermanence()  {
    return finPermanence;
}
void EnfantPermanence::setFinPermanence(string fin) {
    finPermanence = fin;
}
void EnfantPermanence::ajouterActivite(ActiviteSupplementaire* act) {
    activitessupplementaires.push_back(act);
}
void EnfantPermanence::afficher(){
    Enfant::afficher();
    cout<< "Fin du permenance : "<< finPermanence<<endl;
    for (int i = 0; i<activitessupplementaires.size();i++) {
        ActiviteSupplementaire* a = new ActiviteSupplementaire(*activitessupplementaires[i]);
        a->afficher();
        cout<<""<<endl;
    }
}
void EnfantPermanence::saisirEnfant(){
    Enfant::saisirEnfant();
    cout<<"Tapez la fin du permenance : ";
    cin >> finPermanence;
    string choix;
    do{
        ActiviteSupplementaire* act = new ActiviteSupplementaire();
        string a;
        cout<< "Saisir activite supplementaire : ";
        cin>> a;
        *act = a;
        ajouterActivite(act);
        cout << "Ajoutez activite supplementaire ?";
        cin >> choix;
    }while(choix == "oui");
}
istream& operator>> (istream& in,EnfantPermanence& e){
    cout<<"Tapez la fin du permenance : ";
    in >> e.finPermanence;
    string choix;
    do{
        ActiviteSupplementaire* act = new ActiviteSupplementaire();
        string a;
        cout<< "Saisir activite : ";
        cin>> a;
```

```
        *act = a;
        e.ajouterActivite(act);
        cout << "Ajoutez activite?";
        cin >> choix;
    }while(choix == "oui");
    return in;
}
ostream& operator<< (ostream& out,EnfantPermanence& e){
    out<<"Nom : " <<e.nom<<endl;
    out<<"Prenom : " <<e.prenom<<endl;
    out<<"Age : " <<e.age<<endl;
    out<<"Fin du permenance : "<<e.finPermanence<<endl;
    for(int i = 0;i<e.activitessupplementaires.size();i++){
        ActiviteSupplementaire* a = new ActiviteSupplementaire(*e.activitessupplementaires[i]);
        out<<"Activite : "<<a<<endl;
    }
    return out;
}
```

EnfantAvecAllergie.h:

```
#include "Enfant.cpp"
class EnfantAvecAllergie : public Enfant {
    private:
        string allergie;
    public:
        EnfantAvecAllergie();
        EnfantAvecAllergie(string , string , int , string );
        EnfantAvecAllergie(EnfantAvecAllergie& );
        ~EnfantAvecAllergie();
        void setTypeAllergie(string);
        string getTypeAllergie() ;
        void saisirEnfant();
        void afficher();
        friend istream& operator>>(istream&, EnfantAvecAllergie&);
        friend ostream& operator<<(ostream&, EnfantAvecAllergie&);
    };
```

EnfantAvecAllergie.cpp:

```
#include "EnfantAvecAllergie.h"
EnfantAvecAllergie::EnfantAvecAllergie() : Enfant(), allergie("") {
}
EnfantAvecAllergie::EnfantAvecAllergie(string nom, string prenom, int age, string allergie){
    this->nom = nom;
    this->prenom = prenom;
    this->age = age;
    this->allergie = allergie;
}
EnfantAvecAllergie::EnfantAvecAllergie(EnfantAvecAllergie& other){
    nom = other.nom;
```

