v1.0 Projet CodYngame

FILIERE ING1-GI • 2024-2025
AUTEURS E.ANSERMIN – R. GRIGNON
E-MAILS eva.ansermin@cyu.fr – romuald.grignon@cyu.fr

DESCRIPTION DU PROJET

- Créer une application qui permette de résoudre différents exercices de programmation dans différents langages.
- Le but principal est de créer une application graphique dans laquelle on retrouvera plusieurs éléments comme l'énoncé d'un exercice, le choix du langage de programmation, la zone de code avec de la coloration syntaxique et une gestion des indentations, ainsi que la zone de sortie (console) qui permettra de voir si l'utilisateur a réussi à réaliser la solution de l'exercice ou non.
- Ce projet met l'accent sur la gestion des appels système vers les différents compilateurs/interpréteurs, ainsi que sur la base de données des exercices disponibles.

INFORMATIONS GENERALES

- **Taille de l'équipe**
  Ce projet est un travail d'équipe. Il est autorisé de se réunir en groupe de 5 personnes. Si le nombre d'étudiants n'est pas un multiple de 5 et/ou si des étudiants n'arrivent pas à constituer des groupes, c'est aux chargés de projet de statuer sur la formation des groupes. Pensez-donc à anticiper la constitution de vos groupes pour éviter des décisions malheureuses.
- **Démarrage du projet et jalons**
  Vous obtiendrez de plus amples informations quant aux dates précises de rendu, de soutenance, les critères d'évaluation, le contenu du livrable, ..., quand le projet démarrera officiellement.
  Quel que soit le planning initial du projet, vous veillerez à planifier plusieurs rendez-vous d'avancement avec votre chargé(e) de projet. C'est à votre groupe de prendre cette initiative. L'idéal étant de faire un point 1 ou 2 fois par semaine mais cette fréquence est laissée libre en fonction des besoins identifiés avec le tuteur de projet.
- **Versions de l'application**
  Pour éviter d'avoir un projet non fonctionnel à la fin, il vous est demandé d'avoir une version en ligne de commande fonctionnelle. Ceci vous permettra de tester votre modèle de données

indépendamment de l'interface graphique. C'est la version avec l'interface graphique JavaFX qui sera bien entendu évaluée mais dans le cas où certaines fonctionnalités ne seraient pas visibles avec cette interface, vous devez pouvoir présenter toutes les fonctionnalités de votre code Java en ligne de commande.

- **Dépôt de code**
  Vous devrez déposer la totalité des fichiers de votre projet sur un dépôt central Git. Il en existe plusieurs disponibles gratuitement sur des sites web comme ***github.com*** ou ***gitlab.com***. La fréquence des commits sur ce dépôt doit être au minimum de 1 commit / 2 jours.
- **Rapport du projet**
  Un rapport écrit est requis, contenant une brève description de l'équipe et du sujet. Il décrira les différents problèmes rencontrés, les solutions apportées et les résultats. L'idée principale est de montrer comment l'équipe s'est organisée, et quel était le flux de travail appliqué pour atteindre les objectifs du cahier des charges. Le rapport du projet peut être rédigé en français.
  Ce rapport contiendra en plus les éléments techniques suivants : un document de conception UML (diagramme de classe) de votre application, et un document montrant les cas d'utilisations.
- **Démonstration**
  Le jour de la présentation de votre projet, votre code sera exécuté sur la machine de votre chargé(e) de TD. La version utilisée sera la **dernière** fournie sur le dépôt Git **avant** la date de rendu. Même si vous avez une nouvelle version qui corrige des erreurs ou implémente de nouvelles fonctionnalités le jour de la démonstration, c'est bien la version du rendu qui sera utilisée.
  En parallèle, il vous faudra obligatoirement une deuxième machine avec votre application fonctionnelle car une partie de votre groupe aura des modifications de code à faire pendant que l'autre partie fera la présentation/démonstration de votre projet.
- **Organisation de l'équipe**
  Votre projet sera stocké sur un dépôt git (ou un outil similaire) tout au long du projet pour au moins trois raisons :
  - éviter de perdre du travail tout au long du développement de votre application
  - être capable de travailler en équipe efficacement
  - partager vos progrès de développement facilement avec votre chargé(e) de projet.

  De plus il est recommandé de mettre en place un environnement de travail en équipe en utilisant divers outils pour cela (Slack, Trello, Discord, …).

CRITERES GENERAUX

- Le **but principal** du projet est de fournir une **application fonctionnelle** pour l'utilisateur. Le programme doit correspondre à la description en début de document et implémenter toutes les fonctionnalités listées.
- Votre code sera généreusement **commenté**.
- Tous les éléments de **votre code** (variables, fonctions, commentaires) seront écrits **en anglais** obligatoirement.

- Votre code devra être commenté de telle manière que l'on puisse utiliser un outil de génération de documentation automatique de type **JavaDoc**. Un dossier contenant la doc générée par vos soins sera présent dans votre dépôt de code lors de la livraison.
- Votre projet doit être utilisable au clavier ou à la souris en fonction des fonctionnalités nécessaires.
- Votre application ne doit jamais s'interrompre de manière **intempestive** (crash), ou tourner en boucle indéfiniment, quelle que soit la raison.
  Toutes les erreurs doivent être gérées correctement. Il est préférable de d'avoir une application stable avec moins de fonctionnalités plutôt qu'une application contenant toutes les exigences du cahier des charges mais qui plante trop souvent.
  Une application qui se stoppe de manière imprévue à cause d'une exception, par exemple, sera un événement très pénalisant.
- Votre application devra être **modulée** afin de ne pas avoir l'ensemble du code dans un seul et même fichier par exemple. Apportez du soin à la conception de votre projet avant de vous lancer dans le code.
- Le livrable fourni à votre chargé(e) de TD sera simplement l'**URL** de votre **dépôt Git** accessible **publiquement**.

