Haute École Bruxelles-Brabant
École Supérieure d'Informatique
Bachelor en Informatique

ATL – Ateliers Logiciels

Projet

Le jeu Turing Machine

1 Consignes

Votre projet sera déposé régulièrement sur le serveur gitlab [1] dans un dépôt personnel en suivant les instructions de votre enseignant.

Une seule remise est attendue sur ce dépôt :

- ▷ le samedi 16 décembre à 22h ;

Les critères d'évaluation sont décrits dans le document Organisation de l'AA [2] sur *poESI*.

2 Description du jeu

Turing Machine est un jeu de puzzle qui peut se jouer seul ou à plusieurs, de façon coopérative ou compétitive. Il vous est ici demandé d'implémenter une version *en solitaire* de ce jeu.

Au début d'une partie de Turing Machine, la machine génère un code à trois chiffres, et quatre à six *validateurs*. Chaque chiffre peut prendre une valeur entre 1 et 5. Chaque validateur permet d'interroger la machine sur le code secret ; ils sont fixés pour l'entièreté de la partie.

Pour cela, à chaque manche du jeu, le(s) joueur(s) composent un code à trois chiffre. Ensuite, ils choisissent entre un et trois validateurs, qu'ils testent avec le code composé. Chaque validateur teste une caractéristique précise du code, et renvoie un résultat vrai ou faux : *vrai* veut dire que le code à deviner possède la même caractéristique que le code composé par le joueur ; *faux* signifie que les deux codes ont des caractéristiques différentes.

Chaque validateur fonctionne de la façon suivante :

- ▷ Une question est posée, sous la forme d'une expression mathématique
- ▷ La solution du problème est placée dans une parmi plusieurs catégories en réponse à cette question (ceci peut être fait une fois en début de partie et mis en mémoire).
- ▷ Le code entré par le joueur est placé dans une catégorie. S'il s'agit de la **même** catégorie que la solution du problème, le validateur renvoie une réponse favorable. S'il s'agit d'une autre catégorie, le validateur renvoie une réponse défavorable.

1. https://git.esi-bru.be
2. https://poesi.esi-bru.be/pluginfile.php/1897/mod_resource/content/2/ATLG3-Organisation.pdf

Version du document
8 novembre 2023

Prenons par exemple un code secret de 123 et quelques validateurs :

- Le validateur n°2[3] compare la valeur du premier chiffre du code avec 3 : ce premier chiffre peut être *inférieur à 3*, *égal à 3* ou *supérieur à 3*. Puisque le code secret est 123, son premier chiffre est placé dans la catégorie *inférieur à 3*. Dès lors, lorsque le joueur sélectionne ce validateur…
  - …avec le code 253, son premier chiffre est également *inférieur à 3* et le validateur renvoie **vrai** car les deux codes sont dans la même catégorie.
  - …avec le code 325, son premier chiffre est *égal à 3*, le validateur renvoie **faux** car ce n'est pas la catégorie à laquelle le code secret appartient.
  - …avec le code 455, son premier chiffre est *supérieur à 3*, le validateur renvoie **faux** également.
- Le validateur n°10 compte combien de fois le chiffre 4 apparaît dans un code : quatre catégories sont créées selon que le chiffre 4 est présent 0, 1, 2 ou 3 fois. Puisque le code secret est 123, il est placé dans la catégorie *aucun 4*. Par conséquent, si le joueur choisit ce validateur…
  - …avec le code 531, le validateur renvoie **vrai** car le chiffre 4 n'est pas présent
  - …avec les codes 241, 441 ou 444, le validateur renvoie **faux** car le chiffre 4 est présent respectivement une fois, deux fois, ou trois fois, ce qui est différent du code secret.
- Le validateur n°15 regarde quel chiffre est le plus grand (strictement) dans le code, et place le code dans une parmi quatre catégorie : *pas de maximum* (il y a un ex aequo), *le premier chiffre est maximum*, *le deuxième chiffre est maximum*, ou *le troisième chiffre est maximum*. Puisque le code secret est 123, il est placé dans la catégorie *le troisième chiffre est le maximum*. Lorsque le joueur sélectionne ce validateur…
  - …avec le code 424, il n'y a pas de maximum (car le premier et troisième chiffre sont tous deux supérieurs au deuxième chiffre, mais sont égaux). Le validateur renvoie **faux**,
  - …avec le code 521 ou 232, il y a un chiffre maximum mais il s'agit respectivement du premier et du deuxième chiffre ; ces codes sont donc dans des catégories différentes du code secret et le validateur renvoie **faux**.
  - …avec le code 245, le troisième chiffre est le maximum. Il s'agit de la catégorie dans laquelle le code secret se trouve, et le validateur renvoie donc **vrai**.

