Algorithmique & Programmation

Cas d'étude : Jeu du pendu

yann.secq@univ-lille.fr

ABIDI Sofiene, ALMEIDA COCO Amadeu, BONEVA Iovka, CASTILLON Antoine, DELECROIX Fabien, LEPRETRE Éric, Timothé ROUZÉ, SANTANA MAIA Deise, SECQ Yann

Saison 2023-2024 / R1.01+R1.02 : Initiation au développement

```
Mot secret (nombre d'erreur = 0) : ***********
Veuillez entrer une lettre minuscule de l'alphabet : a
Mot secret (nombre d'erreur = 0) : ******a****
Veuillez entrer une lettre minuscule de l'alphabet : i
Mot secret (nombre d'erreur = 0) : i*****a*i**
Veuillez entrer une lettre minuscule de l'alphabet : e
Mot secret (nombre d'erreur = 1) : i*****a*i**
Veuillez entrer une lettre minuscule de l'alphabet : r
Mot secret (nombre d'erreur = 1) : i***r*a*i**
Veuillez entrer une lettre minuscule de l'alphabet : s
Mot secret (nombre d'erreur = 2) : i***r*a*i**
Veuillez entrer une lettre minuscule de l'alphabet : n
Mot secret (nombre d'erreur = 2) : in**r*a*i*n
Veuillez entrer une lettre minuscule de l'alphabet : o
Mot secret (nombre d'erreur = 2) : in*or*a*ion
Veuillez entrer une lettre minuscule de l'alphabet : f
Mot secret (nombre d'erreur = 2) : infor*a*ion
Veuillez entrer une lettre minuscule de l'alphabet : m
Mot secret (nombre d'erreur = 2) : informa*ion
Veuillez entrer une lettre minuscule de l'alphabet : t
Bravo, vous avez gagné !
```

Jeu du pendu

1. **Quelles informations ?**
2. **Quels traitements ?**
3. **Quels types et structure de données ?**

```
Mot secret (nombre d'erreur = 0) : **********
Veuillez entrer une lettre minuscule de l'alphabet : a
Mot secret (nombre d'erreur = 0) : a*********
Veuillez entrer une lettre minuscule de l'alphabet : e
Mot secret (nombre d'erreur = 0) : a********e
Veuillez entrer une lettre minuscule de l'alphabet : i
Mot secret (nombre d'erreur = 0) : a****i***e
Veuillez entrer une lettre minuscule de l'alphabet : o
Mot secret (nombre d'erreur = 0) : a**o*i***e
Veuillez entrer une lettre minuscule de l'alphabet : t
Mot secret (nombre d'erreur = 0) : a**o*it**e
Veuillez entrer une lettre minuscule de l'alphabet : f
Mot secret (nombre d'erreur = 1) : a**o*it**e
Veuillez entrer une lettre minuscule de l'alphabet : e
Mot secret (nombre d'erreur = 2) : a**o*it**e
Veuillez entrer une lettre minuscule de l'alphabet : e
Mot secret (nombre d'erreur = 3) : a**o*it**e
Veuillez entrer une lettre minuscule de l'alphabet : e
Mot secret (nombre d'erreur = 4) : a**o*it**e
Veuillez entrer une lettre minuscule de l'alphabet : e
Désolé, vous avez perdu ...
```

Jeu du Pendu

- **Informations** : un mot secret, une visibilité associée à chaque lettre du mot secret

- **Traitements** (complexes)
  - Mettre à jour le mot en fonction de la lettre
  - Déterminer si le joueur a gagné

- **Types et structure de données**
  - **`Lettre`** = caractère + visible ou pas (nouveau type)
  - Un mot devient un tableau de **`Lettre`**

Jeu du Pendu

- **Informations** : un mot secret, une visibilité associée à chaque lettre du mot secret

- **Traitements** (complexes)
  - Mettre à jour le mot en fonction de la lettre
  - Déterminer si le joueur a gagné

- **Types et structure de données**
  - **`Lettre`** = caractère + visible ou pas (nouveau type)
  - **Mot** = un tableau de lettres

