Comment faire un bon rapport

(pour les noobs)

Que veut un prof ? (ou un autre lecteur…)

Un plan

Le plan de base (modifiable !)

- Introduction
- Présentation générale du projet
- Instance d’exécution
- Explication détaillée des algorithmes importants
- Tests
- Critique du code et perspectives
- Conclusion

Adaptez-le ! Si vous voulez faire d’autres parties pour présenter des choses qui sont importantes pour vous, n’hésitez pas !

Introduction

- On sait déjà c'est quoi le sujet de votre projet, pas besoin de le décrire :p
- Deux ou trois phrases qui décrivent vaguement ce que **vous**, vous avez fait !
- Faites de la publicité ici, donnez au lecteur envie de vous lire, faites un gros titre !

Exemple : "Implémentation du jeu "démineur", avec génération aléatoire de bombes, difficulté du jeu paramétrable, propagation récursive de l'état des cellules, étude de complexité et optimisation des algorithmes." (ok c'est péteux comme exemple… Mais vous avez l'idée.)

But : Donner envie à votre lecteur de vous lire.

Présentation générale

Deux parties importantes :

- Une description brève des classes de votre projet -> Diagramme UML !
- Un mode d'emploi : Elle est où votre classe "main" ? Comment on exécute ? Il faut des paramètres ? (Faites comme si on était débile.)

But : Donner une idée générale de la structure de votre programme, permettre au lecteur de ne pas être trop perdu dans la suite + lui permettre de tester votre programme tout seul.

Instance d'exécution

Après avoir expliqué de manière générale, c'est toujours cool de montrer un exemple d'exécution !

Copiez/collez (si ce n'est pas trop long) la sortie console de votre programme, l'affichage de votre programme + expliquer vite fait ce que le lecteur est censé voir dans l'affichage (décrivez rapidement à quoi correspond chaque élément de votre affichage par exemple).

But : Prouver à votre lecteur que votre projet marche.

Algorithmes utilisés

(Si votre lecteur vous lit toujours jusqu'à là, c'est bon signe, vous avez capté son intérêt. 👍)

C'est la partie la plus **technique** de votre rapport :
Expliquez en détail les algorithmes les plus complexes de votre projet (N'allez pas m'expliquer les trucs débiles comme comment faire des getter/setter… Expliquez les algos qui vous ont pris du temps et des larmes !)

Faites des **schémas** ! SURTOUT PAS DE COPIER/COLLER DE CODE ! (pitié.)

But : Montrer au lecteur que votre projet n'était pas trivial, que vous avez vraiment réfléchi à ce que vous faites et que vous avez apporté de la technicité.

Tests

Quoi de mieux qu'une bonne batterie de tests pour savoir que tout fonctionne ?

Décrivez brièvement les tests que vous avez fait, et si les sorties correspondaient à vos attentes.

Exemple : Test 1 : Test de génération de grands terrains avec N = 1000000. *Résultat :* Le terrain sort de l'écran. :(

But : Rassurer le lecteur que votre programme est robuste et fonctionnel. (ou pas ! Même si le test ne fonctionne pas, vous avez exploré les cas limites, et ça c'est beau.)

Critiques et perspectives

Ne négligez pas cette partie !

Soyez honnête sur votre travail : Dites ce qui fonctionne, **ce qui ne fonctionne pas** (et pourquoi à votre avis), ce que vous auriez pu mieux faire, les décisions que vous avez prises au cours du projet etc… Si vous aviez un algo en tête mais que vous n'avez pas eu le temps de l'implémenter, expliquez-le quand même (c'est arrivé qu'on donne tous les points juste avec l'explication !)

Les perspectives : Qu'est que vous auriez pu faire si vous aviez eu plus de temps et d'énergie ?

But : Montrer votre sens critique et votre esprit scientifico-professionnel au lecteur.

Conclusion

Idem que l'introduction, inutile d'en faire des tonnes, récapitulez brièvement le contenu du rapport + dites ce que ce projet vous a personnellement apporté (travail d'équipe, intérêt pour l'informatique, questionnements quelconques, amusement, ~~désespoir~~) + votre avis.

But : Rassurer le lecteur qu'il ne vous a pas torturé en vous imposant ce projet.

Le Contenu

La BASE : Clair et Concis !

On doit comprendre vite vos idées et vos méthodes !

Do:

- **Des dessins** ! ♥ Schématisez vos idées, mettez des figures à votre document !
- Des phrases courtes, simples !
- Des exemples !
- Un joli rapport avec des titres, de la couleur, une jolie structure… Quelque chose de soigné.
- Une description objective/factuelle/professionnelle du projet (on évite les “je”/“nous”. Exemple : “je me suis dis que c’était mieux de faire cette méthode plutôt que…” -> “Cette méthode fut préférée pour son optimalité/sa simplicité/etc…”)

Don’t :

- Les gros pavés explicatifs, dignes d’un papier administratif. Les phrases longues et alambiquées, qu’il faut relire X fois pour comprendre. Vraiment. Personne n’aime lire ça.
- **Du copier/coller de code**. C’est illisible 99% du temps. A la rigueur, du pseudo-code.
- Zéro plan/structure
- Lé fote dortografe.
- Des plaintes sur le manque de temps, une dispute avec le binôme, le chien qui a mangé le devoir…
- Une description de ce qu’a fait tel ou tel binôme de façon compétitive (souvent pour mettre en valeur que l’autre n’a rien fait), vous êtes une équipe !

