TP noté	
Classe	BTS SN 2ième année
Durée	4h

Héritage

Énoncé :

Une Société de Services en Informatique a pour projet le développement d'un logiciel permettant de gérer les salaires des **employés** d'une **entreprise**.
Cette entreprise possède deux grands types d'employés : des **commerciaux** et des **techniciens**.
Les commerciaux peuvent être soit des **vendeurs**, soit des **représentants**.

Tout **employé** possède un *nom*, un *prénom,* un *age* et un *salaire de base* et on pourra récupérer ses caractéristiques (*getnom, getprenom, getage, getsalaire*).

Les **commerciaux** possèdent *des primes.*
Le salaire de base pour les commerciaux est de 50 IRIS par mois.

Pour les **vendeurs** :
Une prime de 2 IRIS leur est attribuée chaque mois.
Le *calcul du salaire* se fait de la manière suivante :
salaire de base + prime

Pour les **représentants :**
Une prime de 5 IRIS leur est attribuée chaque mois.
Ils sont également dédommagés de leurs frais de déplacement, à la hauteur de 3 IRIS par déplacement.
Il faut donc pouvoir stocker le *nombre de déplacements* et pouvoir *ajouter des déplacements* chaque mois et *remettre à zéro* le nombre de déplacements.
Le *calcul du salaire* se fait de la manière suivante :
salaire de base + prime + 3*nbdeplacements

Le salaire de base d'un **technicien salarié** est de 40 IRIS par mois.

Toute **entreprise** possède un*e raison sociale (nom)*, des ***commerciaux (****10 au max****),*** des ***techniciens*** (10 au maximum).

Travail à réaliser :

Les mots en gras dans l'énoncé représentent des classes et les mots en italiques des méthodes ou des attributs. Pensez à utiliser le type std::string pour les chaines de caractères.

1- Déclarez et implémentez en C++ les différentes classes (n'oubliez pas le constructeur) (24 pts)

2- Créez une application permettant de gérer une entreprise dont la raison sociale est « SNEntreprise » et possède 10 employés (5 commerciaux et 5 techniciens). Concernant les commerciaux, il y a 2 vendeurs et 3 **représentants**). Le DRH entrera toutes les informations des employés. Il pourra également entrer le nombre de déplacements pour les représentants. Il aura la possibilité d'afficher tous les salaires des employés (le nombre de déplacements sera donc ainsi remis à zéro pour les représentants. (10 pts)

