Projets informatiques 2018

ENSG - ING1

Présentation générale

Ce projet informatique intervient pour vous permettre de faire une synthèse entre les différents cours suivi cette année en informatique (algorithmie, analyse informatique, base de données et programmation orientée objet). Etant donné qu'il s'agit d'un premier projet, les contraintes techniques sont assez fortes. Il vous sera ainsi demandé :

- de formaliser une analyse du sujet avec les outils UML;
- de mettre en oeuvre les concepts de l'orienté objet;
- de produire une documentation de votre code.

Si votre projet nécessite d'enregistrer des données, vous pourrez utiliser le SGBD de votre choix. Pour info, une base de données SQLite présente l'avantage d'être portable et vous permettrait donc de travailler de n'importe où...

La livraison d'une interface graphique aboutie n'est pas attendue (des graphiques Java2d ou le dessins de formes géométriques simples en svg suffit dans la plupart des cas). Les groupes pour lesquels les fonctionnalités de base de l'application auront été développées pourront s'attaquer à cette problématique en utilisant javax.swing pour réaliser une application bureautique.

Enfin, pour les sujets de simulation, la gestion du “temps réel” n’est pas demandé : une simulation au “tour par tour” suffit.

Organisation

Ces projets s'étendent sur 12 séances de 3h, étalées sur un peu moins de deux mois, auxquelles s'ajouteront une journée et demi de soutenance. Certaines séances seront encadrées, tandis que d'autres s'effectueront en autonomie. Votre présence est bien entendu obligatoire durant l'ensemble des créneaux réservés.

Vous travaillerez en binômes. Un groupe sera composé de trois étudiants, soit **26 groupes** en tout. La composition des groupes est laissée libre mais svp essayez de faire en sorte qu’ils ne soient pas trop déséquilibrés.

13 sujets sont proposés. **Un sujet sera donc choisi par exactement deux groupes**.
Vos délégués nous feront remonter pour le **mardi 03/04 à 11h au plus tard** les binômes ainsi que les sujets choisis, dans un fichier exel ou libreoffice

La séance du 17 mai matin sera réservée aux derniers préparatifs en vue de la soutenance (préparation d'une démonstration, tests en conditions réelles...).

La planning complet des séances de projet est le suivant. Quelques points d'étape importants devant guider votre travail y sont indiqués :

- 03/04 pm : démarrage du projet, début de l'analyse
- 09/04 journée : validation de l'analyse l'après-midi
- 10/04 matin : début des développement
- 10/04 pm **autonomie**
- 04/05 journée
- 14/05 matin- **autonomie**
- 14/05 pm
- 16/05 matin **autonomie**
- 16/05 pm
- 17/05 matin : préparation de la soutenance
- 22/05 journée et 23/05 matin : soutenances

Rendus

- Un **rapport d'analyse** sera à rendre pour le **lundi 07/05 à 23h59** (un point de pénalité par minute de retard). Le rapport d'analyse devra avoir été validé par l'équipe enseignante pour avoir le droit de soutenir. Il s'agira de rappeler les objectifs du projet, de procéder à une reformulation du sujet et d'en faire une modélisation UML avec a minima un diagramme des cas d'utilisation, des classes et d'activités, puis en fonction du sujet de séquence, d'états-transitions, etc. Il ne s'agira pas de lister de manière exhaustive toutes les fonctionnalités de l'application mais de se concentrer sur les plus importantes / complexes.
  Le rapport devra également présenter un planning prévisionnel du projet (GANT)

- Le **code** source de l'application, la **documentation** utilisateur (installation des outils, de l'application, tutorial, readme, quickstart, vidéo ...) et développeur (javadoc). A rendre pour le **17 mai avant 23h59.**

- un **document de synthèse** à rendre pour le **17 mai avant 23h59** (même pénalité de retard que précédemment). Le document de synthèse (5 pages maximum) doit vous permettre de faire le bilan de votre projet (comment vous vous êtes organisés? ce que vous avez réussi à faire? pourquoi vous n'avez pas fait d'autres choses? ce que vous avez appris?). Il doit également comporter un partie plus technique expliquant comment sont organisés vos développements et comment exécuter votre programme.