À la fin d'une manche, un joueur peut décider d'annoncer qu'il connaît le code à deviner. Il le vérifie auprès de la machine ; si le code est incorrect, le joueur a perdu et est éliminé. Si le code est correct, il a gagné. L'objectif est d'arriver à trouver le code secret *en interrogeant le moins de validateurs possible* (le nombre d'interrogations fait donc office de score, qu'il faut minimiser).

Le jeu vient avec une série de problèmes pré-définis par les créateurs, qui combinent un code avec un ensemble de validateurs choisis afin de régler la difficulté du défi. Une sélection de ces problèmes adaptées aux consignes du projet vous est fournie sur poESI dans le fichier `known_problems.csv` ; ces problèmes sont stockés au format CSV, reprenant le numéro du problème, leur niveau de difficulté, leur degré de hasard, le code secret (solution du problème) et les numéros des validateurs intégrés.

3. Voir la liste des validateurs au point 3

2.1 Références

Pour plus d'informations, une vidéo d'introduction est disponible à l'adresse https://www.youtube.com/watch?v=Mjbj2qIpq8A, qui présente l'ensemble du jeu en 10 minutes. D'autres liens utiles :

- ▷ Le site du jeu [4] contient notamment le manuel de règles complet et un accès à une batterie de problèmes.
- ▷ Le site de l'éditeur [5] propose quelques informations supplémentaires.
- ▷ Le jeu est présent sur Board Game Arena [6], en version non gratuite malheureusement, mais il est possible d'y observer des parties en cours ou obtenir davantage d'informations sur le jeu.

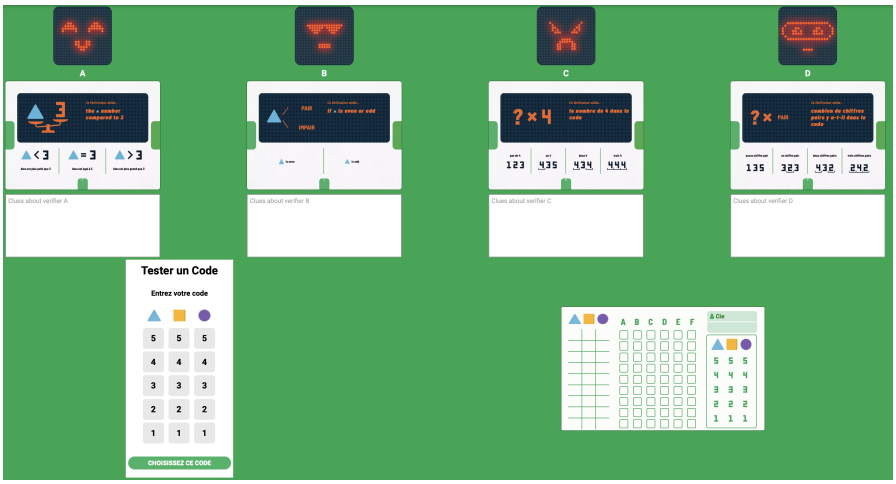

FIGURE 1 – Interface de Turing Machine sur Board Game Arena.

4. https://turingmachine.info
5. https://www.scorpionmasque.com/fr/turingmachine
6. https://boardgamearena.com/gamepanel?game=turingmachine

3 Le projet

Le projet comprend :

1. le modèle, testé et documenté ;
2. une vue console permettant de jouer ;
3. une vue Javafx permettant de jouer.

Modèle

Le modèle comprend toutes les classes gérant la logique du jeu.

