Université Paris Diderot
Master 1 II

Langages à Objets Avancés
Année 2012–2013

# Partiel

## Lundi 5 Novembre 2012

Tous documents autorisés (sauf copie du voisin, appareillage électronique, etc). Les téléphones portables, comme tout autre moyen de communication vers l'extérieur, doivent être éteints. Le temps à disposition est d'une heure. Motivez bien vos réponses.

## 1 Exercice [3 points] (statique *vs.* dynamique)

Étant donné une classe `K` et le code suivant :

```
class K{};
typedef K *PK;

void g(K **k) { delete *k; }

void f(K &k) {
  K *k1, k2(k), **k3;
  k1 = new K(k2);
  k3 = new PK;
  *k3 = new K(*k1);
  g(k3);
}

int main() {
  K k;
  f(k);
}
```

1. combien d'instances de `K` sont créées à l'exécution de ce programme ?
2. combien d'instances sont détruites ?
3. compléter la classe `K` de sorte que toutes les constructions et destructions apparaissent, *via* un affichage, à l'écran.

## 2 Exercice [3 points] (`const` est mon ami)

Soit l'extrait de programme suivant :

```
int i(0);
const int *p = &i;
cout << "i=" << i << endl;
cout << "*p=" << *p << endl;
i = 100;
cout << "i=" << i << endl;
cout << "*p=" << *p << endl;
```

1. quels sont les affichages produits par son exécution ?
2. faites-vous une différence entre `*p` et `i` ?
3. l'utilisation d'une référence `const int &r = i;` en lieu et place du pointeur `p` aurait-elle modifié le comportement ?

# 3 Exercice [2 points] (non-mutable)

Soit le code suivant :

```
class Etalon {
private:
  mutable int valeur;
public:
  Etalon(int valeur) { this->valeur = valeur; }
  int getValeur() const { valeur++; return valeur; }
};

int main() {
  const Etalon e(10);
  cout << e.getValeur() << endl;
  cout << e.getValeur() << endl;
  cout << e.getValeur() << endl;
  return 0;
}
```

1. quels sont les affichages produits à l'exécution de la fonction `main` ?
2. quelle différence faites-vous entre un objet constant et un objet non-mutable ?

# 4 Exercice [2 points] (polymorphisme)

Supposons l'existence de la fonction suivante :

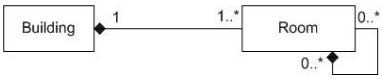
```
void supprimer(K k[],int n) {
  for (int i=0; i<n; i++) {
    delete k[i];
```

```
    k[i] = 0;
  }
}
```

1. quelles précautions doit-on prendre lors de l'écriture de la classe `K` ?
2. écrire une classe `K` minimale et la spécialiser en une sous-classe `L`, contenant toutes deux une(des) instruction(s) d'affichage permettant d'illustrer le bon fonctionnement de la fonction `supprimer`

# 5 Exercice [6 points] (UML)

Soit le diagramme UML suivant :




Écrire en C++ les classes qui correspondent au diagramme précédent avec les accesseurs des attributs et des relations ainsi que les constructeurs et destructeurs adéquats (on supposera que le nombre de pièces d'un immeuble est connu à sa construction).

# 6 Exercice [4 points] (factorisation UML/C++)

1. illustrer en UML la factorisation conceptuelle (interdit d'utiliser un exemple du cours)
2. illustrer en C++ la factorisation d'implémentation (interdit d'utiliser un exemple du cours)