**Durée : 1h30 - AUCUN DOCUMENT - AUCUNE CALCULATRICE - SUJET À JOINDRE À LA COPIE**

**Instructions :**

* Ce TP noté de 1h30 est un travail d’une équipe en binôme sur NetBeans dans le même groupe de TP.
* Compresser par zip ou rar ou 7z tous les dossiers et fichiers de votre projet du code développé sur NetBeans (par exemple,pour le duo Manolo Hina et JP Segado, ils doivent déposer le dossier compressé ***Hina\_Segado.zip***)
* Déposer ce dossier compressé **à la fin de la séance de TP** sur la page campus <https://pedago-ece.campusonline.me/course/view.php?id=124>

Des extraits de la documentation des API Java sont fournis en **annexe** dans la dernière page.

**Votre code devra être commenté pour plus de clarté, sous peine d’être pénalisé.**

**Des extraits de la documentation des api sont fournis en annexe des pages 4 a 8.**

On souhaite écrire un programme qui réserve des voyages en train par une personne. Ci-dessous, un extrait du diagramme de classes vous propose certaines classes nécessaires pour la réservation. Rappel sur les niveaux de visibilité : **- private**, **+ public** et **# protected**

**reserver**

**Personne**

**Voyage**

**# String** nom

**# String** prenom

**# String** email

1..\*

1

**Gare**

* **boolean** retour
* **boolean** direct
* **int** nb\_passagers
* **int** confort
* **String** nom

1..\* **aller** 1

**+** Personne ()

**+ void** aller (…)

**+ void** revenir (…)

**+ void** reserver (…)

**+** Gare (**String** nom)

1..\* **revenir** 1

La classe **Voyage** fournit les informations suivantes :

* Le booléen *retour* indique s’il y a retour ou pas pour le voyage
* Le booléen *direct* indique si le voyage est un trajet direct ou pas
* L’entier *nb\_passagers* indique le nombre de passagers du voyage et l’entier *confort* donne la classe (1 ou 2)
* La méthode *aller(…)* copie la gare de départ, la gare d’arrivée, la date au format ″dd-MM-yyyy″ (exemple : ″15-05-2015″) et l’heure (entier) en paramètres dans les attributs pour l’aller du voyage
* La méthode *revenir(…)* copie la date et l’heure en paramètres dans les attributs pour le retour du voyage
* La méthode *reserver(…)* copie la personne qui réserve et le nombre de passagers en paramètres dans les attributs, puis instancie le tableau de passagers avec ce nombre de passagers.

La classe **Gare** fournit les informations suivantes :

* Le *nom* de la gare et un constructeur qui initialise son nom avec le paramètre

La classe **Personne** fournit les informations suivantes :

* Le *nom*, *prenom* et *email* de la personne, ainsi qu’un constructeur par défaut (sans paramètre)

*Pour chacune des classes suivantes, n’oubliez pas d’importer les éventuelles librairies nécessaires. Nous supposons que ces classes font partie du même package.*

[5 points] Implémenter en Java la classe **Voyage** en respect desconsignes suivantes :

* Les attributs suivants en **private** :
* Les 4 attributs définis dans l’extrait du diagramme de la page précédente
* La date et l’heure à l’aller et au retour
* La personne qui réserve, un tableau de passagers et les gares de départ et d’arrivée
* Implémenter un premier constructeur par défaut (sans paramètre) qui initialise les 4 attributs définis dans l’extrait du diagramme selon les 4 règles suivantes : il n’y a pas de retour, le trajet est direct, il y a 1 seul passager et le confort est en classe 2.
* Implémenter un second constructeur qui copie ses 4 paramètres respectivement dans les 4 attributs définis dans l’extrait du diagramme.
* Implémenter les méthodes *aller(…)*, *revenir(…)* et *reserver(…)* en respect des informations indiquées dans la première page.

[2 points] Implémenter en Java la classe **Gare** en respect desconsignes spécifiées en première page.