Type `Lettre`

```
class Lettre {
  char caractere;
  boolean visible = false;
}
```

- **Type** : un champs pour stocker un caractère + un booléen pour indiquer si le caractère a été découvert ou pas (initialisé à `false` car aucune mettre n'est visible au début du jeu)

- **Traitements**
  - Créer une nouvelle valeur de type `Lettre`
  - Afficher une valeur de type `Lettre`
  - Créer un tableau de `Lettre` à partir d'une chaîne
  - Afficher un tableau de `Lettre`

```
Lettre newLettre(char caractere)

String toString(Lettre l)

Lettre[] convertir(String mot)

String toString(Lettre[] mot)
```

Type Lettre

```
class Lettre {
  char caractere;
  boolean visible = false;
}
```

```
Lettre newLettre(char caractere) {
  Lettre l = new Lettre();
  l.caractere = caractere;
  return l;
}

String toString(Lettre l) {
  if (l.visible) {
    return "" + l.caractere;
  }
  return "*";
}

Lettre[] convertir(String mot) {
  Lettre[] lettres = new Lettre[length(mot)];
  for (int idx=0; idx < length(mot); idx++) {
    lettres[idx] = newLettre(charAt(mot, idx));
  }
  return lettres;
}

String toString(Lettre[] mot) {
  String res = "";
  for (int idx=0; idx < length(mot,1); idx++) {
    res = res + toString(mot[idx]);
  }
  return res;
}
```

DS2 passé !
On s'autorise ce type de raccourci maintenant :)

Jeu du Pendu

- **Traitements complexes**

  - **Déterminer si le joueur a gagné**

    - Si une lettre non visible encore présente, la partie n’est pas gagnée, sinon c’est le cas

  - **Mettre à jour le mot en fonction de la lettre**

    - Rendre visible les occurrences de la lettre présente et savoir si au moins une lettre a été modifiée

Jeu du Pendu

- **Déterminer si le joueur a gagné**
  - Si une lettre non visible est encore présente, la partie n'est pas gagnée, sinon c'est le cas

DS2 passé ! On s'autorise ce type de raccourci maintenant :)

```
boolean decouvert(Lettre[] mot) {
  for (int idx=0; idx < length(mot); idx++) {
    if (!mot[idx].visible) {
      return false;
    }
  }
  return true;
}
```

Jeu du Pendu

- **Mettre à jour le mot en fonction de la lettre**
  - Rendre visible les occurrences de la lettre présente et savoir si au moins une lettre a été modifiée

```
boolean miseAJour(Lettre[] mot, char lettre) {
  boolean changement = false;
  for (int idx=0; idx < length(mot); idx++) {
    if (!mot[idx].visible && mot[idx].caractere == lettre) {
      mot[idx].visible = true;
      changement = true;
    }
  }
  return changement;
}
```

Jeu du Pendu

Algorithme principal

- Tableau de mots et choix aléatoire d'un des mots
- Conversion de la chaîne en mot (`Lettre[]`)
- Tant que mot non découvert et `nbErreurs` acceptable
  - Saisir une nouvelle lettre
  - Mettre à jour le mot secret et incrémenter le `nbErreurs` si aucune nouvelle lettre découverte
- Indiquer si la partie est gagnée ou pas

Algorithme principal

```
void algorithm() {
  final int MAX_ERREURS = 5;
  final String[] MOTS = new String[]{"algorithme", "machine",
                                     "information", "langage", "programme"};
  Lettre[] secret = convertir(motAuHasard(MOTS));
  int nbErreurs = 0;
  while (nbErreurs < MAX_ERREURS && !decouvert(secret)) {
    println("Mot secret (nb erreur = "+nbErreurs+"):"+ toString(secret));
    char lettre = readChar(); // contrôle de saisie ?
    if (!miseAJour(secret, lettre)) {
      nbErreurs++;
    }
  }
  if (nbErreurs == MAX_ERREURS) {
    println("Désolé, vous avez perdu ...");
  } else {
    println("Bravo, vous avez gagné !");
  }
}
```