Les différents rendus seront à déposer sur un dépôt dont l'adresse vous sera communiquée ultérieurement.

Soutenances

Les soutenances auront lieu le **lundi 22 mai toute la journée et le 23 mai matin**. Chaque soutenance durera 20 minutes incluant 5 minutes de démonstration du fonctionnement de votre application. Elle sera suivie de 10 minutes de questions du jury. Tout naturellement, **la présence à l'ensemble des soutenances est obligatoire**.

Evaluation

La livraison d'une application fonctionnelle n'est pas la seule fin du projet. L'analyse du sujet, la modélisation UML, la qualité du code produit (clarté, ré-utilisabilité...) et la documentation comptent pour une part importante de la note finale.

La grille d'évaluation se présente de la manière suivante :

Critères d'évaluation		Barème indicatif
Rapport d'analyse	Analyse du sujet Modélisation UML	4
Fonctionnalités	Nombre et complexité des fonctionnalités réalisées en regard de l'ambition du sujet	4
Code	Mise en oeuvre des concepts de POO Clarté des développement, respect ds conventions	3
Documentation	Exhaustivité de la documentation	3
Présentation	Qualité de l'expression orale, du support Respect des contraintes de temps Réponses aux questions	4
Démonstration	Dynamisme, clarté	2

Quelques conseils

- En matière d'analyse
  - L'analyse du sujet doit aboutir aux spécifications fonctionnelles de l'outil à développer: quels sont les différentes fonctionnalités à implémenter ? Vous pouvez essayer d'évaluer chaque développement (diagramme de GANTT par exemple) et de planifier votre projet.
  - Vous avez le droit, et c'est même bien, de faire évoluer votre modélisation après avoir commencé à développer. Cela étant, si vous vous retrouvez à modifier un diagramme de classe lors de la dernière séance de projet, c'est que vous avez raté quelque chose…
  - Prenez toujours les hypothèses qui vous arrange !
  - Énoncez dans votre rapport d'analyse les hypothèses retenues.
- En matière de gestion de projet
  - Fixez vous des objectifs à court terme et évaluez les : "ce matin, je développe telle fonctionnalité" / "j'ai perdu du temps à cause de problèmes d'imports et finalement, je

n'ai pas encore complètement implémenté cette fonctionnalité". Gardez une trace de cette organisation -> cela s'intègre bien dans le diagramme de GANTT mais ça peut aussi être un document en plus de votre diagramme.
    - Organisez des points d'avancement rapides et réguliers (10min max) pour répondre aux questions suivantes : qu'est ce que j'ai terminé depuis la dernière réunion ? Qu'est ce que j'aurai terminé d'ici la prochaine réunion ? Quels obstacles me retardent ? Vous pouvez inviter l'encadrant à ces réunions en tant que "modérateur".
    - Mettez à l'écrit le bilan de de vos points d'avancement.
- En matière de développement
    - Vous allez probablement écrire beaucoup de code. Respectez des règles simples d'organisation : une classe = un fichier, 20 lignes de code max par méthodes, structure de packages, sous-packages cohérente, etc. Un code bien structuré sera plus facile à comprendre et à débugger.
    - Contrairement à un TP, au cours d'un projet, vous travaillez à plusieurs sur les mêmes fichiers, parfois dans différentes salles de l'école. Pour vous échanger et sauvegarder votre travail, vous vous rendrez compte que les clés USB, les mails ou le lecteur réseau de l'école ne sont pas toujours les plus adaptés. Utiliser une plateforme comme GitHub (<https://github.com>) peut être un bonne alternative. Petit guide GIT http://rogerdudler.github.io/git-guide/
- Pour vos rapports :
    - Si vous abandonnez une solution en cours de route (par exemple parce que vous aurez trouvé une meilleure méthode), ne l'abandonnez pas complètement. Expliquer votre cheminement et montrer pourquoi vous avez privilégié une solution
    - Illustrez vos résultats et expliquez vos choix : une phrase comme "on choisit finalement la première solution, mais avec les bons paramètres" n'explique pas quels sont les paramètres choisis, pourquoi ils le sont, etc.
    - Citez vos sources : vous avez parfaitement le droit de vous inspirer de solutions existantes, voir de les utiliser intégralement (on ne cherche pas à réinventer la roue), mais ayez l'honnêteté intellectuelle de citer les origines des contenus utilisés.
- Pour la présentation :
    - Préparez un scénario pour votre démonstration, ou encore mieux : faites une vidéo. Ouvrir l'application pour montrer qu'il se passe quelque chose lorsque l'on clique sur un bouton n'apporte rien.
    - Ne projetez pas l'intégralité de votre code lors des soutenances. Nous sommes assez grands pour aller regarder par nous même ce qui nous intéresse.