> Le modèle sera **complet**, **testé** et **documenté**.

Afin qu'une vue puisse interagir avec le modèle sans en connaitre toute la complexité, le modèle doit présenter une **façade** qui contient les méthodes qui peuvent être appelées par la vue et par le contrôleur.

Facade pattern

Une **façade** (*facade pattern*) est un patron de conception structurel qui procure une interface offrant un accès simplifié à une librairie, un framework ou à n'importe quel ensemble complexe de classes.
Voir, par exemple, Wikipedia [a] ou refactoring.guru [b]

a. https://fr.wikipedia.org/wiki/Fa%C3%A7ade_(patron_de_conception)
b. https://refactoring.guru/design-patterns/facade

Cette façade contiendra, entre autres, des méthodes afin de :

- ▷ démarrer une partie en choisissant un problème parmi les problèmes connus. Le jeu doit aussi proposer au joueur de choisir un problème au hasard. ;
- ▷ entrer un code ;
- ▷ choisir un validateur ;
- ▷ passer à la manche suivante ;
- ▷ deviner un code (et donc vérifier s'il est correct) ;
- ▷ défaire et refaire un coup (*undo* / *redo*) ;
- ▷ abandonner une partie.

Vous êtes libres de la manière de découper votre projet en classes, *mais* nous vous rappelons que vous êtes évalués sur la qualité de l'architecture de votre code (encapsulation, usage du polymorphisme, etc.). Veillez à bien séparer les classes du jeu proprement dit des classes des interfaces utilisateur et suivez les consignes de votre enseignant.

Command pattern

La gestion de l'*undo* et du *redo* sera implémentée avec le patron de conception commande (*command pattern*).
Voir la fiche associée.

Validateurs

Dans le cadre du projet, il vous est demandé d'implémenter les 22 premiers validateurs définis dans les règles du jeu. Ils sont repris ci-dessous :

N°	Type	Description	Catégories
1	Compare un chiffre à une valeur	Compare le **premier** chiffre du code avec **1**	- Plus petit - Égal - Plus grand
2		Compare le **premier** chiffre avec **3**	
3		Compare le **deuxième** chiffre avec **3**	
4		Compare le **deuxième** chiffre avec **4**	
5	Vérifie la parité d'un chiffre	Vérifie la parité du **premier** chiffre du code	- Pair - Impair
6		Vérifie la parité du **deuxième** chiffre	
7		Vérifie la parité du **troisième** chiffre	
8	Compte une valeur de chiffre	Compte combien de fois la valeur **1** apparaît dans le code	- Aucune fois - Une fois - Deux fois - Trois fois
9		Compte combien de fois la valeur **3** apparaît	
10		Compte combien de fois la valeur **4** apparaît	
11	Compare deux chiffres	Compare le **premier** chiffre du code avec le **deuxième**	- Plus petit - Égal - Plus grand
12		Compare le **premier** chiffre du code avec le **troisième**	
13		Compare le **deuxième** chiffre du code avec le **troisième**	
14	Chiffre extremum	Détermine quel chiffre est **strictement** le plus petit	- Aucun chiffre - 1er chiffre - 2e chiffre - 3e chiffre
15		Détermine quel chiffre est **strictement** le plus grand	
16	Parité la plus fréquente	Détermine s'il y a plus de chiffre **pairs** ou **impairs** dans le code	- Majorité pair - Majorité impair
17	Compte les chiffres pairs	Compte **combien** de chiffres dans le code sont **pairs**	- Aucun - Un chiffre - Deux chiffres - Trois chiffres
18	Parité de la somme des chiffres	Détermine si la **somme** des chiffres du code est **paire** ou **impaire**	- Paire - Impaire
19	Compare la somme de deux chiffres à une valeur	Compare la **somme** du **premier** du **deuxième** chiffre du code avec la valeur **6**	- Plus petite - Égale - Plus grande
20	Nombre de répétitions	Détermine si un chiffre du code se **répète**, et si oui, **combien de fois**	- Pas de doublon - Chiffre doublon - Chiffre triple
21	Chiffre jumeau	Détermine si un chiffre du code se trouve en exactement **deux exemplaires** dans le code (mais pas trois)	- Vrai - Faux
22	Ordre des chiffres	Détermine si **les trois chiffres** du code sont en ordre **croissant ou décroissant**	- Croissant - Décroissant - Aucun

Notez qu'il n'est pas nécessaire de simuler les cartes perforées du jeu physique : puisque l'ordinateur connaît le code secret, la réponse de chaque validateur peut être calculée directement.