Algorithmique & Programmation

Cas d'étude : LetterPress

yann.secq@univ-lille.fr

ABIDI Sofiene, ALMEIDA COCO Amadeu, BONEVA Iovka, CASTILLON Antoine, DELECROIX Fabien, LEPRETRE Éric, Timothé ROUZÉ, SANTANA MAIA Deise, SECQ Yann

Saison 2023-2024 / R1.01+R1.02 :
Initiation au développement

Analyse, modélisation et programmation

Objectifs du cas d'étude

- Renforcer vos capacités d'analyse et résolutions de problèmes
  - Identifier informations pour la modélisation des types et structures de données
  - Identifier les traitements pour la modélisation algorithmique et la décomposition en fonction
- Se poser les bonnes questions avant de se lancer dans la programmer
- Avancer par petites itérations lors du développement

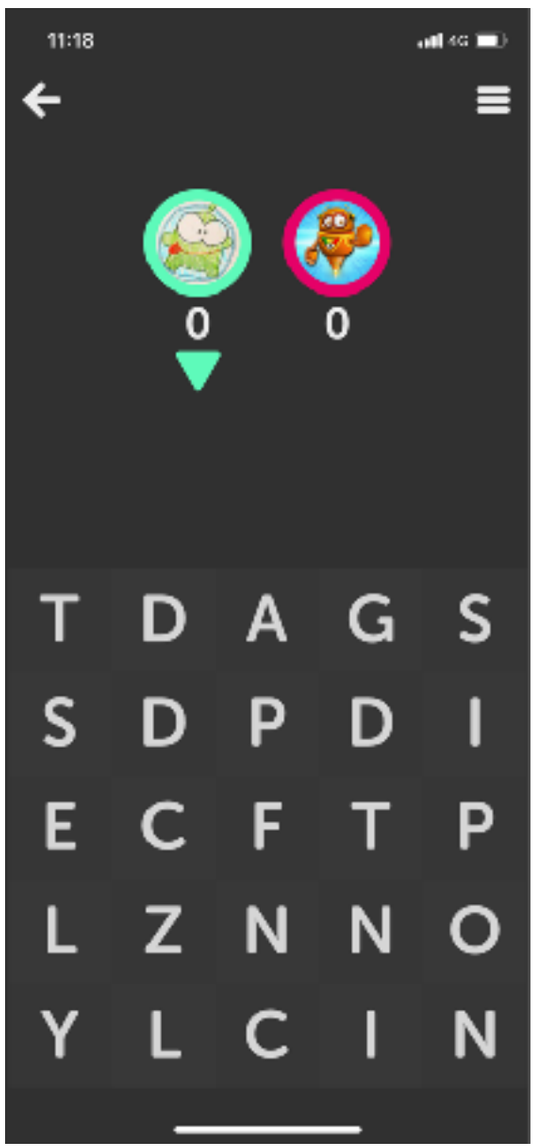
LetterPress

Un jeu de lettres … un peu plus complexe !

- Jeu à deux joueurs s'affrontant sur **un plateau de 5x5 lettres**
- Le but du jeu est de **posséder plus de lettres que l'adversaire** une fois que toutes les lettres de la grille sont utilisées
- Les joueurs alternent les coups **en proposant des mots composés de lettres présentes (n'importe où) sur la grille et « colorent » les lettres utilisées par leur mot**
- La partie s'arrête lorsque toutes les lettres sont capturées et **la joueuse ayant capturé le plus de lettres a gagné**

