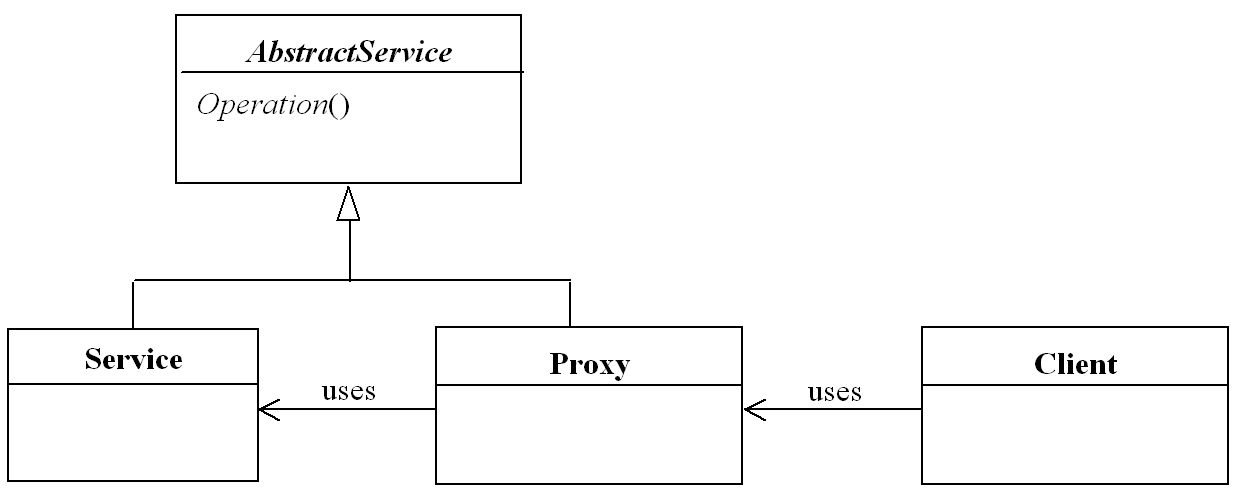
# AP4B - Examen final du 23 Juin 2022

## Durée 2h – aucun document autorisé

**Exercice 1 (2 pts).** **Questions diverses.**

1. Rappeler la ligne de commande pour respectivement compiler et exécuter un programme Java.
2. Quelle est la fonction du programme « ramasse miettes » de la machine virtuelle Java ?
3. Quel est l’effet de l’opérateur « = » en Java respectivement sur des valeurs et des objets ?
4. Quelle est la différence en Java entre une classe abstraite et une interface ?

**Exercice 3 (1,5 pts).** Considérons le schéma de conception (*design pattern*) Proxy.



1) Quel peut être l’intérêt de ce modèle dans une application ?

2) Compléter le programme ci-dessous selon le pattern Proxy. Compléter à l’endroit des pointillés (…).

/\*\* Représentation d’un espace de dessin \*/

public ... Canvas {

/\*\* Ajout d'un objet graphique \*/

public void addGraphicElement( GraphicElement ge );

}

/\*\* Proxy pour l'espace de dessin \*/

public class CanvasProxy ... Canvas {

private ... c;

public CanvasProxy( ... c ) {

this.c = c;

}

void codeSpecifique() {// ??? }

Public void addGraphicElement( ... ge ) {

// code ajouté par le proxy

codeSpecifique();

// acces au service de base

... ( ge );

}

}

**Exercice 2 (2 pts).** **Gestion de carrière professionnelle.**

Un logiciel de gestion de carrière doit mémoriser des informations pertinentes sur une carrière professionnelle. Modéliser par un diagramme de classe les deux situations ci-dessous, sachant que le modèle doit être adapté au besoin.

a) Une personne peut travailler pour plusieurs entreprises. Pour une entreprise, elle exerce une unique fonction (ou métier) à laquelle sont rattachées une rémunération horaire et une durée totale de travail.

b) Une personne peut travailler pour plusieurs entreprises et exercer en même temps plusieurs fonctions dans l’entreprise auxquelles sont rattachées une rémunération horaire et une durée totale de travail.

**Exercice 3 (10 pts).** **Etude de cas. Commissariat de police.**

Un commissariat de police fonctionne de la façon suivante. Le standard reçoit les appels téléphoniques de plaignants ou de témoins qu’il doit alors aiguiller vers la division (ou service) adéquate. Auparavant, il doit enregistrer sur ordinateur chaque appel (date, coordonnées du plaignant ou témoin, type de plainte ou témoignage). Les plaignants doivent ensuite venir vérifier puis signer leur plainte à l’accueil du commissariat. Les témoins ne sont convoqués pour compléments d’informations que si une plainte a été déposée.

Il existe trois divisions au commissariat, une pour chaque type de plainte :

• la division des délits mineurs (vols, violence, etc.),

• la division des trafics de stupéfiants,

• la division des crimes.

Le commissaire répartit les missions de la journée sur les ressources dont il dispose (inspecteurs, policiers, véhicules, chiens). Une mission traite un ensemble de délits qui ont eu lieu dans la même zone géographique. Il définit l’ordre des étapes puis affecte à la mission au moins un véhicule et deux policiers. Pour toute plainte/témoignage qui relève des deux dernières divisions, au moins un inspecteur est affecté à l’affaire. Les délits liés à un trafic de stupéfiants requièrent aussi l’attribution d’un ou deux chiens. Le secrétariat du commissariat doit alors enregistrer la mission et son heure de départ.