Pour rappel, le fichier `known_problems.csv` reprend la liste des problèmes que vous devez implémenter dans le cadre du projet. Dotez-vous des classes nécessaires pour lire

le contenu de ce fichier et créer les objets de modèle appropriés.

Vue console

Afin de vérifier si le modèle est complet, nous vous demandons d'implémenter une application avec une interface en mode console permettant de tester votre modèle et de visualiser l'état d'un jeu.

Votre application suivra l'architecture `MVC` et utilisera le patron de conception `Observateur / Observé`.

Observer pattern

L'**observateur** est un patron de conception comportemental qui permet de mettre en place un mécanisme de souscription pour envoyer des notifications à plusieurs objets, au sujet d'événements concernant les objets qu'ils observent.
Voir la fiche associée.

Model-View-Controller pattern

Model-View-Controller est un modèle d'architecture qui découpe l'application en trois parties : le modèle qui gère les données et leurs manipulations, la vue qui contient la présentation de l'interface graphique et le contrôleur qui contient la logique des actions effectuées par l'utilisateur ou l'utilisatrice.
Voir la fiche associée.

La vue console fonctionne en deux temps. Au démarrage de l'application, l'utilisateur choisit un problème parmi la liste des problèmes existants, ou laisse le jeu choisir un problème au hasard. Il faut aussi permettre au jeu d'afficher la liste des problèmes avec leur difficulté et leur facteur chance afin que l'utilisateur puisse faire son choix.

Une fois la partie lancée, la vue fonctionne ainsi :

- ▷ les validateurs de la machine sont affichés ;
- ▷ les scores (nombre de validateurs vérifiés et nombre de manches) sont affichés ;
- ▷ l'utilisateur peut alors :
  - ▷ entrer un code (uniquement si aucun validateur n'a été vérifié à cette manche) ;
  - ▷ choisir un validateur (dans la limite de 3 validateurs par manche) ;
  - ▷ choisir de passer à la manche suivante (pour tester un nouveau code) ;
  - ▷ vérifier si son code est le bon (ce qui met fin au jeu et lui dit s'il a gagné ou perdu) ;

Ce mode console est placé dans un package spécifique et utilise uniquement la *façade* du modèle.

Vue JavaFX

Implémentez la vue `JavaFX` du jeu.

Votre interface :

- ▷ permet de commencer une partie en choisissant un problème ou en laissant le jeu choisir un problème au hasard ;

- ▷ affiche les validateurs et les scores (nombre de vérifications et nombre de manches) ;
- ▷ permet au joueur d'entrer un code (si aucun validateur n'a été sélectionné durant cette manche) ;
- ▷ permet au joueur de valider son code en sélectionnant un validateur (dans la limite de 3 par manche) ;
- ▷ permet au joueur de passer à la manche suivante ;
- ▷ permet au joueur de tenter de deviner la bonne solution, ce qui met fin au jeu et indique au joueur s'il a gagné ou perdu ;
- ▷ gère les messages d'erreurs lorsqu'une commande incorrecte est effectuée ;
- ▷ gère les *undo* / *redo* ;
- ▷ permet d'abandonner une partie en cours .

Il ne vous est pas demandé d'implémenter une interface graphique pour la prise de notes au sein de l'application comme il existe dans le jeu complet.

Afin de clarifier votre code, pensez à créer des classes séparées pour différents éléments de l'interface, en particulier les validateurs (*don't repeat yourself !*).

Des fichiers images vous sont fournis sur poESI afin de peupler l'interface avec les symboles de robots correspondant aux six emplacements de validateur de la machine (A à F) ainsi que les cartes de chacun des validateurs de 1 à 22.

Bon travail !