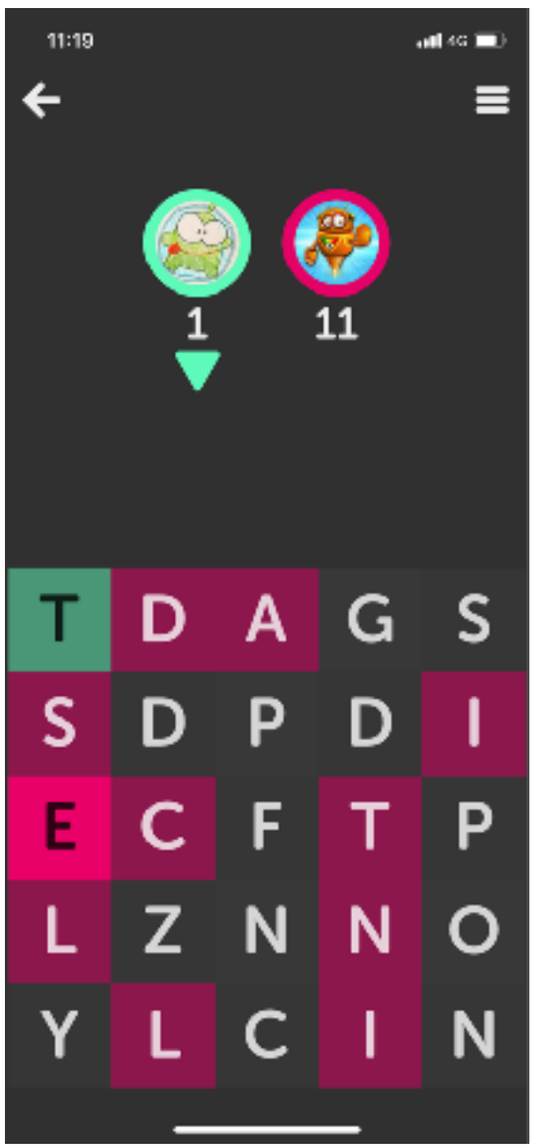
LetterPress

La version originale



Situation initiale : 5x5 lettres « libres » et 2 participant·e·s

J1 joue le mot « attends » (in english ;))

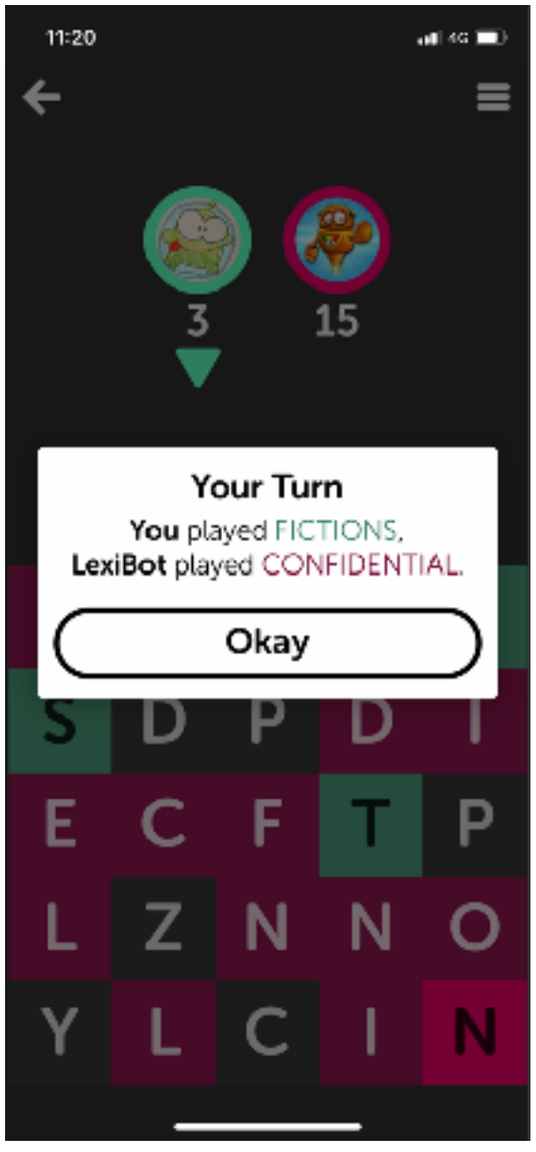


J2 joue le mot « scintillated » ARGL!