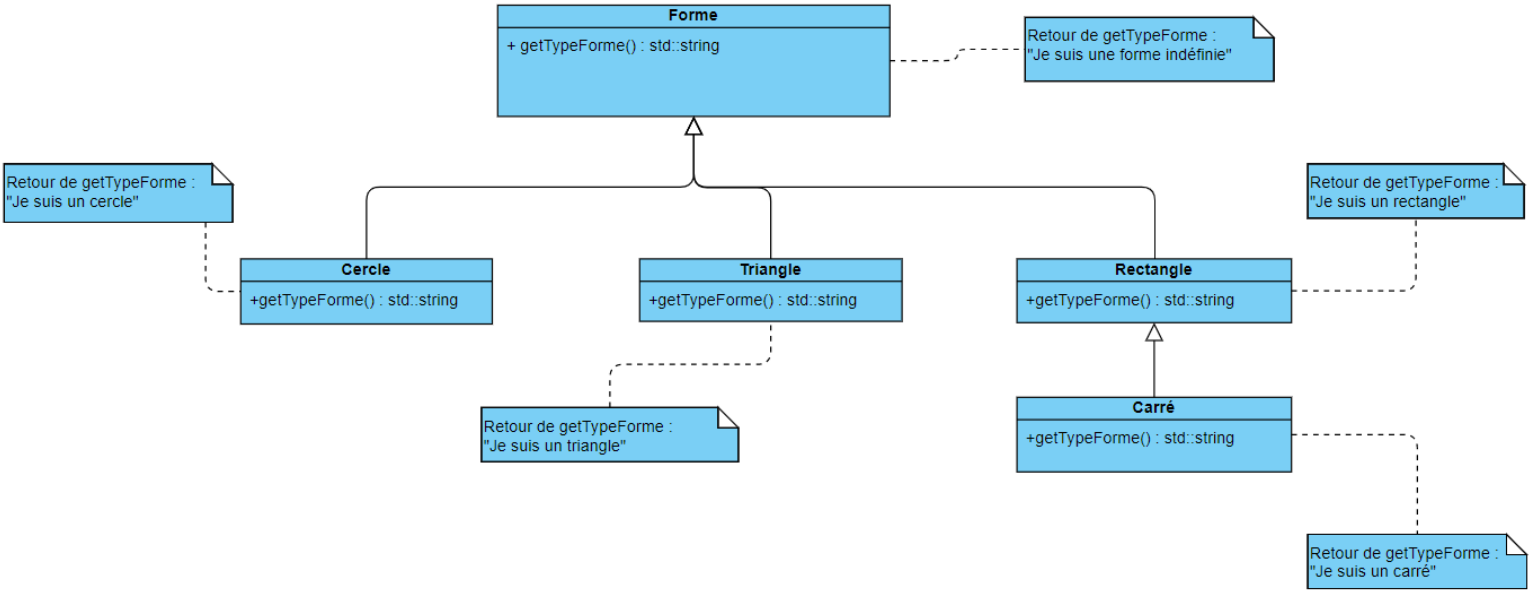
Divers :

- Commentaires (/1)
- Noms des variables (/1)
- Organisation projet (en fichiers) (/2)
- Projet complet (/2)

Annexe - C++ : Mise en place d'un héritage avec redéfinition de fonction

Impact de l'utilisation du mot clef virtual

Soit la hiérarchie de classe suivante :



Soit le programme suivant :

```
int main(int argc, char ** argv)
{
	Forme * f1 = new Forme();
	Forme * f2 = new Triangle();
	Forme * f3 = new Cercle();
	Cercle * f4 = new Cercle();

	cout << f1->getTypeForme() << endl;
	cout << f2->getTypeForme() << endl;
	cout << f3->getTypeForme() << endl;
	cout << f4->getTypeForme() << endl;

	return 0;
}
```

On propose 2 définitions des classes différentes ; une sans l'utilisation du mot clé virtual, l'autre avec. Voici le résultat de l'exécution de la fonction main avec ces deux proposition :

<table>
<tr><th>Sans virtual

-

Résolution statique effectuée à la compilation</th><th>En utilisant le mot clé virtual

-

Résolution dynamique effectuée à l'exécution</th></tr>
<tr><td>Définition de la classe forme :
class Forme
{
public:
 std::string getTypeForme() {
  return "Je suis une forme indéfinie";
 }
};

Définition de la classe cercle :
class Cercle : public Forme
{
public:
 std::string getTypeForme() {
  return "Je suis un cercle";
 }
};

Définition de la classe Triangle :
class Triangle : public Forme
{
public:
 std::string getTypeForme() {
  return "Je suis un triangle";
 }
};</td><td>Définition de la classe forme :
class Forme
{
public:
 virtual std::string getTypeForme() {
  return "Je suis une forme indéfinie";
 }
};

Définition de la classe cercle :
class Cercle : public Forme
{
public:
 virtual std::string getTypeForme() {
  return "Je suis un cercle";
 }
};

Définition de la classe Triangle :
class Triangle : public Forme
{
public:
 virtual std::string getTypeForme() {
  return "Je suis un triangle";
 }
};</td></tr>
<tr><td>Résultat de l'exécution :
Je suis une forme indéfinie
Je suis une forme indéfinie
Je suis une forme indéfinie
Je suis un cercle</td><td>Résultat de l'exécution :
Je suis une forme indéfinie
Je suis un triangle
Je suis un cercle
Je suis un cercle</td></tr>
</table>