Exercices sur les objets

1. Trouver les erreurs dans le programme suivant (l’absence des librairies n’est pas une erreur) :

class Temps{

private

int \_heure;

int \_minute;

int \_seconde;

public

temps(int heure=0, int min=0, int sec=0);

~temps(void);

int heure()const;

int minute()const;

int seconde()const;

void setTemps(int heure);

void affiche();

}

void temps::temps(int heure, int min, int sec)

{

assert(heure < 0 || min < 0 || sec < 0);

heure = \_heure;

min = \_minute;

sec = \_seconde;

}

void temps::~temps(void)

{

\_heure == \_minute == \_seconde == 0;

}

int temps::heure()

{

return \_heure;

}

int temps::minute()

{

return \_minute;

}

int temps::seconde()

{

return \_seconde;

}

void temps::setTemps(int heure)

{

\_heure = (heure >= 0 && heure < 24)? heure : 0;

}

void affiche()

{

cout << \_heure << ":" << \_minute << ":" << \_seconde

<< (\_heure < 12)? " AM" : " PM" << endl;

}

void main(){

temps T1, T2(5, 22), T3(T2);

T1.setTemps(16, 10, 3);

T1.affiche(cout);

cout << T2;

}

1. Trouver les bonnes associations des termes en orientée objet avec leur définition

(Objet, class, instanciation, héritage, polymorphisme, encapsulation, méthode, propriété, accesseur, constructeur, constructeur par défaut)

|  |  |
| --- | --- |
| **Définition** | **Terme** |
| Créer un objet à partir d’un autre objet |  |
| Action de déclarer une variable objet |  |
| Méthode ayant un nom identique à une autre, mais exécutée en fonction du nombre et du type des paramètres de l’appel |  |
| Parties privées de l’objet contenant les données |  |
| Méthode permettant de manipuler les propriétés privées |  |
| Action que l’objet peut effectuer sur les propriétés |  |
| Méthode appelée lors de l’instanciation d’un objet |  |
| Créer des nouveaux types de données près de la pensée humaine |  |
| Mot clé permettant de définir de nouveau objet |  |
| Cacher les propriétés avec le mot clé privé pour plus de sécurité |  |
| Constructeur sans paramètre |  |

1. Coder le constructeur de copie de la classe temps. Existait-il par défaut ? Faisait-il notre affaire?

1. Nommez 2 autres contextes où le copieur est appelé à votre insu et écrire un exemple de code pour appuyer votre explication. (autre que l’instantiation comme cedi temps T3(T2); )
2. Créer le méthode operator= de la classe temps. Existait-elle par défaut ? Faisait-elle notre affaire?
3. Créer les méthodes qui surchargent l’opérateur >, < et == de la classe temps
4. Créer les méthodes qui surchargent l’opérateur >> et << de la classe temps
5. Ajouter les références constantes à la classe ci-dessous pour protéger les paramètres.

class point

{

private:

double \_abscisse,

\_ordonnee;

public:

point();

point(double x,double y);

point(point p);

~point();

double abscisse();

double ordonnee();

void abscisse(double);

void ordonnee(double);

void affiche(ostream& sortie);

void lire(istream& entree);

point operator=(point p);

bool operator==(point p);

//la somme de 2 points retourne un point

point operator+(point p);

};

ostream& operator<<(ostream& sortie, point p);

1. Dans le programme suivant, identifier les paramètres **formels**, les paramètres **effectifs**, les paramètres **implicites** et les paramètres **explicites**.

point saisirPoint(const char\* message);

void main(){

point p1, p2(5, 22), p3(p2);

p1 = saisirPoint("Veuillez entrer un point ");

p1.setX(16);

p1.affiche(cout);

cout << p2;

}

point saisirPoint(const char\* message)

{

point p;

cout << message;

p.read(cin);

return p;

}