Polymorphisme
- Infoperso

Complétez et modifiez le code de l'énoncé afin que lorsque la fonction affiche() est appelée, celle-ci affiche les informations de la classe de base + celle de la classe dérivée. Raturez le code qui n'est pas bon ou modifiez-le. Ajoutez le code manquant.

```
class InfoPerso
{
public:
//...
	void afficher();
private:
	string nom_;
	int age_;
};


class Etudiant : public InfoPerso
{
public:
	void afficher();
private:
	double moyenne_;
};

class Professeur : public InfoPerso
{
public:
	void afficher();
private:
	double salaire_;
};



int main()
{
	vector < InfoPerso > vec;
	Etudiant Champion("Eric",20,3.95);
	vec.push_back( Champion );
	Professeur Excellent("Paul",55, 123000.0);
	vec.push_back( Excellent );
	vec[1].afficher();
	vec[2].afficher();
}
```

```
void InfoPerso::afficher()
{
	cout << nom_ << " " << age_ << endl;
}

void Etudiant::afficher()
{
	/* */
}

void Professeur::afficher()
{
	/* */
}
```

Solution à la page suivante

```
class InfoPerso
{
public:
//...
	virtual void afficher();
private:
	string nom_;
	int age_;
};

class Etudiant : public InfoPerso
{
public:
	virtual void afficher();
private:
	double moyenne_;
};

class Professeur : public InfoPerso
{
public:
	virtual void afficher();
private:
	double salaire_;
};


int main()
{
	vector < InfoPerso* > vec;

	Etudiant* Champion = new Etudiant("Eric",20,3.95);
	vec.push_back(Champion);

	Professeur* Excellent= new Professeur("Paul",55, 123000.0);
	vec.push_back(Excellent);

	vec[0]->afficher();
	vec[1]->afficher();
}

void InfoPerso::afficher()
{
	cout << nom_ << " " << age_ << endl;
}

void Etudiant::afficher()
{
	InfoPerso::afficher();
	cout << moyenne_ << endl;
}

void Professeur::afficher()
{
	InfoPerso::afficher();
	cout << salaire_ << endl;
}
```