M2 Informatique - HMIN304 - "Réutilisation et Composants"

Année 2015-2016.

Christophe Dony, Chouki Tibermacine

Durée : 2h00. Documents non autorisés. La précision et la concision des réponses sont notées ainsi que la lisibilité des codes. Soyez précis et concis, vous gagnez du temps.

1 Réutilisation - Patterns (environ 5 points)

Considérons la hiérarchie de classes des éléments de stockage (Directory, File, Link), architecturée selon le schéma *Composite*, et dotée d'un mécanisme de visite, selon le schéma *Visiteur*, étudié dans le TD/TP no 1. Vous avez réalisé en TP plusieurs visiteurs dont `RazVisitor`. Soit (ci-dessous) le code de la classe `Visitor` et le code (incomplet) de la classe `Directory` dans lequel l'attribut `elements` sert à stocker les éléments contenus dans un *directory* (répertoire).

```
1 public abstract class Visitor {
2     public void visitFile(File f){}
3     public void visitDirectory(Directory f){}
4     public void visitLink(Link f){} }

6 public class Directory extends ElementStockage{
7     protected Collection<ElementStockage> elements ;
8     ... }
```

Questions

1. Expliquez en quoi *Composite* et *Visitor* sont des schémas en relation avec les structures de données arborescentes.
2. Pour les éléments de stockage, donnez le code de la méthode `getAbsoluteAddress`
3. Pour *visitor* appliqué aux élements de stockage, Justifiez le choix *protected* pour `elements` dans le code précédent. Ce choix implique pour le schéma "visiteur" que le parcours de la structure arborescente ne peut être programmé que dans la méthode `accept(Visitor v)` de la classe `Directory`. Donnez dans ce contexte le code de cette méthode.
4. Réaliser une classe `RazVisitor` qui implante une fonction qui remet à zéro la taille de tous les fichiers d'un répertoire d et récursivement de tous les fichiers des sous-répertoires de d.
5. Imaginez que vous n'avez pas mis en place le schéma *Visitor* sur les éléments de stockage. Pourriez-vous définir une méthode `raz()` sur la classe Directory qui ferait globalement la même chose que le RazVisitor précédent ? Si oui quel est l'intérêt de mettre en place le schéma *Visitor* ?

2 Composants JavaBeans et Aspects - Environ 6 points

Plaçons nous dans le contexte de réalisation d'une classe `JBPoint` définissant des points classiques mais compatibles avec l'environnement *JavaBeans*, qui pourront être disponible sur la palette, comme dans le TP NetBeans . Les `JBPoint` ("JavaBeans Points") seront des *beans* de type "écouté", dont les propriétés `X` et `Y` représentant l'abcisse et l'ordonnée doivent être des propriétés liées ("*Bound Properties*").

1. Donnez en Java le code des méthodes de la classe `JBPoint` relatives à la propriété `X`.
2. Expliquez comment un EJB écouteur peut s'abonner aux modifications de valeurs de la propriété `X`.
3. Donnez un code d'un adapteur permettant à un écouteur non équipé de s'abonner aux modifications de valeurs de la propriété `X`.
4. Définissez en **AspectJ** un aspect `PointTrace` tel que le message `''un point va être modifié''` soit affiché au terminal à chaque fois que l'abcisse d'un `JBPoint` va être modifiée.
5. Définissez en **AspectJ** une autre version plus sophistiquée avec paramètres et avec un *advice* de type *around*, telle que le code suivant :

```
1     public static void main (String[] args){
2         JBPoint p1 = new JBPoint();
3         System.out.println("Begin ---");
4         p1.setX(33);
5         p1.setY(44); }
```

produise l'affichage ci-desous. On suppose que la méthode `toString()` existe sur `JBPoint`, inutile de l'écrire.

```
Begin ---
(0, 0) va être modifié --- 33
(33, 0) a été modifié ---
(33, 0) va être modifié --- 44
(33, 44) a été modifié ---
```

3 Composants EJB (environ 9 points)

Questions

1. Choisir la ou les réponses justes en justifiant. Les beans session sont des composants Java EE qui servent :
   1. à implémenter une partie de la logique métier d'une application utilisable par invocation de méthodes (envoi synchrone de messages), potentiellement distribuée
   2. à traiter la réception asynchrone, dans une file, de messages envoyés par des message-driven beans
   3. à produire dynamiquement du contenu Web aux clients en utilisant des objets de réponse HTTP
2. Écrire un bean session Compteur séquentiel, qui fournit deux méthodes, la première retourne à chaque invocation un nombre différent, et la deuxième ré-initialise le compteur.
3. Écrire un client de ce bean sous la forme d'une application Java SE. Celle-ci doit invoquer les deux méthodes du bean.

   Que faut-il paramétrer pour que cette application puisse être exécutée sur une machine différente de celle où le serveur est démarré ?
4. Choisir le ou les réponse(s) juste(s) et justifier. Avec Java EE, nous pouvons :
   1. Utiliser uniquement des services Web dans les beans
   2. Définir un bean session comme service Web et invoquer les opérations d'autres services Web
   3. Implémenter des composants qui ne peuvent jamais interagir avec des services Web
5. Est-ce que le bean session Compteur séquentiel discuté ci-dessus peut être déployé comme service Web ? Justifier votre réponse.

   Expliquer brièvement les étapes nécessaires à la publication des méthodes d'un bean comme opérations d'un service Web.
6. Choisir le ou les réponse(s) juste(s). Supposons l'existence de deux composants (bundles) OSGi. Si l'on souhaite les connecter en affectant la référence d'un objet (de type T) dans le deuxième bundle au champ d'un objet dans le premier bundle, on doit :
   1. déclarer un import-package dans le premier bundle (contenant l'interface requise T) et un export package dans le second (interface fournie T), puis laisser le framework rechercher les classes dans chaque bundle, qui doivent être instanciées
   2. déclarer un import-package et un export-package (comprenant chacun le package de T), puis instancier les classes (implémentant T) manuellement à l'intérieur de chaque bundle
   3. récupérer une référence de type T en utilisant une classe Factory ou l'annuaire de services, mais ne pas déclarer le package de T dans import-package et export-package
   4. faire ces déclarations (import- et export- package, comprenant chacun le package de T) d'abord, et ensuite récupérer une référence de type T (avec une Factory ou l'annuaire de services)
   5. déclarer un require-bundle (comprenant un bundle ayant une classe qui implémente T), simplement. Le framework gère l'instanciation et l'injection des dépendances
7. Que faut-il faire dans le code (et autour de celui-ci) du composant EJB Compteur séquentiel des questions précédentes pour le déployer comme composant OSGi ?
8. Supposons que l'on souhaite déployer l'application cliente du compteur comme un deuxième composant OSGi. Que faut-il faire ?
9. . Un composant (plugin) Eclipse est un bundle OSGi qui déclare des extensions et éventuellement des points d'extension. Soit un plugin A qui déclare un point d'extension P, et un plugin B qui contribue par des extensions à ce point. Dans le contrat de ce point d'extension P, il est déclaré un type abstrait I. Laquelle de ces propositions vous semble-t-elle correcte ?
   1. C'est le plugin A qui déclare l'interface fournie I et le plugin B l'interface requise I
   2. C'est le plugin B qui déclare l'interface fournie I et le plugin A l'interface requise I
   3. Aucun des deux composants ne déclare une interface fournie ou requise. Les deux composants sont très fortement couplés (ils se connaissent mutuellement statiquement)