eric.blaudez@u-psud.fr

IUT 2015-2016  
TD / TP JAVA

Programmation avancée

Nom Prénom e-mail Groupe TD/TP

MARTINEZ Guillaume 3E2

LAPEYRE Remi 3E2

Table des matières

Evaluation 3

Organisation des TD/TP 4

Avant de commencer … 4

A propos des TD/TP 4

TD/TP 1 – Rappels 5

Exercices 5

Exercice 1 : Classe Abstraite, Interface & Héritage 5

Exercice 2 : Collections 5

Exercice 3 : GUI – Visualisation de Widgets sous forme de table : implémentation partielle d’un agenda 6

Aide 6

Rapport 6

Description des travaux 6

Tous les exercices sont faits. 6

Difficultés rencontrées 6

TD/TP 2 – Gestion des erreurs, tests & bonnes pratiques 7

Exercices 7

Exercice 1 : Tests unitaires et debugger 7

Exercice 2 : Internationalisation 7

Exercice 3 : Gestion des erreurs (Exceptions) 7

Exercice 4 : GUI – Changer la langue d’une application 7

Exercice 5 : Créer sa bibliothèque & construire un livrable 7

Rapport 7

Description des travaux 7

Tous les exercices sont faits. 7

Difficultés rencontrées 7

TD/TP 3 – Généricité, Collections & Design patterns 9

Exercices 9

Exercice 0 9

Exercice 1 : Généricité – Création d’un lecteur d’arguments 9

Exercice 2 : Collections - créer une collection d’ExamEvent 9

Exercice 3 : Design pattern 9

Exercice 4 : GUI – Menu 9

Aide 10

Rapport 10

Description des travaux (+réponses aux questions) 10

L’application peut maintenant gérer la création et la suppression d’étudiant, de jury, de document, de salle de classe et d’examen. Il peut aussi gérer la modification d'un examen. Cependant, on ne peut pas encore choisir la date de l’examen car l’utilisateur ne peut pas sélectionner le jour de l’examen mais seulement le mois et l’année. 10

Difficultés rencontrées 10

TD/TP 4 – Les entrées / sorties 11

Exercices 11

Exercice 1 : Lire & écrire un fichier XML 11

Exercice 2: Créer et lire un fichier de configuration pour une application 11

Exercice 3 : Sauvegarder et charger l’état d’une application 11

Exercice 4 : GUI 11

Exercice 5 - facultatif: Base de données 11

Aide 11

Rapport 11

Description des travaux 11

Difficultés rencontrées 11

TD/TP 5 – Threads & Calcul distribué 12

Exercices 12

Exercice 1 : Sauvegarde en tâche de fond 12

Exercice 2 : ShutdownHook 12

Exercice 2 : Lancer des calculs distribués 12

Exercice 3 : GUI – configuration 12

Rapport 12

Description des travaux 12

Difficultés rencontrées 12

TD/TP 6 – Les applications en réseau 13

Exercices 13

Exercice 1 : Créer une application de Chat 13

Exercice 2 : Créer une Shoutbox avec une servlet (ou une page JSP) 13

Aide 13

Rapport 13

Description des travaux 13

Difficultés rencontrées 13

# Evaluation

|  |  |  |
| --- | --- | --- |
| **td /tp** | **Note** | **Commentaire** |
| **0 - Github** | /2 |  |
| **1** | /3 |  |
| **2** | /3 |  |
| **3** | /3 |  |
| **4** | /3 (+1) |  |
| **5** | /3 |  |
| **6** | /3 |  |

# Organisation des TD/TP

## Avant de commencer …

* Ces TD/TP sont accompagnés des codes sources qu’il s’agit de compléter.
* Il est **obligatoire** de créer un compte **GitHub** (***https://github.com***) par groupe de travail (normalement un binôme) **avant** le premier TD/TP (vous fournirez les adresses à vos chargés de TD). Chaque semaine, vous y déposerez vos travaux. La documentation de **GitHub** est disponible sur ***https://help.github.com/***
* Une copie de ce document se trouve dans le dossier « ***train*** », il devra être complété (nom, prénom, email et groupe - en première page) et *versionné* au même titre que vos travaux.
* La hiérarchie de dossiers (*versionnées*) devra être la suivante (le fichier zip avec le premier TP respecte cette hiérarchie, il suffira de *versionner* son contenu)

iutjava

workspace

iuttp

bin

src

edu/iut/...

