TP-Projet

V-Finale

Notation du projet

Versions

- Existence d'une branche « Prod »
  - Dernière version corrigée
- Rendu d'une version
  - Dès que possible : Merge request
    - Master -> Prod
  - Deadline: Validée par l'enseignant
  - Centralise les remarques et discussions
  - Contiendra les notes (temporaires et finales)

Notation

- Note intermédiaire divisée en 2 moitiées
  1. **Livraison** – par rendu (notes fixes)
  2. **Complétude** – final (notes préliminaires)
- Note finale = Intermédiaire x %Qualité
  - Va de 50% à 100%
  - Lisibilité
  - Maintenabilité
  - Estimée en cours de projet, figée en fin

Livraison

- ~2 points par version
- % de tests fonctionnels automatisés
- Accordé si réalisé à la date de rendu

Complétude

- 10 points
- % de tous les tests automatisés
- Inclus les tests de version
- Peut couvrir plus de cas
- Accordé à la date du rendu final

Qualité

- Evaluation humaine
  - Nommage limpide et parlant
  - Fonctions courtes
  - Algorithmes simples
  - Qualité du design
- Sans garantie : évaluation en continue
- Accordé à la date du rendu final

Le sujet

Casper-Sky

- Simulation « ludique »
  - Librement inspirée des antivirus et des ransomware
  - Permet d'aborder la modélisation objets, le le multithread, le réseau, l'événementiel etc…

Casper-Sky

- Un réseau est touché par un ransomware
  - Se répand de manière exponentielle
  - Un antivirus tente de l'endiguer
  - De plus en plus d'ordinateurs sont définitivement cryptés

V1 – Moteur

Un système événementiel extensible

Objectifs

- Tests d'intégrations automatisés
  - Une classe de test préexistante
  - L'implémenter
  - Faire passer les tests
- Moteur pour événements différés
  - File d'attente d'événements
  - Déclenchement à chaque update
- Programme temps réel

Système d'événement

- Tout se basera sur des événements différés
  - Un ordinateur sera infecté
  - Un événement d'encryption sera enregistré
  - Cet événement se déclenchera N secondes plus tard
  - Le temps de déclenchement peut changer

Concept d'événement

L'architecture imposée à la solution

Interface des événements

```
public interface Event {
    void trigger();
    Duration getDuration();
}
```

- Trigger : déclenche l'évènement
- getDuration : durée avant le déclencement

Détail des évènement

- Trigger: déclenche l'évènement
  - Ex: installe le ransomware sur le système cible
  - Ex: marque le système courant comme encrypté
- getDuration: durée totale
  - Même résultat quelque soit la date ou la situation

Gestionnaire d'événement

- Stocke les événements en attente
- Possède une méthode « **register** »
  - Stocke en interne un nouvel événement
- Possède une méthode « **update** »
  - Déclenche tous les événements expirés
  - Dans l'ordre en avançant dans le temps
  - Stocke la date du dernier update

Interfaces du Gestionnaire

- ```
  public interface EventQueue {
      void register(Event... events);
  }
  ```
- ```
  public interface TimeManager {
      void update();
      Instant getCurrentInstant();
  }
  ```

V2 – Entités

Une simulation de base

Entités

Les objets du monde réels que l'on va simuler

Les entités majeures

- Réseau
  - Contient des ordinateurs (postes utilisateurs et serveurs)
- Ordinateur
  - Peut ou non être infecté
  - Peut ou non avoir ses données encryptées

Actions

Ce que l'on peut faire pour altérer la simulation

Ordinateur: on peut

- Installer le ransomware sur un ordinateur
  - Utilisé une fois en début de partie
- Propager le ransomware (ex: via phishing ou partage réseau public)
  - S'il est infecté et connecté au réseau
- Compléter l'encryption des données
  - S'il est infecté.

Informations et affichages

Les données que l'on souhaite tirer de la simulation

Statistiques

- On peut obtenir le nombre actuel d'ordinateur
  - Non-formaté vs total
  - Infecté

Types d'événements

Ce qu'il peut se passer dans la simulation

L'infection initiale

- Causes :
  - Lancement de la simulation
- Caractéristiques:
  - Temps avant déclenchement : 3 secondes
- Conséquence :
  - Choisis un ordinateur au hasard
  - Y installe et lance le Ransomeware

La propagation

- Causes :
  - Un ordinateur est initialement infecté
  - Fin d'une propagation précédente (répétitif)
- Caractéristiques:
  - Temps avant déclenchement : 5 secondes
- Conséquence :
  - Infecte un ordinateur sain au hasard

Contraintes et tests

Interface Simulation

- Deux méthodes de statistiques
  - Nombre d'ordinateurs d'origine
  - Nombre d'ordinateurs infectés
- Une méthode update
  - Fait avancer la simulation jusqu'à la date référencée par l'horloge
- Une méthode getCurrentInstant
  - Renvoi la dernière date d'exécution de la simulation

Système aléatoire

- Sélection aléatoire d'un ordinateur
  - Pour en infecter un
  - Pour que l'un contamine un autre
- On définira une interface
  - Prends une liste d'ordinateur
  - En retourne un
- On créera une implémentation
  - Retourne un ordinateur au hasard

Interface de Selecteur (Mock)

- `public interface` **`Selector`** `{`

```
    <TItem> TItem selectAmong(
        List<TItem> choices,
        SelectionContext context);
}
```

- `public enum` **`SelectionContext`**
  - Peut valoir: `InitialInfection, Spreading`
- Utile uniquement pour les tests

Main

- Instancie un Selecteur basé sur Random
  - Pas utilisé par les tests
  - Utilise Random
- Instancie une SystemClock
- Instancie la classe Simulation
- Lance un Thread d'update
  - Appelle update toutes les 200 ms
- A la première saisie utilisateur, arrête la simulation

Ce qui est attendu

Ce qui est attendu

- Créer la simulation
  - Une classe qui implémente l'interface Simulation
  - Un ensemble de classe représentant la simulation (modèle)
- Brancher la seconde batterie de tests
  - Une classe qui étends BaseSimulationTests
  - Elle instancie la classe de simulation avec les bons paramètres
- Faire un main