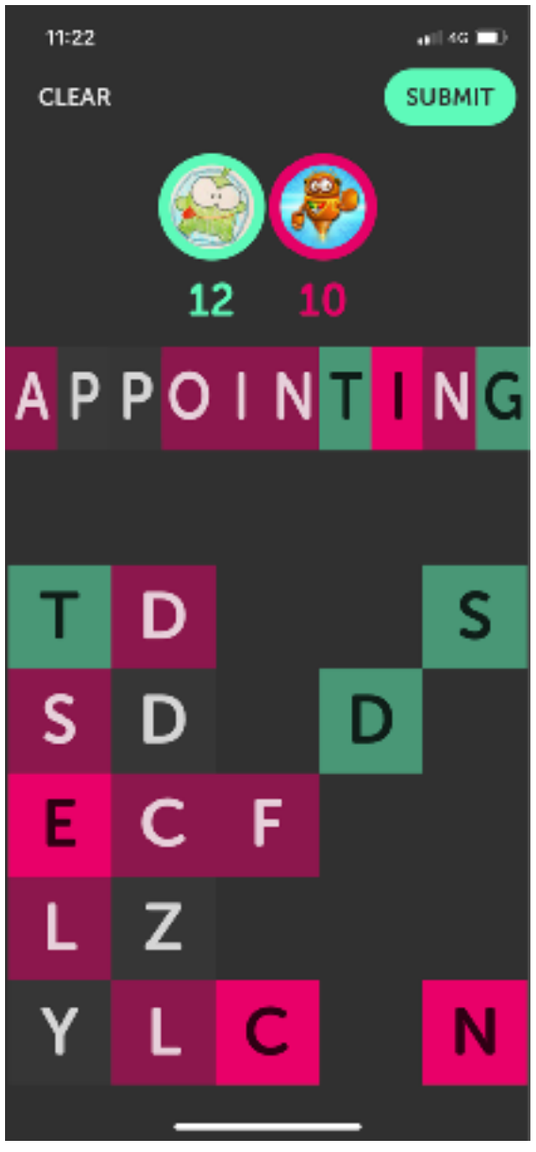
J1 contre, technique Judo ;)

LetterPress

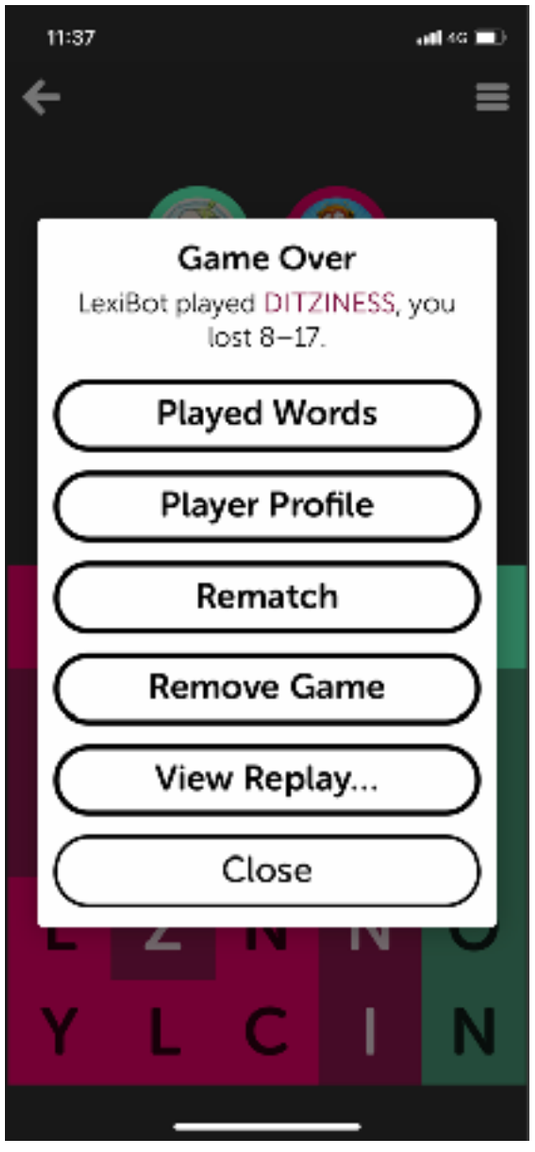
La version originale



La partie se poursuit …



Les lettres à la couleur plus saillantes ne sont pas capturables !



Mince, d’habitude je gagne :o)

Partie conséquente !