```
    prenom = other.prenom;
    age = other.age;
    allergie = other.allergie;
    for (int i = 0 ;i<other.activites.size();i++){
        Activite* a = new Activite(*other.activites[i]);
        activites.push_back(a);
    }
}
EnfantAvecAllergie::~EnfantAvecAllergie() {
    for (int i = 0;i<activites.size();i++){
        delete activites[i];
    }
}
void EnfantAvecAllergie::afficher() {
    Enfant::afficher();
    cout << "Allergie : " << allergie << endl;
}
void EnfantAvecAllergie::saisirEnfant(){
    Enfant::saisirEnfant();
    cout <<"Saisir allergie : ";
    cin >> allergie;
}
string EnfantAvecAllergie::getTypeAllergie(){
   return allergie ;
}
void EnfantAvecAllergie::setTypeAllergie(string t){
    allergie = t;
}
ostream& operator<< (ostream& out , EnfantAvecAllergie& e ){
    out<<"Nom : "<<e.nom<<endl;
    out<<"Prenom : "<<e.prenom<<endl;
    out<<"Age : "<<e.age<<endl;
    for (int i = 0 ;i<e.activites.size();i++){
        out<<"Activite "<<(i+1)<<" : "<<e.activites[i]<<endl;
    }
    out<<"Allergie : "<<e.allergie<<endl;
    return out;
}
istream& operator>>(istream& in, EnfantAvecAllergie& e ){
    cout<<"Tapez nom : ";
    in >>e.nom;
    cout<<"Tapez Prenom : ";
    in >>e.prenom;
    cout<<"Tapez Age : ";
    in >>e.age;
    string choix;
    do{
        Activite* act = new Activite();
        string a;
        cout<< "Saisir activite : ";
        in>> a;
        *act = a;
        e.ajouterActivite(act);
```

```
        cout << "Ajoutez activite?";
        cin >> choix;
    }while(choix == "oui");
    return in;
}
```

Activite.h:

```
#pragma once
#include <iostream>
#include <string>
#include <vector>
#include <typeinfo>
using namespace std;
class Activite {
    private:
        string typeActivite;
    public:
        Activite();
        Activite(string type);
        Activite(const Activite&);
        ~Activite();
        string getTypeActivite() ;
        void setTypeActivite(string);
        void afficher();
        friend istream& operator>>(istream&, Activite&);
        friend ostream& operator<<(ostream&, Activite&);
    };
```

Activite.cpp:

```
#include "Activite.h"
Activite::Activite() : typeActivite("Inconnu") {
}
Activite::Activite(string type) : typeActivite(type) {
}
Activite::Activite(const Activite& a ){
    typeActivite = a.typeActivite;
}
string Activite::getTypeActivite()  {
    return typeActivite;
}
void Activite::setTypeActivite(string type) {
    typeActivite = type;
}
void Activite::afficher(){
    cout<<"Type activite : "<<typeActivite;
}
istream& operator>>(istream& in, Activite& a){
    in >> a.typeActivite;
    return in;
}
```

```
ostream& operator<<(ostream& out, Activite& a){
    out<<"Type activite : " <<a.typeActivite<<endl;
    return out;
}
Activite::~Activite() {}
```

Activitesupp.h:

```
#include <iostream>
#include <string>
#include <vector>
#include <typeinfo>
using namespace std;
class ActiviteSupplementaire{
    private:
        string nomActivite;
    public:
        ActiviteSupplementaire();
        ActiviteSupplementaire(string nom);
        ActiviteSupplementaire(ActiviteSupplementaire&);
        ~ActiviteSupplementaire();
        void SetNomActivite(string);
        string GetNomActivite();
        void afficher();
        friend istream& operator>>(istream&, ActiviteSupplementaire&);
        friend ostream& operator<<(ostream&, ActiviteSupplementaire&);
    };
```

Activitesupp.cpp:

```
#include "Activitesupp.h"
ActiviteSupplementaire::ActiviteSupplementaire(): nomActivite("Inconnu") {
}
ActiviteSupplementaire::ActiviteSupplementaire(string nom) : nomActivite(nom){
}
ActiviteSupplementaire::ActiviteSupplementaire(ActiviteSupplementaire& other){
    nomActivite = other.nomActivite;
}
ActiviteSupplementaire::~ActiviteSupplementaire() {
}
void ActiviteSupplementaire::afficher()  {
    cout << "Activite Supplementaire : " << nomActivite << endl;
}
void ActiviteSupplementaire::SetNomActivite(string nom){
    nomActivite = nom;
}
string ActiviteSupplementaire::GetNomActivite(){
    return nomActivite;
}
istream& operator>>(istream& in, ActiviteSupplementaire& a){
    in >> a.nomActivite;
```

```
    return in;
}
ostream& operator<<(ostream& out, ActiviteSupplementaire& a){
    out << "Activite Supplementaire : " << a.nomActivite << endl;
    return out;
}
```

Execution:

Saisie d'un enfant normal:

```
Vous-voulez ajouter un Enfant ou un Employee (Enfant = 1 , Employee = 2 , Tapez 9 pour acceder au menu principal) : 1
Vous-voulez ajouter un Enfant Normal, En Permenance ou avec une Allergie (1 = Normal, 2 = Permenance, 3 = Allergie) : 1
Tapez le nom du Enfant : Mhamdi
Tapez le prenom du Enfant : Becem
Tapez l'age du Enfant : 5
Saisir activite : Padel
Ajoutez activite?oui
Saisir activite : basket
Ajoutez activite?non
```

Saisie d'un enfant en permanence:

```
Vous-voulez ajouter un Enfant ou un Employee (Enfant = 1 , Employee = 2 , Tapez 9 pour acceder au menu principal) : 1
Vous-voulez ajouter un Enfant Normal, En Permenance ou avec une Allergie (1 = Normal, 2 = Permenance, 3 = Allergie) : 2
Tapez le nom du Enfant : Ncir
Tapez le prenom du Enfant : Ala
Tapez l'age du Enfant : 6
Saisir activite : foot
Ajoutez activite?non
Tapez la fin du permenance : 17
Saisir activite supplementaire : echec
Ajoutez activite supplementaire ?non
```

Saisie d'un enfant avec allergie :

```
Vous-voulez ajouter un Enfant ou un Employee (Enfant = 1 , Employee = 2 , Tapez 9 pour acceder au menu principal) : 1
Vous-voulez ajouter un Enfant Normal, En Permenance ou avec une Allergie (1 = Normal, 2 = Permenance, 3 = Allergie) : 3
Tapez le nom du Enfant : Selmi
Tapez le prenom du Enfant : Wael
Tapez l'age du Enfant : 7
Saisir activite : ping-pong
Ajoutez activite?non
Saisir allergie : Chats
```

Saisie d'un employé :

```
Vous-voulez ajouter un Enfant ou un Employee (Enfant = 1 , Employee = 2 , Tapez 9 pour acceder au menu principal) : 2
Vous-voulez ajouter un Employee Normal ou Bureautique (1 = Normal, 2 = Bureautique) : 1
Tapez Nom : hechmi
Tapez prenom : adel
Tapez age : 35
Tapez SalaireHoraire : 6
```


Saisie d'un employé bureautique :

```
Vous-voulez ajouter un Enfant ou un Employee (Enfant = 1 , Employee = 2 , Tapez 9 pour acceder au menu principal) : 2
Vous-voulez ajouter un Employee Normal ou Bureautique (1 = Normal, 2 = Bureautique) : 2
Tapez Nom : trabelsi
Tapez prenom : tawfik
Tapez age : 22
Tapez SalaireHoraire : 10
Tapez tachebureautique : chef-equipe
```

Menu principale :

```
--- MENU PRINCIPAL ---
1. Ajouter
2. Supprimer
3. Afficher
4. Traitement
9. Quitter
Votre choix:
```

1. Ajouter :

```
--- MENU AJOUT ---
1. Ajouter un enfant normal
2. Ajouter un enfant en permanence
3. Ajouter un enfant avec allergie
4. Ajouter un employe normal
5. Ajouter un employe bureautique
Choix:
```

1.1. Ajouter un enfant normal :

```
Choix: 1
Tapez le nom du Enfant : semah
Tapez le prenom du Enfant : ourabi
Tapez l'age du Enfant : 3
Saisir activite : jeux-video
Ajoutez activite?non
```

1.2. Ajouter un enfant en permanence :

```
Choix: 2
Tapez le nom du Enfant : wehed
Tapez le prenom du Enfant : thnin
Tapez l'age du Enfant : 9
Saisir activite : football
Ajoutez activite?non
Tapez la fin du permenance : 19
Saisir activite supplementaire : ping
Ajoutez activite supplementaire ?non
```

1.3. Ajouter un enfant avec allergie :

```
Choix: 3
Tapez le nom du Enfant : mohammed
Tapez le prenom du Enfant : rebei
Tapez l'age du Enfant : 5
Saisir activite : handball
Ajoutez activite?non
Saisir allergie : chiens
```

1.4. Ajouter un employé normal :

```
Choix: 4
Tapez Nom : guesmi
Tapez prenom : abderrahmen
Tapez age : 36
Tapez SalaireHoraire : 5
```

1.5. Ajouter un employé bureautique :