[2 points] Implémenter en Java la classe **Personne** en respect desconsignes spécifiées en première page.

[3 points] Implémenter en Java la classe **Passager** en respect desconsignes suivantes :

* Un passager est une personne fournissant les informations suivantes : la tranche d’âge (exemples : ″26-59 ans″, ″60 et +″ etc.), la carte d’abonnement (exemples : ″carte 12-25″, ″famille nombreuse″ etc.) et le programme de fidélité (exemples : ″grand voyageur″, ″S’Miles″ etc.). La personne qui réserve n’est pas obligatoirement un passager mais un passager peut être la personne qui réserve.

[5 points] Implémenter en Java une classe avec le **main** qui effectue les actions suivantes :

* Déclarer et lire au clavier (voir la classe **Scanner** en annexe page 4) les données suivantes sans tester leurs valeurs :
* Les booléens pour le retour et le trajet direct.
* Les noms des gares de départ et d’arrivée.
* La date à l’aller au format ″dd-MM-yyyy″ (voir classes **DateFormat** et **SimpleDateFormat** en annexe pages 4 et 5) et l’heure à l’aller.
* Le nombre de passagers et la classe de confort.
* Si le retour lu est vrai, lire la date et l’heure au retour.
* En cas d’erreur de format de lecture sur l’une des données, générer l’exception **Exception** et afficher un message d’erreur.
* Instancier un objet de **Personne** avec son constructeur par défaut (sans paramètre).
* Instancier les gares de départ et d’arrivée (voir le constructeur de **Gare** dans le diagramme) avec les noms lus.
* Instancier un objet de **Voyage** avec votre second constructeur de la question précédente et les paramètres adéquats lus précédemment.
* A partir de votre objet instancié de **Voyage**, appeler vos méthodes *aller(…),* *revenir(…)* seulement si le retour lu est vrai et *reserver(…)* de la question précédente avec les paramètres adéquats lus précédemment.

**ANNEXE : extraits de la documentation API**

## java.util

## Class Scanner

|  |  |  |
| --- | --- | --- |
| **Constructor Summary** | | |
| [**Scanner**](http://docs.oracle.com/javase/1.5.0/docs/api/java/util/Scanner.html#Scanner%28java.io.InputStream%29)([InputStream](http://docs.oracle.com/javase/1.5.0/docs/api/java/io/InputStream.html" \o "class in java.io) source)            Constructs a new Scanner that produces values scanned from the specified input stream. | |  |
| **Method Summary** | |
| void | [**close**](http://docs.oracle.com/javase/1.5.0/docs/api/java/util/Scanner.html#close%28%29)()            Closes this scanner. |
| boolean | [**hasNext**](http://docs.oracle.com/javase/1.5.0/docs/api/java/util/Scanner.html#hasNext%28%29)()            Returns true if this scanner has another token in its input. |
| [String](http://docs.oracle.com/javase/1.5.0/docs/api/java/lang/String.html) | [**next**](http://docs.oracle.com/javase/1.5.0/docs/api/java/util/Scanner.html#next%28%29)()            Finds and returns the next complete token from this scanner. |
| boolean | [**nextBoolean**](http://docs.oracle.com/javase/1.5.0/docs/api/java/util/Scanner.html#nextBoolean%28%29)()            Scans the next token of the input into a boolean value and returns that value. |
| double | [**nextDouble**](http://docs.oracle.com/javase/1.5.0/docs/api/java/util/Scanner.html#nextDouble%28%29)()            Scans the next token of the input as a double. |
| float | [**nextFloat**](http://docs.oracle.com/javase/1.5.0/docs/api/java/util/Scanner.html#nextFloat%28%29)()            Scans the next token of the input as a float. |
| int | [**nextInt**](http://docs.oracle.com/javase/1.5.0/docs/api/java/util/Scanner.html#nextInt%28%29)()            Scans the next token of the input as an int. |
| [String](http://docs.oracle.com/javase/1.5.0/docs/api/java/lang/String.html) | [**nextLine**](http://docs.oracle.com/javase/1.5.0/docs/api/java/util/Scanner.html#nextLine%28%29)()            Advances this scanner past the current line and returns the input that was skipped. |
| [Scanner](http://docs.oracle.com/javase/1.5.0/docs/api/java/util/Scanner.html) | [**useDelimiter**](http://docs.oracle.com/javase/1.5.0/docs/api/java/util/Scanner.html#useDelimiter%28java.lang.String%29)([String](http://docs.oracle.com/javase/1.5.0/docs/api/java/lang/String.html) pattern)            Sets this scanner's delimiting pattern to a pattern constructed from the specified String. |