Les sujets

La difficulté indicative d'un sujet est indiquée par des *.

Organisation de festivals de musique **

Vous devez proposer une application permettant de planifier l'organisation de festivals de musique.
Les utilisateurs de l'application seront les organisateurs du festival.
L'application doit permettre gérer les entités intervenant lors d'un festival :

- artistes (invitation, annulation);
- scènes (ajout, affectation de matériel, modification de la configuration);
- concerts.

Lors de la programmation d'un concert (=attribution d'une scène à un artiste à un horaire donné), on veillera à respecter un laps de temps suffisant avec les concerts précédent et suivant.
Ce laps de temps sera d'autant plus important que la configuration de la scène sera modifiée. Si du matériel spécifique est nécessaire, l'application doit permettre de vérier que ce matériel est disponible.
Naturellement, l'application n'est utile que si les informations saisies pour un festival sont enregistré quelque part pour pouvoir être ré-exploitées ultérieurement. On utilisera pour cela une base de données.
En bonus, les fonctionnalités suivantes peuvent être ajoutées à l'application :

- gestion de la billetterie : vérification des places disponibles et envoi de mails avec des informations sur le programme;
- édition de plans ou cartes du festival (format svg).

Tamagotchi *

Un tamagotchi est un animal virtuel qui a des besoins semblables à ceux d'un animal réel : manger, dormir, jouer, se promener. Ce projet vise à réaliser un animal virtuel de ce type.

La communication du tamagotchi avec son maître se fait sous forme de messages textuels. La réalisation d'une interface graphique n'est donc pas attendue au premier abord. La mise à jour en temps réel de l'état du tamagotchi n'est pas non plus attendue initialement mais serait intéressante à implémenter.
L'animal peut effectuer différentes actions. Chacune d'elles se caractérise par l'affichage d'un message.
Le tamagotchi dispose de points de santé, satiété, énergie, bonheur, et éducation qui décroissent progressivement si l'on ne s'occupe plus du tamagotchi. Si les points deviennent trop faibles, le tamagotchi meurt.

Pour remonter les niveaux de santé, faim et sommeil de son animal, le dresseur peut lui envoyer des messages :

- viande : augmente la satiété
- légume : augmente la satiété et la santé
- bonbon : augmente la satiété et diminue la santé
- aspirine : augmente la santé

- dodo : augmente l’énergie (bloque les autres actions)
- éveil : pour mettre fin à une nuit

Pour remonter le niveau d'éducation, le dresseur peut apprendre de nouveaux mots à son animal. Idéalement, on vérifiera qu'il s'agit de mot vraiment nouveaux pour le tamagotchi (ne pas prendre en compte les singuliers/pluriels par exemple).

Pour remonter le niveau de bonheur de son animal, un dresseur peut jouer avec lui. De même que pour les autres options, il lui envoie un message qui déclenche l'ouverture d'une session de jeu. Les jeux à développer sont le blackjack et le morpion.

En bonus: Comment faire vivre votre tamagotchi même lorsque l’application n’est pas ouverte ?

Gestion d’un hôtel **

Ce projet vise à réaliser une application de gestion des chambres d'un hôtel. L'application est destinée à être utilisée par le personnel de l'hôtel.

Les fonctionnalités à développer doivent permettre :

- de réserver une chambre pour un(des) client(s) (nouveau ou ayant déjà séjourné à l'hôtel);
- de préciser les options d'une réservation (petit-déjeuner dans la chambre/au restaurant, dîner, spa, etc.);
- d'annuler une réservation;
- de facturer les clients;
- de comptabiliser des points de fidélité donnant droit à des réductions;
- d'éditer un planning d'occupation des chambres pour une période donnée.