libs

train

IUT-TD-TP-2015.doc

*<livrables>*.[jar & war]

run.sh

run.bat

***Dossiers***

***Workspace Eclipse***

***Fichiers***

## A propos des TD/TP

* A partir du TP 2, tous les codes fournis devront être **commentés** en ***JavaDoc*** et les classes que vous jugez critiques devront faire l’objet de **tests unitaires** (il faudra effectuer ce travail sur le TP1 également). Une note comprise entre 0,25 et 0,75 par TP sera donnée pour ce travail.
* Le code source à compléter est une base de travail, vous pouvez ajouter ou supprimer des objets.
* A l’issu de chaque TD/TP, vous **devez** compléter la partie « ***rapport*** » de ce document afin d’expliquer votre démarche et les difficultés rencontrées. Vous pouvez ajouter des diagrammes de classes et d’états.
* Tous les TP sont basés sur le même code : chaque TP est donc une amélioration du TP précédent. Sur votre compte ***GitHub***, il sera inutile de faire un dossier par TP, vous compléterez l’existant : suivez la hiérarchie de dossier proposée.
* La plupart des exercices seront utiles pour le projet.

# TD/TP 1 – Rappels

## Exercices

### Exercice 1 : Classe Abstraite, Interface & Héritage

Pour l’exercice suivant les classes se situent dans les *packages* ***edu.iut.app*** et ***edu.iut.gui.listeners***

* Compléter l’interface ***IApplicationLogListener*** ayant une méthode ***void newMessage(String level, String message)***
* Compléter une interface ***IApplicationLog*** ayant les méthodes suivantes
* ***void setMessage(String message)***
* ***String getMessage()***
* ***void addListener(IApplicationLogListener listener)***
* ***IApplicationLogListener[] getpplicationLogListeners()***
* Compléter la classe ***AbstractApplicationLog*** implémentant l’interface ***IApplicationLog***
  + La méthode ***setMessage*** sera abstraite
  + Cette classe devra contenir un tableau de ***IApplicationLogListener*** (sous forme de collection) avec une méthode d’ajout et une méthode permettant d’obtenir le tableau/
* Compléter les classes ***ApplicationErrorLog, ApplicationWarningsLog, ApplicationInfoLog*** dérivant de ***AbstractApplicationLog*** 
  + Ces classes devront contenir un tableau de ***IApplicationLogListener*** (sous forme de collection)
  + La méthode ***setMessage*** doit appeler la méthode ***newMessage*** de chaque élément du tableau de ***IApplicationLogListener*** et conserver le message
* Compléter la boîte de dialogue abstraite ***AbstractApplicationMessageDialog*** implémentant l’interface ***IApplicationLogListener***
  + ajouter une méthode abstraite ***void showMessage( )***, cette méthode doit être accessible par les classe filles seulement.
  + la méthode ***void newMessage(String message)*** doit appeler la méthode ***showMessage( )***
* Compléter les 3 boites de dialogue dérivant de ***AbstractApplicationMessageDialog***:
  + ***ApplicationErrorMessageDialog ,*** affichant un message d’erreur
  + ***ApplicationWarningMessageDialog***, affichant un avertissement
  + ***ApplicationInfoMessageDialog***, affichant un message

### Exercice 2 : Collections

Créer une table de logs ***ApplicationLogs***

* La classe ***ApplicationLogs*** doit dériver d’une Collection représentant un tableau de ***IApplicationLog***
* Implémenter les fonctions suivantes permettant de catégoriser les erreurs
  + ***ArrayList<IApplicationLog> getErrors(),***
  + ***ArrayList<IApplicationLog> getWarnings(),***
  + ***ArrayList<IApplicationLog> getInfos()***

### Exercice 3 : GUI – Visualisation de Widgets sous forme de table : implémentation partielle d’un agenda

* Implémenter 3 vues (Mois / Semaine / Jour) : classes ***EventPanel, MonthPanel, WeekPanel, DayPanel***
* Trouver un *Layout* permettant de superposer ces trois vues : classe ***SchedulerFrame***
* Créer une ***Factory*** (classe ***AgendaPanelFactory***)permettant de gérer les trois classes ***MonthPanel, WeekPanel, DayPanel***
* Créer une Panel permettant de gérer les vues ***MonthPanel, WeekPanel, DayPanel :*** en cliquant sur un bouton ‘***next***’ nous devons pouvoir voyager entre les vues

## Aide

* Operateur ***instanceof***
* ***JOptionPane***: <https://openclassrooms.com/courses/apprenez-a-programmer-en-java/les-menus-et-boites-de-dialogue>

## Rapport

### Description des travaux

### Tous les exercices sont faits.

### Difficultés rencontrées

Enoncés pas très clairs. Plus de mal à comprendre les énoncés et les objectifs qu’à programmer.

# TD/TP 2 – Gestion des erreurs, tests & bonnes pratiques

## Exercices

### Exercice 1 : Tests unitaires et *debugger*

* Pour chaque classe ‘non graphique’ du TP, créer un test unitaire.
* A l’aide du débugger, trouver l’(es) erreur(s) introduite(s) dans le code du TP2.