FONCTIONNALITES DU PROJET

- L'application doit permettre de lister tous les exercices disponibles. Si certains exercices ne sont disponibles QUE dans certains langages, il doit alors être possible de filtrer par langage de programmation.
  L'utilisateur doit pouvoir sélectionner n'importe quel exercice qui doit alors s'afficher sur la fenêtre principale
  Dans la vue principale, on retrouvera l'énoncé de l'exercice suffisamment détaillé pour que l'utilisateur comprenne correctement les consignes attendues.
  On retrouvera également une zone de code intégrant si possible une coloration syntaxique des mots-clés du langage, les valeurs numériques et les chaines de caractères.
- Un changement du langage de programmation pourra être fait à n'importe quel moment : cette action effacera la zone de code.
  Quand un langage est choisi, la zone de code contient un code minimal indiquant comment récupérer une ligne depuis l'entrée standard, et comment sur la sortie standard (exemple ***scanf()/printf()*** en ***C***, ***input()/print()*** en ***python***, ***fgets(STDIN)/echo()*** en ***PHP***, …).
  Votre application doit pouvoir exécuter du code dans les langages suivants : ***C***, ***Java***, ***Python*** ; PHP, JavaScript. Bien entendu, tout langage supplémentaire est autorisé.
- Le code écrit par l'utilisateur est soumis à l'aide d'un bouton, l'application compilant/exécutant/interprétant le code en fonction du langage.
  L'exercice doit pouvoir générer plusieurs cas de tests : des cas de tests triviaux pour valider le comportement nominal simple, puis des cas d'erreurs également. Enfin des jeux de tests avec des données aléatoires doivent être lancés pour éviter des programmes utilisateur qui codent les réponses directement.

- Les exercices disponibles, doivent être stockés dans une base de données locale ou des fichiers. Vous êtes libres de définir Le format des données qui vous convient.
  Lors du lancement de l'application, si vous utilisez une base de données, il faudra que votre programme lance automatiquement le serveur si il ne l'est pas déjà.
- Dans Dans la liste des exercices, doit pouvoir s'afficher le nombre de fois que cet exercice a été tenté (nombre de fois où un code a été soumis), et le nombre de fois qu'il a été réussi.

- Les exercices doivent implémenter l'un ou l'autre (OU inclusif) des 2 modes suivants :

  MODE ***STDIN / STDOUT***
  Dans ce mode, l'exercice envoie des données sur l'***entrée standard*** du programme de l'utilisateur avec un format explicitement décrit dans l'énoncé de l'exercice. Le programme de l'utilisateur récupère ces valeurs, réalise le traitement demandé, et renvoie la réponse attendue sur sa ***sortie standard***. Cette sortie est récupérée par l'application qui analyse si les données sont celles attendues.
  Dans tous les cas l'application récupère le résultat de l'exercice pour indiquer si le programme de l'utilisateur est correct ou non.
- Si le programme de l'utilisateur rencontre une erreur (compilation, exécution, faute de segmentation, exception, …) alors le message d'erreur sera affiché sur l'application.
  Si le programme utilisateur s'exécute correctement mais que le résultat n'est pas correct, les données d'entrée seront affichées, ainsi que le résultat donné par l'utilisateur et le résultat attendu, pour que l'utilisateur puisse identifier pus facilement son erreur.
- Dans ce mode, le langage utilisé pour générer les données d'entrée et vérifier les données de sortie est totalement décorrélé du langage utilisé par l'utilisateur. De fait, l'utilisateur peut réaliser cet exercice dans tous les langages disponibles.

  MODE ***INCLUDE***
- Dans ce mode, le code de l'utilisateur sera inclus dans le code de l'exercice. C'est un mode qui est plus adapté à l'écriture d'une fonction/procédure qu'à l'écriture d'un programme principal. De fait les données d'entrée ici seront passées par paramètre.
- L'énoncé de l'exercice doit donc indiquer le nom de la fonction/procédure à écrire, ainsi que les types et l'ordre des paramètres attendus (et leurs rôles fonctionnels), et enfin le type de retour et/ou l'affichage attendus en sortie.
- Dans ce mode, le langage utilisateur ne peut être que le même langage que celui utilisé pour générer les données de l'exercice et vérifier les résultats. Il faut donc que le langage utilisé par l'exercice figure dans les propriétés de ce dernier.

RESSOURCES UTILES

- **Github**
  https://www.github.com
  https://docs.github.com/en/get-started/quickstart/hello-world
- **Patrons de conception**
  https://fr.wikipedia.org/wiki/Patron_de_conception
- **Sérialisation**
  https://fr.wikipedia.org/wiki/Sérialisation
- **Connexion à une base de données**
  https://docs.oracle.com/javase/tutorial/jdbc/basics/connecting.html
- **Appels système depuis Java**
  https://docs.oracle.com/javase/7/docs/api/java/lang/Runtime.html
- **Sites de codage en ligne**
  https://www.codewars.com
  https://www.codingame.com
  https://www.isograd-testingservices.com/FR/solutions-challenges-de-code