L'application doit également permettre d'identifier les périodes où l'hôtel sera sous-occupé et proposer des offres promotionnelles pour ces périodes.

En plus des classiques chambres, l'hôtel dispose de salles de réunion et est fréquemment utilisé par des entreprises ou associations pour y organiser des séminaires. L'application gagnerait à inclure la gestion des salles de réunion.

Simulation d’une fourmilière ****

On cherche dans ce projet à simuler le comportement de fourmis à la recherche de nourriture. Le monde dans lequel évoluent les fourmis est également peuplé d'autres espèces hostiles aux fourmis.

Pour la partie recherche de nourriture, les fourmis parcourent le terrain et reviennent vers la fourmilière en libérant des phéromones “nourriture” lorsqu'elles en ont trouvé. Guidées par ces phéromones nourriture, les autres fourmis parviennent à identifier la position de la nourriture et petit à petit elles finissent par toutes emprunter le chemin le plus court jusqu'à la nourriture.

Deux types de fourmis peuvent libérer des phéromones nourriture : les fourmis éclaireuses qui ne font qu'identifier la position de la nourriture et les fourmis transporteuses qui en plus rapportent la nourriture à la fourmilière.

Pour les autres espèces présentes dans la simulation, différents types d'ennemis peuvent être définis. Ils se déplacent aléatoirement sur le terrain, et peuvent attaquer et tuer les fourmis lorsqu'ils s'en sont suffisamment approchés. A la rencontre d'un ennemis, les fourmis éclaireuses et transporteuses continuent leur chemin en sécrétant des phéromones “danger”. Des fourmis de type combattantes détectent ces phéromones et s'en servent pour localiser les ennemis et défendre la fourmilière.

Pour la visualisation de la simulation on pourra utiliser Java2d.

Pour améliorer la simulation, on pourra également ajouter des obstacles infranchissables sur le terrain. En arrivant sur un obstacle, les fourmis tournent à gauche ou à droite pour le contourner. Il serait également intéressant d'ajouter d'autres types de fourmis (reine, larves, couveuses, etc.) pour simuler une fourmilière dans sa globalité.

Simulation de la propagation d’un incendie **

Le but de ce projet est de réaliser une application de simulation de la propagation d'un incendie à destination des collectivités publiques. L'incendie évolue en fonction de la nature du terrain et des pompiers, envoyés sur le terrain, peuvent aider à contenir la propagation.

La simulation se déroule sur un environnement fini découpé selon une grille régulière. Les cases de la grille peuvent être de type eau, plaine, forêt ou maison. L'état initial de la simulation est enregistré dans une base de données. La simulation se déroule tour par tour.

A chaque tour les pompiers peuvent se déplacer sur la grille (haut, bas, droite, gauche) ou combattre le feu lorsqu'ils sont situés à côté d'une case en feu. A tout moment, il est possible de sauvegarder l'état de la simulation dans la base pour y revenir ultérieurement.

Règles de la simulation :

- une case de type eau ne brûle jamais;
- une case en feu brûle avec une intensité variant de 1 à 5;
- à chaque tour, l'intensité d'une case en feu augmente de 1;
- l'intensité augmente de 2 par tour pour les maisons;
- à chaque tour, l'incendie d'une case peut se propager jusqu'à n cases voisines, où n est compris entre 0 et l'intensité du feu;
- la probabilité de propagation est doublée pour les cases de forêt (tout en ne pouvant dépasser 8);
- lorsque l'intensité du feu dépasse 5, la case passe à l'état carbonisée : l'intensité redescend à 1 et le feu ne se propage plus;
- un pompier peut diminuer de 2 l'intensité du feu de 3 cases qui lui sont adjacentes.

Vous pourrez utiliser Java2d pour la visualisation de la simulation.
En bonus, les points suivant peuvent être traités :

- les pompiers ne sont pas immortels : attribuez un nombre de points de vie à chaque pompier et chaque tour passé sur une case en feu leur fait perdre des points de vie;
- interface graphique : proposer un affichage de la grille plus convivial à l'aide d'une interface graphique;
- déplacement automatique : proposez des règles de déplacement pour les pompiers qui permettent de passer la simulation en mode automatique et comparez les différentes règles;

- génération automatique du terrain : faites en sorte que la carte de départ soit composée de zones réalistes générées de manière automatique.

