Compte Rendu 2

Fichier .h :

```
#pragma once
#include <iostream>
#include <string>
#include <vector>
using namespace std;

class date {
	int jour, mois, annee;
public:
	date(int = 1, int = 1, int = 2020);
	~date();
	void afficherDate();
	void saisirDate();
	int getJour() { return jour; };
	int getMois() { return mois; };
	int getAnne() { return annee; };
	bool equal(date);
};


class compte{
protected:
	long RIB;
	string nom;
	double solde;
	date dateCreation;
public:
	compte(long,string,double,date);
	compte();
	~compte();
	void saisir();  // pour modifier un compte
	void deposer(double);
	bool retirer(double);  // si solde suffisant
	void afficher(string = "");
	long getRib() { return RIB; };
	string getNom() { return nom; };
	virtual double getSolde() { return solde; };
	date getDateCreation() { return dateCreation; };
};


class compteEpargne : public compte {
protected :
	float taux;
public:
	virtual double getSolde() { return solde * (1 - taux); };
	compteEpargne(long, string, double, date, float);
};

class compteEpargneEtranger : public compteEpargne {
protected :
	int pays;
	// 1 : France , 2 : Italie
public:
	compteEpargneEtranger(long, string, double, date,float,int);
	int getPays() { return pays; };
```

```
};

class banque{
	date dateCreation;
	string adresse;
	vector<compte*> tab;
public:
	// ecrire 2 constructeurs
	banque();
	banque(date, string);
	banque(const banque&);
	void remplir();
	int taille();
	void afficher();
	void ajouter(compte, int = 0);
	void ajouter(compte*, int = 0);
	void supprimer(int = 0);
	~banque(void);
	int chercher(compte);
	int chercher(compte*);
	int getNbComptesEpargne();
	int getNbComptesEtranger();
	string getNomPlusRiche();
	string getNomPlusPauvreEpargne();
	int getNbEpargneEn(int);
	int getNbCompte(int);
};
```

Fichier .cpp :

```
#include <iostream>
#include <string>
#include "Header.h"
#include <vector>
#include <typeinfo>
using namespace std;

date::date(int j, int m, int a) {
	jour = j;
	mois = m;
	annee = a;
}

date::~date() {

}

void date::saisirDate() {
	while (1) {
		cout << "Saisir le jour " << endl;
		cin >> jour;
		cout << "Saisir le mois " << endl;
		cin >> mois;
		cout << "Saisir le annee " << endl;
		cin >> annee;
		if (0 < jour < 32 && 0 < mois < 13) {
			break;
		}
		cout << "SAISIR UNE DATE VALIDE" << endl;
	}
}

bool date::equal(date d) {
```

```
	return (d.jour == jour && d.mois == mois && d.annee == annee);
}

void date::afficherDate() {
	cout << jour << "/" << mois << "/" << annee << endl;
}

compte::compte(long r,string n,double s,date d) {
	RIB = r;
	nom = n;
	solde = s;
	dateCreation = d;
}

compte::compte() {
	RIB = 0;
	nom = "";
	solde = 0;
	date d(11, 2, 2020);//Date d'aujourdhui
	dateCreation = d;
}

compte::~compte() {
}

void compte::saisir() {
	cout << "Saisr le RIB du compte " << endl;
	cin >> RIB;
	cout << "Saisr le nom du proprietaire du compte " << endl;
	cin >> nom;
	cout << "Saisr le solde du compte " << endl;
	cin >> solde;
	cout << "Saisr la date de creation du compte " << endl;
	date d;
	d.saisirDate();
	dateCreation = d;
}

void compte::deposer(double valeur) {
	solde += valeur;
}

bool compte::retirer(double valeur) {
	if (valeur > solde) {
		return false;
	}
	else {
		solde -= valeur;
		return true;
	}
}

void compte::afficher(string msg) {
	if (msg != "") cout << msg << endl;
	cout << "Adresse mémoire : " << this << endl;
	cout << "RIB : " << RIB << endl;
	cout << "Nom : " << nom << endl;
	cout << "Solde : " << solde << endl;
	cout << "Date de creation : ";
	dateCreation.afficherDate();
}


banque::banque() {
```

```
        cout << "Saisir la date de création de la banque " << endl;
        date d;
        d.saisirDate();
        dateCreation = d;
        cout << "Saisir l'adresse de la banque : " << endl;
        cin >> adresse;
}

banque::banque(date d, string adr) {
        dateCreation = d;
        adresse = adr;
