FICHIER LOG

Fichier log

- Fichier texte, extension .log
- Regroupe les activités de l'application
- Appelé fichier journal ou journal d'évènement

Informations stockées

- Date
- Heure
- Classe/Méthode
- Message

Les fichiers log

Intérêt

- Aide à la programmation
- Traçabilité des activités
- Identification d'un problème
- Gestion des erreurs

Inconvénient

- L'accès en écriture au fichier prend du temps

Jeu des héros

- On souhaite gérer le déroulement des phases de jeu dans un fichier log

- Les erreurs liées à des exceptions seront écrites dans un fichier log

Eléments à enregistrer

Déroulement du jeu

- Début d'une partie
- Type de création des héros
- Nombre de joueurs
- Héros éliminés et héros vainqueurs
- Fin de la partie

Exceptions

- HerosException
- Erreurs de saisie dans la création des héros

Gestion des exceptions

- Dans la fonction creationHeros du moteur de jeu, ajouter la gestion des exceptions suivantes :
  - InputMismatchException en cas de type entré erroné
  - ValeurException (exception personnalisée) si le nombre de points de vie ou la puissance de défense sont négatifs

- Lever les exceptions en cas d'erreur
- Faire gérer les exceptions au niveau de la création des objets (main)
  - Dans un premier temps, les messages définis pour chaque erreurs seront affichés à l'écran (System.out.println)

Utilisation d'un fichier log

Gestion du fichier log

- Dans un second temps, on souhaite écrire les messages d'erreur dans un fichier log

- Utilisation de :
  - Librairie java.util.logging
  - Classe Logger : permet de gérer l'écriture des logs
  - Classe FileHandler : permet d'utiliser un fichier comme sortie

Création du logger

- protected static Logger logger = Logger.getLogger("myPackage.mySubPackage.myClasse");

- Possibilité de créer un logger pour le package
- Possibilité de créer un logger par classe

- Le journal est static : créé une seule fois puis partagé dans toutes les classes du package

Constructeurs de FileHandler

- FileHandler()
- FileHandler(String pattern)
- FileHandler(String pattern, boolean append)
- FileHandler(String pattern, int limit, int count)
- FileHandler(String pattern, int limit, int count, boolean append)

Attributs supplémentaires

- Pattern : le nom du fichier (défini par le système par défaut)
- Limit : taille limite du fichier (en octets)
- Count (nombre de fichiers cycliques)
- Mode d'appel : le fichier est recrée (false) ou repris tel quel (true)

Définition de la sortie

- On associe le logger à une sortie : handler
  - Console
  - Fichier texte
  - Fichier XML…

Exemple, définir un fichier en sortie

- Utilisation de la classe FileHandler

- Handler fh = new FileHandler("myLog.log");
- logger.addHandler(fh);

Enregistrer un message

- Utilisation de la méthode log(Level level, String msg) de l'objet Logger

- Indique le niveau et le contenu du message

Niveaux de message

Niveau	Description
ALL	Tous les niveaux
SEVERE	Niveau le plus élevé
WARNING	Avertissement
INFO	Information
CONFIG	Configuration
FINE	Niveau faible
FINER	Niveau encore plus faible
FINEST	Niveau le plus faible
OFF	Aucun niveau

Définition des messages enregistrés

- Par défaut, à partir du niveau INFO
- Peut être modifié avec la méthode de la classe Logger
  - Void setLevel(Level level)

Envoi de message simplifié

- Méthodes alias pour envoyer des messages sans indiquer le niveau
  - void severe(String msg)
  - void warning(String msg)
  - void info(String msg)
  - void config(String msg)
  - void fine(String msg)
  - void finer(String msg)
  - void finest(String msg)

Journaliser une exception

- Méthode throwing de la classe Logger
  - void throwing(String sourceClass, String sourceMethod,Throwable thrown)

- Permet d'enregistrer la classe et la méthode qui ont levé l'exception

Format du fichier

- Format fichier texte : SimpleFormatter
- Format XML : XMLFormatter

- Pour modifier le format, utiliser la méthode de la classe Handler
  - void setFormatter(Formatter newFormatter)

Fichier de configuration

- Lib/logging.properties
  - handlers=java.util.logging.FileHandler, java.util.logging.ConsoleHandler
  - .level= INFO
  - java.util.logging.FileHandler.pattern = %h/java%u.log
  - java.util.logging.FileHandler.limit = 50000
  - java.util.logging.FileHandler.count = 1
  - java.util.logging.FileHandler.formatter = java.util.logging.XMLFormatter
  - java.util.logging.ConsoleHandler.level = INFO
  - java.util.logging.ConsoleHandler.formatter = java.util.logging.SimpleFormatter
  - myPackage.mySubPackage.myClass.level = SEVERE

Utilisation du fichier log

- Créer le Logger comme static dans la classe du main
- Configurer le logger et son fichier dans le main
- Capter les différentes exceptions pour faire écrire les messages dans le fichier log
  - Erreur de saisie : niveau sévère
  - Déroulement du jeu : niveau warning
  - HerosException: niveau info

Gestion des messages

- Tester l'exécution de l'application
  - 1- en écrivant les messages dans le logger avec le niveau par défaut
  - 2- en écrivant tous les messages dans le logger
  - 3- en écrivant les messages dans le logger avec le niveau severe