Application de calcul d'itinéraires à partir des données de la RATP **

Des millions d'utilisateurs utilisent chaque jour les transports en commun. Les applications de calcul d'itinéraires et de temps de trajets ne tiennent généralement pas compte des correspondances entre le lignes, ou alors appliquent un temps fixe pour toutes les correspondances, ce qui ne reflète pas la réalité du voyageur.

Le but de ce projet est de réaliser une application permettant de calculer un itinéraire dans les transports en commun ferrés franciliens en tenant compte des correspondances.
L'application devra permettre de recherche le meilleur itinéraire entre une station de départ et une station d'arrivée selon différents critères :

- trajet le plus rapide;
- trajet avec le moins de changements ;
- trajet avec le moins de marche à pied.

L'application devra également inclure un module de mise à jour interactive du réseau.
Par exemple lorsqu'une station est fermée pour travaux, il doit être possible de facilement l'indiquer pour que cette information soit prise en compte dans le calcul des itinéraires. Ou encore afin de prendre en compte les difficultés de déplacement du voyageur.

Vous exploitez à cet effet les données de la RATP mises en OpenData (https://opendata.stif.info/page/home/)

Plusieurs solutions s'offrent à vous : mise en base de ces données, construction d'un graphe en Java (http://jung.sourceforge.net/ par exemple), etc.

De même, plusieurs algorithmes de calcul de plus court chemin peuvent être implémentés.
Vous pourrez librement vous inspirer du site suivant : http://www.dericbourg.net/2015/12/10/calcul-ditineraire-a-partir-des-donnees-ratp/

Une visualisation du graphe peut être envisagée en bonus.

**Visualisation d'un MNT et calcul de lignes de niveaux **

Pour rappel, un Modèle Numérique de Terrain (MNT) est une représentation 3D de la surface d'un terrain, créé à partir de données d'altitude. Il est généralement construit à partir d'une grille régulière, en fournissant l'altitude de chaque point de la grille.

A partir d'une grille d'altitudes, l'application à développer doit proposer dans un premier temps une visualisation 2D ou 3D de ce MNT.
Dans un second temps, il faut pouvoir afficher la courbe de niveau passant par une altitude saisie par l'utilisateur.
Puis afficher les courbes de niveau avec un pas choisi par l'utilisateur (tous les 10m, tous les 5m). Enfin, avec un affichage différents pour un second pas tous les 100 mètres par exemple.

Bonus : Vous pourrez construire un MNT aléatoirement ou en charger un depuis différents format.

Gestion d'un arbre généalogique ***

L'objectif de ce projet est le développement d'une application permettant de gérer la généalogie d'une famille.

Votre application doit permettre d'effectuer les tâches suivantes :

- identifier par un numéro unique les individus ;
- enregistrer les principales caractéristiques de la vie civile d'une personne : nom, prénoms, sexe, métier, date et lieu de naissance, date(s) et lieu(x) de mariage, date et lieu de décès ;
- enregistrer les principales caractéristiques de la vie religieuse d'une personne : baptême, mariage, sépulture (dans la tradition catholique) ;
- enregistrer les relations entre les personnes apparentées ou en relation : conjoint, parents, enfants, frère ou sœur, témoins d'actes officiels, parrains et marraines ;
- enregistrer les caractéristiques des sources correspondant à ces informations (actes officiels, témoignages, documents, archives) ;
- effectuer une recherche dans la base des personnes ainsi constituée par nom, prénom, commune, numéro, etc. ;
- Visualiser l'arbre généalogique (sous forme de graphe, de roue généalogique, etc.) ;
- contrôler la cohérence des informations saisies (par exemple : alerte sur la cohérence d'une personne décédée à plus de cent ans ou mère d'un enfant à l'âge de cinquante ans) ;