Les policiers partis en mission avec les bordereaux décrivant chaque plainte/témoignage doivent indiquer en temps réel l’état de leur mission (sur terminal portable), les arrêts et les départs aux différentes étapes ainsi que les incidents occasionnels qu’ils peuvent rencontrer (panne, retard). Si, au-delà d’une heure, le système n’enregistre aucun suivi de cette mission, il doit alerter le commissaire immédiatement. Celui-ci crée alors une mission de renfort dont la seule étape est de rejoindre la dernière étape rapportée par la mission qui s’est probablement déroulée de façon anormale.

Les inspecteurs de la division des crimes doivent mener leur enquête en étroite collaboration avec un juge du tribunal d’instance. Le service des archives du commissariat les aide dans leur enquête en leur fournissant toutes les informations qu’ils trouvent en relation avec leur affaire. Une fois l’enquête terminée, les policiers, munis d’un mandat d’arrêt du juge affecté à l’affaire, procèdent à l’arrestation du ou des coupables désignés. Chaque arrestation doit alors faire l’objet d’un rapport (date, coordonnées des coupables, lieu) que les policiers doivent enregistrer sur ordinateur.

1. Donner le diagramme des cas d’utilisation qui décrit le fonctionnement du commissariat de police.

2. Décrire la structure statique de ce système par un diagramme de classes.

3. Pour chaque cas d’utilisation important, décrire les principaux scénarios par des diagrammes de séquence. On considérera trois cas d’utilisation.

**Exercice 4 (4,5 pts). Trieur dichotomique en Java.**

Voici un algorithme de tri en ordre croissant d'une tranche de tableau comprise entre les éléments d'indices debut et fin :

trier(debut, fin) {   
  si (fin - debut < 2) { // on a fini, pas d'appel récursif  
    si (t[debut] > t[fin])   
      echanger(t[i], t[j])   
  }   
  sinon {   
    milieu = (i + j) / 2   
    trier(debut, milieu)   
    trier(milieu + 1, fin)   
    triFusion(milieu) // tri fusion des 2 moitiés de la tranche du tableau   
  }   
}

On remarque que les 2 tris qui sont effectués avant la fusion sont indépendants l'un de l'autre et il est donc facile de les faire exécuter en parallèle par 2 threads. La structure du programme étant donnée dans les deux pages suivantes, le but de l'exercice est de la compléter aux endroits indiqués par des pointillés "**..."** .

/\*\* Tri d'un tableau d'entiers multi-thread. \*/

**public** **class** Trieur **extends** **...** {

**private** **int**[] t; // tableau à trier

**private** **int** debut, fin; // tranche de ce tableau qu'il faut trier

**private** Trieur parent; // tread Trieur qui a lancé ce (this) Trieur

**private** **int** nbNotify = 0; // Nombre de notifys envoyés à ce (this) Trieur

**public** Trieur(**int**[] t) {

**this**(**null**, **...**, **...**, **...**);

}

**private** Trieur(Trieur parent, **int**[] t, **int** debut, **int** fin) {

**this**.parent = parent;

**this**.t = t;

**this**.debut = debut;

**this**.fin = fin;

start();

}

**public** **...** **void** notifier() {

**this**. **...**

**this**. **...**

}

**public** **void** run() {

**if** (fin - debut < 2) {

**...**

}

**else** {

**int** milieu = debut + (fin - debut) / 2;

Trieur trieur1 = **...**

Trieur trieur2 = **...**

// attend les 2 threads

**synchronized**(**this**) {

**try** {

// Tant que 2 notifications n'ont pas été reçues (1 par

// trieur "fils"), on attend.

**while** (**...**) {

**...**

}

}

**catch**(**...** e) {}

}

triFusion(**...**);

}

**if** (parent != **...**) {

parent. **...**

}

}

/\*\* Echanger t[i] et t[j] \*/

**private** **void** echanger(**int** i, **int** j) {

**int** valeur = t[i];

t[i] = t[j];

t[j] = valeur;

}

/\*\*

\* Fusionne 2 tranches déjà triées du tableau t.

\* - 1ère tranche : de debut à milieu = (debut + fin) / 2

\* - 2ème tranche : de milieu + 1 à fin

\*/

**private** **void** triFusion(**int** debut, **int** fin) {

// tableau où va aller la fusion

**int**[] tFusion = **new** **int**[fin - debut + 1];

**int** milieu = (debut + fin) / 2;

// Indices des éléments à comparer

**int** i1 = debut,

i2 = milieu + 1;

// indice de la prochaine case du tableau tFusion à remplir

**int** iFusion = 0;

**while** (i1 <= milieu && i2 <= fin) {

**if** (t[i1] < t[i2]) {

tFusion[iFusion++] = t[i1++];

}

**else** {

tFusion[iFusion++] = t[i2++];

}

}

**if** (i1 > milieu) {

// la 1ère tranche est épuisée

**for** (**int** i = i2; i <= fin; ) {

tFusion[iFusion++] = t[i++];

}

}

**else** {

// la 2ème tranche est épuisée

**for** (**int** i = i1; i <= milieu; ) {

tFusion[iFusion++] = t[i++];

}

}

// Copie tFusion dans t

**for** (**int** i = 0, j = debut; i <= fin - debut; ) {

t[j++] = tFusion[i++];

}

}

**public** **static** **void** main(String[] args) {

**int**[] t = {5, 8, 3, 2, 7, 10, 1, 6, 9, 4};

Trieur trieur = **new** Trieur(t);

**try** {

trieur.join();

}

**catch**(InterruptedException e) {}

**for** (**int** i = 0; i <t.length; i++) {

System.***out***.print(t[i] + " ; ");

}

System.***out***.println();

}

}