### Exercice 2 : Internationalisation

Créer un ***Singleton*** session, permettant de gérer l’internationalisation (français / anglais) ***edu.iut.app.ApplicationSession***

### Exercice 3 : Gestion des erreurs (Exceptions)

* Créer vos propres classes d’exceptions : créer une classe ***IUTException*** de laquelle dériveront toutes vos futures classes d’exception. ***IUTException*** devra pouvoir ‘logger’ les erreurs : en vous appuyant sur la classe de Session pour initialiser le ***logger***.
* Dans le TP1, nous avons créé les classes ***ApplicationErrorLog, ApplicationWarningsLog, ApplicationInfoLog*** permettant une gestion avancée des erreurs utilisant des ***listeners*** (permettant d’afficher des messages à l’écran). Il faudrait que ces messages soient aussi ‘***loggés***’ : modifier ces classes afin d’utiliser un ***logger*** initialiser dans la session.

### Exercice 4 : GUI – Changer la langue d’une application

En vous appuyant sur le code d’internationalisation de la session, faite en sorte que les éléments affichés (dans l’interface) ou dans les erreurs, soient dépendant de ***Locale*** (autrement dit, l’application devra pouvoir afficher de l’anglais et du français sans avoir été recompilée).

### Exercice 5 : Créer sa bibliothèque & construire un livrable

A l’aide d’***Eclipse***, créer une application (un ‘***.jar***’) livrable (à vos chargés de TD par exemple)

***Ce livrable devra désormais être disponible tout le temps (pour chaque TD/TP ou à chaque évolution majeur de vos codes, il faudra le reconstruire). Il faut donc le ‘versionner’ dans le dossier train. Les scripts ‘run.sh’ et ‘run.bat’ permettent d’exécuter votre application.***

## Rapport

### Description des travaux

### Tous les exercices sont faits.

### Difficultés rencontrées

Problèmes lors de la compréhension des objectifs des énoncés. L’appropriation du code était assez dur au début.

# TD/TP 3 – Généricité, Collections & *Design patterns*

## Exercices

### Exercice 0

Pourquoi, dans le TP1, a-t-on créé un ***Singleton*** (classe ***ApplicationSession***) ?

### Exercice 1 : Généricité – Création d’un lecteur d’arguments

* Créer une classe générique **Option,** Les options peuvent être des booléens, des entiers, des réels, des chaînes de caractères ou encore des fichiers. Une option est caractérisée par une clé, une valeur et la description de l’option (Note : pensez à l’internationalisation). Ici vous devrez travailler sur la classe ***CommandLineOption.***
* Créer une classe ***CommandLineParser***
* Modifier le programme principal pour utiliser votre ***parser*** d’options

### Exercice 2 : Collections - créer une collection d’***ExamEvent***

* Créer les classes :
  + Person,
  + Classroom,
  + Document
  + … toutes celles qui vous semblent nécessaires
* Créer la classe ***ExamEvent*** (cette classe doit répondre à la problématique soumise dans le projet … à vous d’en faire le ***design***). Elle doit contenir a minima un objet ***Person*** (celle passant l’examen), une collection de ***Person*** (le jury), un objet ***Classroom*** et naturellement toutes les informations ‘normales’ pour évènement.
* Créer la class ***Agenda*** dérivant d’une collection d’***ExamEvent***

### Exercice 3 : Design pattern

Créer un Filtre (voir aide) sur la collection ***ExamEvents***

* Critère pour les dates
* Critère pour les personnes
* Critère pour les salles
* Imaginer et implémenter d’autres critères (voir le projet)

### Exercice 4 : GUI – Menu

* Ajouter un menu à votre application

File

Load

Save

quit

Edit

View

Month

Week

Day

Help

Display

About

Implémenter les fonctionnalités du ‘menu’ ***Edit*** : les ***menuitems*** de ‘***View***’ doivent repoduire le comportement du bouton ‘***Next***’ de l’interface. Pour les autres menus, afficher un message (***DialogBox***) d’information prévenant l’utilisateur que la fonctionnalité n’est pas encore implémentée.

* Remplacer le bouton ‘***Next***’ par
  + Un ***Spinner*** contenant les années de 2010 à l’année 2020 (positionné sur l’année courante)
  + Une ***ComboBox*** contenant les mois de l’année (positionné sur le mois courant)
  + Une ***ComboBox*** contenant les jours de la semaine (positionné sur le jour courant)

(Pensez à l’internationalisation)

* Créer les widgets pour l’affichage d’un ‘***ExamEvent***’, en lien avec une Collection d’***ExamEvent*** (Exercice 2). Ces widgets devront permettre l’ajout/la suppression et la modification d’un ***ExamEvent*.**

## Aide

**Filter Pattern** : <http://www.tutorialspoint.com/design_pattern/filter_pattern.htm>

**Menu :** <https://docs.oracle.com/javase/tutorial/uiswing/components/menu.html>

**Spinner :** <https://docs.oracle.com/javase/tutorial/uiswing/components/spinner.html>

## Rapport

### Description des travaux (+réponses aux questions)

### L’application peut maintenant gérer la création et la suppression d’étudiant, de jury, de document, de salle de classe et d’examen. Il peut aussi gérer la modification d'un examen. Cependant, on ne peut pas encore choisir la date de l’examen car l’utilisateur ne peut pas sélectionner le jour de l’examen mais seulement le mois et l’année.