Gestion des cuisines d'un grand restaurant **

Une chaîne de restaurant parisienne nous demande de réaliser une base de données permettant de stocker des recettes de cuisine et gérant également les stocks des ingrédients que les restaurants de la chaîne possèdent. Chaque recette de cuisine a un nom, une description, un type (Exemples : **Orientale** qui se compose de 3 sous types Proche-Orient, le Moyen-Orient et l'Extrême-Orient. **Européenne** qui se distingue aussi par pays (française, italienne, allemande, …) ou par Europe de l'est et Europe de l'ouest, …) ainsi que la durée de préparation et la durée de cuisson, le nombre de calories par personne, le nombre de parts et le niveau de difficulté : difficile, moyen ou facile. Pour chaque recette, la chaîne veut savoir quels sont les ingrédients nécessaires et la quantité associée à chaque ingrédient. Pour chaque ingrédient possède un nom et le nombre de calories pour 100 grammes de cet ingrédient. Chaque ingrédient a un type, par exemple féculent pour l'ingrédient pomme de terre. Un même ingrédient peut avoir plusieurs conditionnements, par exemple, l'ingrédient farine peut-être stocké sous forme d'un paquet de 1 kg ou de 500 g. Ces deux conditionnements seront considérés comme des produits différents. Pour gérer les stocks des ingrédients, le lieu de stockage des produits dans l'établissement est mémorisé. Chaque rangement est nommé et pour

chaque rangement nous devons savoir quels sont les produits qui y sont stockés. Un même ingrédient peut être stocké dans plusieurs rangements.

Créez la base de données relationnelle de cette application et développer les fonctionnalités souhaitées.
Votre application pourra proposer des menus au restaurant. (avec une nombre d'entrée, de plat et de dessert). Et ce en tenant compte des stocks. Votre application pourra alors afficher une liste de produits en fonction d'un menu souhaité.

Gestion d'une académie **

Une académie souhaite gérer les cours dispensés dans plusieurs collèges. Pour cela, on dispose des renseignements suivants :

- Chaque collège est décrit par un nom, un numéro académique, adresse ;
- Chaque collège possède d'un site Internet ;
- Chaque collège est structuré en départements, qui regroupent chacun des enseignants spécifiques. Parmi ces enseignants, l'un d'eux est responsable du département ;
- Un enseignant se définit par son nom, prénom, adresse, tél, mail, date de prise de fonction et son indice ;
- Un enseignant peut être affecté à 2 collègues, un principal et l'autre secondaire ;
- Chaque enseignant ne dispense qu'une seule matière ;
- Les étudiants suivent quant à eux plusieurs matières et reçoivent une note pour chacune d'elle ;
- Pour chaque étudiant, on veut gérer son nom, prénom, tél, mail, ainsi que son année d'entrée au collège ;
- Une matière peut être enseignée par plusieurs enseignants mais a toujours lieu dans la même salle de cours (chacune ayant un nombre de places déterminé) ;
- On désire pouvoir calculer la moyenne par matière ainsi que par département ;
- On veut également calculer la moyenne générale d'un élève et pouvoir afficher les matières dans lesquelles il n'a pas été noté ;
- Enfin, on doit pouvoir imprimer la fiche signalétique (nom, prénom, tél, mail) d'un enseignant ou d'un élève.

L'académie souhaite calculer la distance entre le domicile d'un enseignant et les collèges d'affectation, à savoir le principal et le secondaire.
Créez la base de données relationnelle de cette application et développez les fonctionnalités souhaitées par l'académie en Java.

Collecte de déchets valorisables ***

Vous devez développer une application permettant à une entreprise d'enlèvement des déchets valorisables d'organiser leur collecte. Les utilisateurs de l'application seront les gestionnaires de l'entreprise.

L'application doit permettre :

- d'attribuer un secteur de collecte à une équipe d'éboueurs;
- de planifier/annuler une opération de collecte;
- d'affecter un lieu de décharge à la suite d'une collecte;
- d'éditer des statistiques sur les collectes (poids, volume, temps, etc.), sur les équipes (carte des tournées, etc.) et sur les zones de collecte (caractéristiques, etc.).

Comme l'application est destinée à gérer les tournées de l'entreprise, elle doit aussi permettre d'ajouter/supprimer des véhicules et éboueurs dans la base de données où sont enregistrées toutes les informations.