LetterPress

Version mode texte

```
[yannsecq@YANNs-MacBook-Pro fouillis % java -cp
 Bienvenue à  L etter  P ress !

 Dictionnaire chargé et contenant 336531 mots.
```

A	B	E	R	S
V	E	T	U	N
S	O	I	D	C
G	A	R	Z	F
E	M	H	D	E

a b c d e
f g h i j
k l m n o
p q r s t
u v w x y

LetterPress

Version mode texte

```
Bienvenue à Letter Press !

Dictionnaire chargé et contenant 336531 mots.

A B E R S      a b c d e
V E T U N      f g h i j
S O I D C      k l m n o
G A R Z F      p q r s t
E M H D E      u v w x y

Alan, veuillez entrer votre mot à l'aide des lettres minuscules : bafg
Vous proposez le mot : B A V E

Score : 4 (Alan) vs. 0 (Ada)

A B E R S      a b c d e
V E T U N      f g h i j
S O I D C      k l m n o
G A R Z F      p q r s t
E M H D E      u v w x y
```

LetterPress

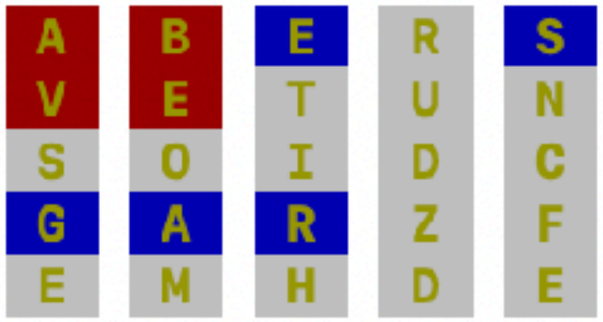
Version mode texte

```
Alan, veuillez entrer votre mot à l'aide des lettres minuscules : bafg
Vous proposez le mot : B A V E

Score : 4 (Alan) vs. 0 (Ada)

A B E R S      a b c d e
V E T U N      f g h i j
S O I D C      k l m n o
G A R Z F      p q r s t
E M H D E      u v w x y

Ada, veuillez entrer votre mot à l'aide des lettres minuscules : zhdqs
Ada, veuillez entrer votre mot à l'aide des lettres minuscules : ddfjq
Vous proposez le mot : R R V N A

Désolé ce mot ne fait pas partie du dictionnaire. Veuillez saisir une nouvelle proposition.
Ada, veuillez entrer votre mot à l'aide des lettres minuscules : pqrce
Vous proposez le mot : G A R E S

Score : 4 (Alan) vs. 5 (Ada)

A B E R S      a b c d e
V E T U N      f g h i j
S O I D C      k l m n o
G A R Z F      p q r s t
E M H D E      u v w x y
```

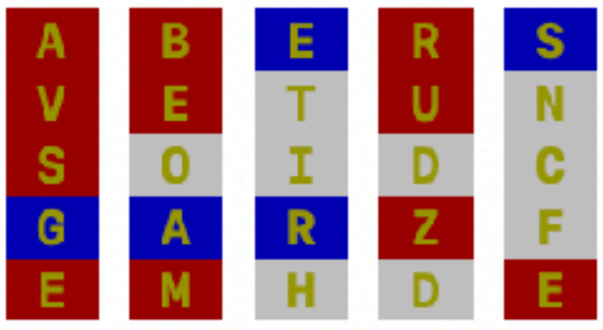
Saisie invalide sur les indices proposés

Saisie invalide pour mot inexistant

```
Score : 4 (Alan) vs. 5 (Ada)
```

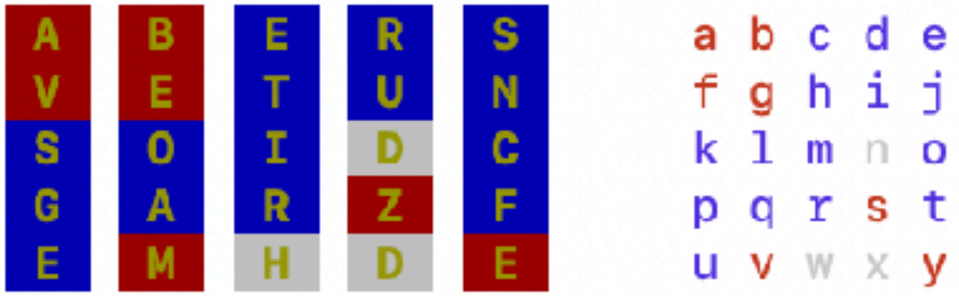



```
Alan, veuillez entrer votre mot à l'aide des lettres minuscules : vukidys
Vous proposez le mot : M E S U R E Z

Score : 11 (Alan) vs. 5 (Ada)
```