}

void banque::remplir() {
        compte* q;
        char rep;
        cout << "Vous allez ajouter des comptes à la banque " << endl;
        while (1) {
                q = new compte();
                q->saisir();
                tab.push_back(q);
                cout << "Voulez vous ajouter un autre compte ? Y/N : " << endl;
                cin >> rep;
                if (rep == 'N') break;
        }
}

int banque::taille() {
        return tab.size();
}

void banque::afficher() {
        cout << "Adresse memoire Banque : " << this << endl;
        cout << "Informations relatives à la banque" << endl;
        cout << "Date de création de la banque : " << endl;
        dateCreation.afficherDate();
        cout << "Adresse : " << adresse << endl;
        cout << "Nombre de comptes : " << this->taille() << endl;;
        if (this->taille() > 0) {
                cout << "Informations relatives aux comptes" << endl;
                for (int i=0 ; i < this->taille(); i++) {
                        cout << "Compte " << i + 1 << endl;
                        tab[i]->afficher();
                }
        }
}

int banque::chercher(compte c) {
        for (int i = 0; i < this->taille(); i++) {
                if (c.getRib() == tab[i]->getRib()) return i;
        }
        return -1;//introuvable
}

int banque::chercher(compte* q) {
        for (int i = 0; i < this->taille(); i++) {
                if (q->getRib() == tab[i]->getRib()) return i;
        }
        return -1;
}

void banque::ajouter(compte c, int ind) {
        compte* q = new compte(c);
        this->tab.insert(tab.begin() + ind,q );
```

```
}

void banque::ajouter(compte *c, int ind) {
	this->tab.insert(tab.begin() + ind, c);
}

void banque::supprimer(int ind) {
	delete tab[ind];
	tab.erase(tab.begin() + ind);
}

banque::banque(const banque &b) {
	int n = b.tab.size();
	adresse = b.adresse;
	dateCreation = b.dateCreation;
	for (int i = 0; i < n; i++) {
		compte* q = new compte(*b.tab[i]);
		tab.push_back(q);
	}
}

banque::~banque() {
	while (tab.size() > 0) {
		this->supprimer();
	}
	tab.clear();
}

compteEpargne::compteEpargne(long r, string n, double s, date d,float t) {
	compte(r, n, s, d);
	taux = t;
}

compteEpargneEtranger::compteEpargneEtranger(long r, string n, double s, date d, float t, int p) {
	compteEpargne(r, n, s, d, t);
	pays = p;
}


int banque::getNbComptesEpargne() {
	int s = 0;
	for (int i = 0; i < tab.size(); i++) {
		if (typeid(tab[i]) == typeid(compteEpargne)) s += 1;
	}
	return s;
}

int banque::getNbComptesEtranger() {
	int s = 0;
	for (int i = 0; i < tab.size(); i++) {
		if (typeid(tab[i]) == typeid(compteEpargneEtranger)) s += 1;
	}
	return s;
}

string banque::getNomPlusRiche() {
	float sd=0;
	string nom = "NONE";
	for (int i = 0; i < tab.size(); i++) {
		if (tab[i]->getSolde() > sd) {
			sd = tab[i]->getSolde();
			nom = tab[i]->getNom();
		}
	}
```

```
	return nom;
}

string banque::getNomPlusPauvreEpargne() {
	float sd = 10000000;
	string nom = "NONE";
	for (int i = 0; i < tab.size(); i++) {
		if ((tab[i]->getSolde() < sd) && (typeid(tab[i])==typeid(compteEpargneEtranger)) ){
			sd = tab[i]->getSolde();
			nom = tab[i]->getNom();
		}
	}
	return nom;
}

int banque::getNbEpargneEn(int annee) {
	int s = 0;
	for (int i = 0; i < tab.size(); i++) {
		if ((typeid(tab[i]) == typeid(compteEpargne)) && (tab[i]->getDateCreation().getAnne() >
annee)) {
			s += 1;
		}
	}
	return s;
}

int banque::getNbCompte(int pays) {
	int s = 0;
	for (int i = 0; i < tab.size(); i++) {
		if (typeid(tab[i]) == typeid(compteEpargneEtranger)) {
			if (tab[i]->getPays() == pays) {
				s++;
			}
		}
	}
	return s;
}


/*
void main() {
	banque b1;
	b1.remplir();
	cout << "Banque B1" << endl;
	b1.afficher();
	banque b2(b1);
	cout << endl;
	cout << "*******************************" << endl;
	cout << "Banque B2" << endl;
	b2.afficher();
}
*/
```