Organisation de tournois sportifs ***

Dans ce projet on cherche à créer un logiciel de d'organisation de tournois sportifs. Le logiciel doit être le plus générique possible pour être adapté à un maximum de sports. Il est destiné à être utilisé par les organisateurs des tournois eux-même.
Les fonctionnalités attendues sont les suivantes :

- gestion des listes de participants :(import, ajout, suppression, répartition par catégories, ajout de propriétés personnalisées, etc.);
- organisation de phases qualificatives (génération automatiques de poules, attribution de points en fonction des résultats, etc.);
- organisation de phases éliminatoires (génération automatique d'une grille, petite nale, représentation graphique, etc.).

L'application doit permettre de gérer le maximum de configurations possibles (phase éliminatoire directe en fonction d'un classement initial, plusieurs tours de poules sans sportif "protégé" initialement ni élimination, un seul tour de poule avec des sportifs "protégés" initialement et des éliminés, etc.). Toutes les informations (participants, rencontres, résultats, classements, etc.) doivent pouvoir être restituées à tout moment, ce qui suppose l'enregistrement dans une base de données. Un soin particulier devra être attaché à la modélisation pour éviter les duplications d'informations.
De manière optionnelle, il serait intéressant d'intégrer la gestion temporelle d'un tournoi : nombre de terrains disponibles et durée moyenne d'une rencontre.
De manière optionnelle, ajouter la notion de lieux et utiliser la généricité de votre application pour organiser un tournoi multisports (attention un lieu doit être libre et cela modifie la gestion du calendrier). Un événement comme les championnats d'athlétisme ou comme les JO pourrait être alors suivi.

Simulation de tournoi de cartes à jouer et à collectionner. (JCC) **

Un JCC est un jeux de carte différent d'un jeu classique par la grande variété de cartes éditées. Et par la conception de paquets personnalisés (deck) par les joueurs.
Les tournois récompensent les joueurs capables des meilleures conceptions, de la meilleure stratégie.

Votre application permettra de simuler des parties entre deux joueurs. Et des tournois impliquant plusieurs joueurs.
Dans "ENSG the card game" les joueurs jouent avec des paquets (deck) de 6 cartes choisies parmi un vaste catalogue de carte edité toutes les années.
A chaque tour les joueurs poseront une carte (tirée aléatoirement dans le cas du joueur “ordinateur”). Après 5 tours la partie est finie et on compte les points.

Votre programme construira les decks les plus efficaces en fonctions des cartes disponibles.
Deux types (ou formats) de parties seront gérés :
- Le format OUVERT: toutes les cartes sont disponibles. Certaines cartes seront pourtant interdites dans ce format. Les joueurs choisiront les 6 cartes.
- Le format DRAFT: les adversaires se voient distribuer 7 cartes chacun. Chaque adversaire en conserve une et donne les cartes restantes à un autre adversaire. Avec ces cartes on répète l'opération jusqu'à ce que chaque joueurs ait conservé 6 cartes. La dernière est défaussée.

La modélisation rendra compte des éléments suivants :
Une carte possède un nom, une édition (année de publication), un type (parmi une liste de 6 types), un élément (parmi une liste de 6 éléments). Elle pourra être interdite pour le format OUVERT.
Une carte jouée donne un score (des points) au joueur en fonction de sa rareté, et de sa force de combinaison i.e qui rapporte des points en fonction des cartes du même type ou du même élément qu'elle.
Un deck est consitué de 6 cartes. Chaque carte sera unique.
Un tournoi se verra affronter plusieurs joueurs.
Il mettra à l'honneur une édition (année de publication), un type, et un élément. C'est le métagame.
Pour chacune de ces caractériques de tournoi les joueurs gagneront des points si leur deck contient au moins une carte de cette édition, de ce type, de cet élément. On cumule donc jusqu’à trois fois le bonus.

Votre programme jouera un tournoi (éliminatoire ou championnat) entre 8 joueurs. Donnera le classement et la composition du deck de chaque joueur.
De manière optionnel votre programme permettra à deux joueurs de faire une partie.
De manière optionnel votre programme permettra d’évaluer des règles décrite au niveau de chaque carte (ajout de point, vol de point, suppression d’une carte adverse, possibilité de jouer une 6ième carte, protection contre le prochain effet adverse...)