### Difficultés rencontrées

La seule difficulté était de créer un environnement intuitif pour l’utilisateur avec toutes les options pour gérer de façon optimale et rapide les examens.

# TD/TP 4 – Les entrées / sorties

## Exercices

### Exercice 1 : Lire & écrire un fichier XML

* Sauvegarder le contenu de l’application (agenda) dans un fichier XML.
* Utiliser ***CommanLineParser*** (TP3) avec l’option ***–project=<FILE>*** pour pouvoir charger un contenu d’application préalablement sauvegarder.

### Exercice 2: Créer et lire un fichier de configuration pour une application

* La configuration doit contenir le chemin du fichier de log et le chemin d’un dossier temporaire (il faudra donc ajouter dans la session la possibilité de ***loggé*** dans un fichier et les accesseurs fournissant le chemin vers ce dossier temporaire)
* Utiliser ***CommanLineParser*** (TP3) avec l’option ***–config=<FILE>*** pour pouvoir charger une configuration

### Exercice 3 : Sauvegarder et charger l’état d’une application

Sérialiser les objets contenant de l’information dans le dossier temporaire (initialiser avec la configuration)

### Exercice 4 : GUI

* Lire et visualiser une aide HTML (vous pouvez activer le ***menuitem*** ‘help’ et ‘about’)
* Activer les ***menuitems*** ‘save’ et ‘load’
* Ajouter une interface permettant d’ajouter/modifier une personne (ces informations seront sauvegardées à part dans un fichier XML)

### Exercice 5 - facultatif: Base de données

Vous pouvez ajouter une fonctionnalité offrant la possibilité d’effectuer une sauvegarde dans une base de données.

## Aide

* Afficher de l’HTML : <http://alvinalexander.com/blog/post/jfc-swing/how-create-simple-swing-html-viewer-browser-java>
* Base de données : <http://www.tutorialspoint.com/sqlite/sqlite_java.htm>

## Rapport

### Description des travaux

### Difficultés rencontrées

# TD/TP 5 – *Threads* & Calcul distribué

## Exercices

### Exercice 1 : Sauvegarde en tâche de fond

En utilisant un Thread et le système de sauvegarde du TD/TP 4 (exercice 1), sauvegarder le projet toute les minutes.

### Exercice 2 : ***ShutdownHook***

En utilisant le code implémenté dans le TD/TP 4 (exercice 3) écrire une ***ShutdownHook*** pour faire un système de reprise sur erreur en cas de crash.

### Exercice 2 : Lancer des calculs distribués

Dans le TP2 nous avons créé un Filtre, il s’agit ‘de l’externaliser’ afin que les recherches selon un critère (qui peut être complexe) soient effectuées sur une autre machine.

Cette fonctionnalité ne doit en rien casser l’existant : l’utilisateur devra pouvoir utiliser des calculs externalisés ou bien ses propres ressources.

Ajouter dans le fichier de configuration l’adresse du server RMI, avec un booléen permettant d’activer (ou de désactiver) la fonctionnalité.

### Exercice 3 : GUI – configuration

Créer une ‘***DialogBox***’ *Settings* permettant de configurer entre autre l’***autosave*** (est-ce que l’***autosave*** est activé ? à quelle fréquence ?).

* Cette ‘***DialogBox***’ devra être accessible depuis le menu (***edit->settings***)
* La configuration devra être sauvegardée dans un fichier de configuration
* Les ***settings*** ne concernent pas uniquement la sauvegarde : ajouter d’autre settings (comme le chemin pour le fichier log, l’adresse du server RMI ou encore la langue par exemple)

## Rapport

### Description des travaux

### Difficultés rencontrées

# TD/TP 6 – Les applications en réseau

## Exercices

### Exercice 1 : Créer une application de *Chat*

* Prévoir un mode ‘***public’*** et ‘***private***’
* Ajouter dans le fichier de configuration l’adresse et le port de votre server

### Exercice 2 : Créer une *Shoutbox* avec une *servlet (ou une page JSP)*

* Cette ***shoutbox***  devra se connecter au server pour afficher les commentaires ‘public’ uniquement
* Il sera aussi possible de poster un commentaire

## Aide

***Shoutbox*** : <https://fr.wikipedia.org/wiki/Shoutbox>

## Rapport

### Description des travaux

### Difficultés rencontrées