Code VS Pseudo-code

Code

```
public void affichage(){
          String[][] tab;
          tab = new String[this.lesCellules[0].length][this.lesCellules.length];
          for(int i = 0; i < this.lesCellules.length; i++){
          for(int j = 0; j < this.lesCellules[0].length; j++){
                    if(this.lesCellules[j][i].getNature().equals("pion")){
                    tab[i][j] = "P";
                    }
                    else if(this.lesCellules[j][i].getNature().equals("mur")){
                    tab[i][j] = "X";
                    }
                    else{
                    tab[i][j] = " ";
                    }
          }
          }

          tab[this.laColo.getPositionNid().get(1)][this.laColo.getPositionNid().get(0)] =
"O";

          for(int e = 0; e < this.laColo.getLesPions().size(); e++ ){

tab[this.laColo.getLesPions().get(e).getPosition().get(1)][this.laColo.getLesPions().get(e
).getPosition().get(0)] = "8";
          }



          //ETC….
```

Pseudo-Code

Affichage():

Pour tout y entre 0 et N :
  Pour tout x entre 0 et N :
    si (x,y) = *“pion”* :
      Ecrire(“P”)
    si (x,y) = *“mur”* :
      Ecrire(“X”)
    sinon :
      Ecrire(“ “)

Lequel est le plus clair ? :)

Code VS Pseudo-code

Code

```
public void affichage(){
        String[][] tab;
        tab = new String[this.lesCellules[0].length][this.lesCellules.length];
        for(int i = 0; i < this.lesCe
        for(int j = 0; j < this.lesCe
                if(this.lesCellules
                tab[i][j] = "P";
                }
                else if(this.lesCe
                tab[i][j] = "X";
                }
                else{
                tab[i][j] = " ";
                }
        }
        }

        tab[this.laColo.getPositionNi
"O";

        for(int e = 0; e < this.laColo

tab[this.laColo.getLesPions().get(e).get
).getPosition().get(0)] = "8";
        }



        //ETC….
```

Pseudo-Code

Affichage():

N :

et N :

” :

:

Il n’y a pas vraiment de “règles” pour écrire du pseudo-code, c’est surtout là pour résumer votre vrai code en focalisant l’attention sur ce qui est vraiment important. Faites attention toutefois à expliquer les variables/paramètres quand ce n’est pas clair.

Lequel est le plus clair ? :)

Les clichés habituels

“Plus y’a de pages, mieux c’est !” FAUX

“Plus ça blablate, plus c’est sérieux !” FAUX

“C’est pro d’utiliser un vocabulaire qu’on ne saurait même pas définir !” FAUX

“Impossible de faire un bon rapport si on a pas un bon projet” FAUX

“Le rapport n’est pas important par rapport au code” FAUX

Le mot de la fin

- Pas tout le monde est égal en TP, certains d'entre vous connaissaient déjà Java, d'autres débutent complètement : **ce n'est pas grave** ! 👍 Ce qui nous intéresse, c'est vos progrès, votre persistance et votre curiosité.

- Je vois beaucoup d'étudiants pour qui le projet ne s'est pas passé comme prévu, et qui rendent un rapport brouillon en se disant "foutu pour foutu…", c'est dommage ! Il y a **forcément** des choses bien dans votre travail, mettez-les en valeur ! Votre rapport, c'est votre vitrine, c'est la seule chose qui va véritablement marquer le correcteur, soignez-le à fond même si vous estimez qu'il y a peu de choses à dire !

- Vous avez passé 5h sur une partie du projet et vous avez abandonné ? Ça nous intéresse énormément ! Décrivez vos approches, pourquoi ça n'a pas marché, ce qui vous a bloqué. On adore comprendre en même temps que vous ce qui a posé problème, c'est même bien plus stimulant que les choses qui marchent (si si !). **Les erreurs sont bien plus pédagogiques après tout.**

Quelques outils

DrawIO

Pour dessiner des figures/diagrammes UML :

→ https://app.diagrams.net/ [disponible sur navigateur !]

Google Drive

Pour partager avec votre binôme le document (nécessite un compte gmail). Permet l'édition partagée. Beaucoup de fonctionnalités qui permettent la correction du rapport, la mise en page etc…

→ Compatible avec DrawIO !

QUELQUES RESSOURCES

C’est quoi un diagramme UML ?

→ https://fr.wikipedia.org/wiki/Diagramme_de_classes

C’est bien expliqué mais je préfère clairement la version anglaise, qui est plus complète et plus illustrée.

→ https://en.wikipedia.org/wiki/Class_diagram

Pour aller plus loin

- **LateX** [se prononce **“latek”** ;)]**:** Langage d’édition de documents (principalement des documents scientifiques). Pour tester, essayer Overleaf (éditeur en ligne usant Latex) : https://fr.overleaf.com/

Latex est parfait quand on a beaucoup de formules mathématiques ou de code/pseudo-code à écrire. Tous les journaux scientifiques utilisent cet outil !

Bon courage et bonne rédaction ! :)