```
Choix: 5
Tapez Nom : lassouad
Tapez prenom : mounir
Tapez age : 28
Tapez SalaireHoraire : 8
Tapez tachebureautique : organisateur
```

2. Supprimer :

```
--- MENU SUPPRESSION ---
1. Supprimer un enfant
2. Supprimer un employe
```

2.1. Supprimer un enfant :

```
Choix: 1
Tapez le nom de l'enfant que vous voulez supprimez : becem
Enfant supprime avec succes
```

2.2. Supprimer un employé :

```
Choix: 2
Tapez le nom de l'employee que vous voulez supprimez : hechmi
Employee supprime avec succes
```

3. Afficher :

```
--- MENU AFFICHAGE ---
1. Enfants normaux
2. Enfants en permanence
3. Enfants avec allergies
4. Employes normaux
5. Employes bureautiques
Choix:
```

3.1. Afficher les enfants normaux :

```
=== Enfants Normaux ===
Enfant no 1 :
Nom: mhamdi
Prenom: becem
age: 5
Type activite : padelType activite : basketEnfant no 2 :
Nom: semah
Prenom: ourabi
age: 3
Type activite : jeux-videoTotal enfants normaux: 2
```

3.2. Afficher les enfants en permanence :

```
=== Enfants en Permanence ===
Enfant no 1 :
Nom: ncir
Prenom: ala
age: 6
Type activite : footFin du permenance : 17
Activite Supplementaire : ping-pong

Enfant no 2 :
Nom: wehed
Prenom: thnin
age: 9
Type activite : footballFin du permenance : 19
Activite Supplementaire : ping
```

3.3. Afficher les enfants avec allergie :

```
=== Enfants avec Allergies ===
Enfant no 1 :
Nom: selmi
Prenom: wael
age: 7
Type activite : echecAllergie : chats
Enfant no 2 :
Nom: mohammed
Prenom: rebei
age: 5
Type activite : handballAllergie : chiens
Total enfants avec allergies: 2
```

3.4. Afficher les employés normaux :

```
=== Employ├®s Normaux ===
Employ├® no 1 :
Nom: hechmi
Prenom: adel
Age: 35 ans
Salaire horaire: 6 Ôé¼
Employ├® no 2 :
Nom: guesmi
Prenom: abderrahmen
Age: 36 ans
Salaire horaire: 5 Ôé¼
Total employ├®s normaux: 2
```

3.5. Afficher les employés bureautiques :

```
=== Employ├®s Bureautiques ===
Employ├® no 1 :
Nom: trabelsi
Prenom: tawfik
Age: 22 ans
Salaire horaire: 10 Ôé¼
Tache bureautique: chef-equipe
Employ├® no 2 :
Nom: lassouad
Prenom: mounir
Age: 28 ans
Salaire horaire: 8 Ôé¼
Tache bureautique: organisateur
Total employ├®s bureautiques: 2
```

4. Traitement :

```
--- MENU TRAITEMENT ---
1. Lister enfants avec allergie
2. Afficher le plus vieux enfant
3. Enfants de moins de 10 ans
4. Compter enfants en permanence
5. Afficher le plus employee salarie
Choix:
```

4.1. Lister enfants avec allergie :

```
Tapez une allergie : chat

=== ENFANTS AVEC ALLERGIE A chat ===
Nom: ncir
Prenom: ala
age: 5
Type activite : footballAllergie : chat
Type d'allergie: chat
```

4.2. Afficher le plus vieux enfant :

```
=== ENFANT LE PLUS AGE ===
Nom: mhamdi
Prenom: becem
age: 9
Type activite : basket
```

4.3. Enfant moins de 10 ans :

```
=== ENFANTS DE MOINS DE 10 ANS ===
Nom: ncir
Prenom: ala
age: 5
Type activite : footballAllergie : chat
----------------------
Nom: mhamdi
Prenom: becem
age: 9
Type activite : basket------------------
```

4.4. Compter enfants en permanence :

```
Il ya 2 enfant en permanence
```

4.5. Afficher le plus employé salarie :

```
=== EMPLOYE AVEC LE SALAIRE LE PLUS ELEVE ===
Nom: aberrahmen
Prenom: laajimi
Age: 22 ans
Salaire horaire: 8 Ôé¼
Tache bureautique: chef
Salaire: 8
```