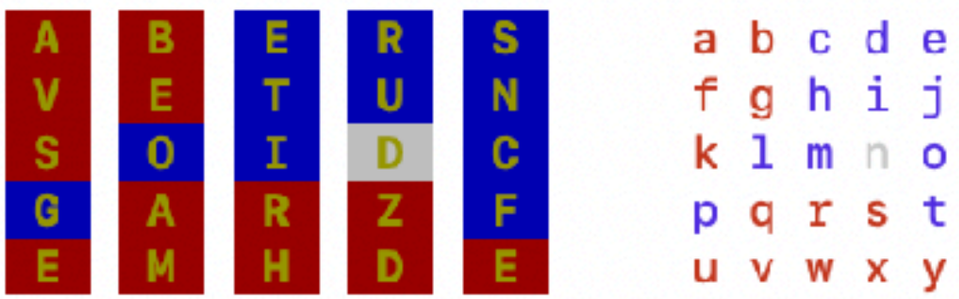

```
Ada, veuillez entrer votre mot à l'aide des lettres minuscules : oljtmhiduk
Vous proposez le mot : C O N F I T U R E S

Score : 7 (Alan) vs. 15 (Ada)
```

Score : 7 (Alan) vs. 15 (Ada)



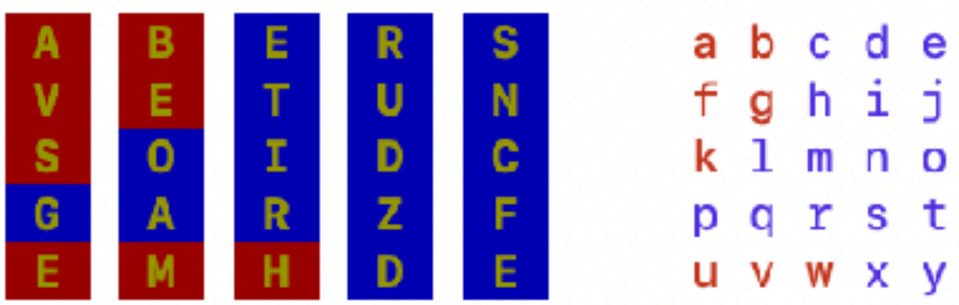
Alan, veuillez entrer votre mot à l'aide des lettres minuscules : wqrxuk
Vous proposez le mot : H A R D E S

Score : 13 (Alan) vs. 11 (Ada)



Ada, veuillez entrer votre mot à l'aide des lettres minuscules : nqrxys
Vous proposez le mot : D A R D E Z

Score : 8 (Alan) vs. 17 (Ada)



Partie terminée ... bravo a Ada qui remporte la partie :)

LetterPress

Analyse des informations

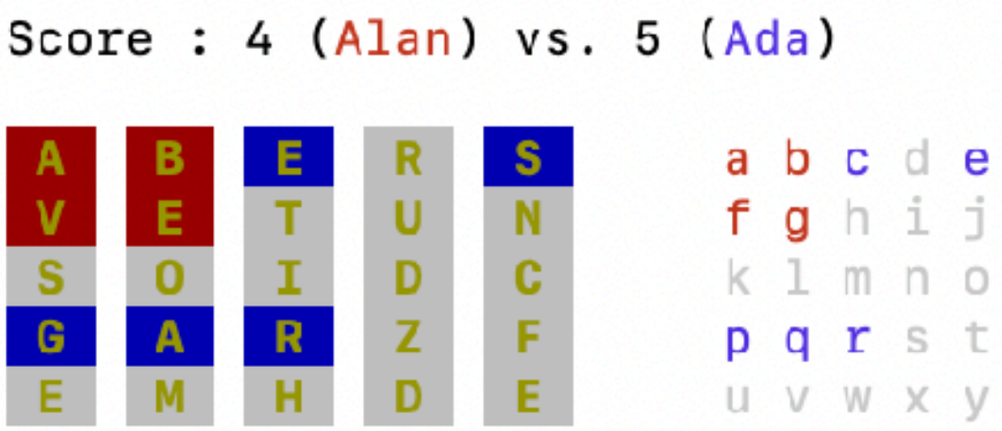
- **Quelles informations sont importantes à modéliser ?**

```
Score : 4 (Alan) vs. 5 (Ada)
```