## 

## java.text

## Class DateFormat

[java.lang.Object](http://docs.oracle.com/javase/1.4.2/docs/api/java/lang/Object.html)

extended by[java.text.Format](http://docs.oracle.com/javase/1.4.2/docs/api/java/text/Format.html)

extended by**java.text.DateFormat**

|  |  |  |
| --- | --- | --- |
| **Constructor Summary** | | |
| protected | [**DateFormat**](http://docs.oracle.com/javase/1.4.2/docs/api/java/text/DateFormat.html#DateFormat%28%29)()            Create a new date format. | |
| **Method Summary** | | |
| [**String**](http://docs.oracle.com/javase/1.4.2/docs/api/java/lang/String.html) | | [**format**](http://docs.oracle.com/javase/1.4.2/docs/api/java/text/DateFormat.html#format%28java.util.Date%29)**(**[**Date**](http://docs.oracle.com/javase/1.4.2/docs/api/java/util/Date.html)**date)** Formats a Date into a date/time string. |
| [Date](http://docs.oracle.com/javase/1.4.2/docs/api/java/util/Date.html) | | [**parse**](http://docs.oracle.com/javase/1.4.2/docs/api/java/text/DateFormat.html#parse%28java.lang.String%29)([String](http://docs.oracle.com/javase/1.4.2/docs/api/java/lang/String.html) source)            Parses text from the beginning of the given string to produce a date. |

## java.text

## Class SimpleDateFormat

[java.lang.Object](http://docs.oracle.com/javase/1.4.2/docs/api/java/lang/Object.html)

extended by[java.text.Format](http://docs.oracle.com/javase/1.4.2/docs/api/java/text/Format.html)

extended by[java.text.DateFormat](http://docs.oracle.com/javase/1.4.2/docs/api/java/text/DateFormat.html)

extended by**java.text.SimpleDateFormat**

|  |  |  |
| --- | --- | --- |
| **Constructor Summary** | | |
| [**SimpleDateFormat**](http://docs.oracle.com/javase/1.4.2/docs/api/java/text/SimpleDateFormat.html#SimpleDateFormat%28java.lang.String%29)([String](http://docs.oracle.com/javase/1.4.2/docs/api/java/lang/String.html) pattern)            Constructs a SimpleDateFormat using the given pattern and the default date format symbols for the default locale. | |  |
| **Method Summary** | | |
| [**String**](http://docs.oracle.com/javase/1.4.2/docs/api/java/lang/String.html) | [**toPattern**](http://docs.oracle.com/javase/1.4.2/docs/api/java/text/SimpleDateFormat.html#toPattern%28%29)()            Returns a pattern string describing this date format. | |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Letter** | **Date or Time Component** | **Presentation** | **Examples** |  | **Date and Time Pattern** | **Result** |
| y | Year | [Year](http://docs.oracle.com/javase/1.4.2/docs/api/java/text/SimpleDateFormat.html#year) | 1996; 96 |  | "dd-MM-yyyy" | 14-04-2012 |
| M | Month in year | [Month](http://docs.oracle.com/javase/1.4.2/docs/api/java/text/SimpleDateFormat.html#month) | July; Jul; 07 |  |
| d | Day in month | [Number](http://docs.oracle.com/javase/1.4.2/docs/api/java/text/SimpleDateFormat.html#number) | 10 |  |  |  |