A	B	E	R	S
V	E	T	U	N
S	O	I	D	C
G	A	R	Z	F
E	M	H	D	E

a b c d e
f g h i j
k l m n o
p q r s t
u v w x y

```
Alan, veuillez entrer votre mot à l'aide des lettres minuscules : vukidys
Vous proposez le mot : M E S U R E Z
```



LetterPress

Analyse des informations

- **Quelles informations sont importantes à modéliser ?**
  - L’existence de 2 **joueuses** ayant un **nom** et une **couleur**
  - La notion de **grille** de **5x5 lettres**
  - Des **lettres** ayant en plus une **couleur** (et **capturables** ?)
  - La notion de **mot**, constitué de **lettres**
  - La notion de **score** (nombre de lettres capturées par un joueur)

LetterPress

Analyse des informations

Une **couleur** (combien ?)

Un type `Couleur` avec :
- ROUGE, BLEU, NOIR

joueuses ayant un **nom** et une **couleur**

Un type `Joueuse` avec :
- un **nom** (String)
- une **couleur** (énumération ?)
- un **score** ? Donnée ou calcul ?
- (un nombre de victoires et de défaites ?)

lettres ayant une **couleur** (et **capturables** ?)

Un type `Lettre` avec :
- une `valeur` (`char`)
- une `couleur` (`Couleur`)
- Capturable : donnée ou calcul ?

mot, constitué de **lettres**

Un type `Mot` avec :
- un tableau de `lettres` (`Lettre[]`)
- a-t-on besoin d'un type ?

grille de **5x5 lettres**

- un tableau de `grille` (`Lettre[][]`)
- a-t-on-besoin d'un type ? Pas sûr

LetterPress

Une première modélisation possible

```
enum Couleur {
  BLEU, ROUGE, NOIR
}
```

```
class Joueur {
  Couleur couleur;
  String nom;
  int lettresCapturees = 0;
}
```

```
class Lettre {
  char valeur;
  Couleur couleur = Couleur.NOIR;
}
```

```
class Mot {
  Lettre[] lettres;
}
```

```
class Grille {
  Lettre[][] lettres = new Lettre[5][5];
}
```

Remarque : ne pas hésiter à utiliser des initialisations par défaut lorsque c'est possible !

LetterPress

Premières réflexions sur les fonctions associées

```
enum Couleur {
  BLEU, ROUGE, NOIR
}
```

Si envisageable d'avoir les couleurs dans le terminal (en jouant avec les codes ANSI !)

```
String codeCouleur(Couleur c)
```

```
class Joueur {
  Couleur couleur;
  String nom;
  int lettresCapturees = 0;
}
```

Les classiques fonctions pour construire une nouvelle valeur de ce type et sa visualisation sous forme de chaîne de caractères

```
Joueur newJoueur(Couleur c, String nom)
String toString(Joueur j)
```

```
class Lettre {
  char valeur;
  Couleur couleur = Couleur.NOIR;
}
```

Les fonctions classiques lors de la création de type

```
Lettre newLettre(char l, Couleur c)
String toString(Lettre l)
```

LetterPress

Premières réflexions sur les fonctions associées

```
class Mot {
  Lettre[] lettres;
}
```

Pas sûr de garder ce type mais bon … cela explicite ce que représente ce tableau

```
Mot newMot(???)
String toString(Mot m)
```

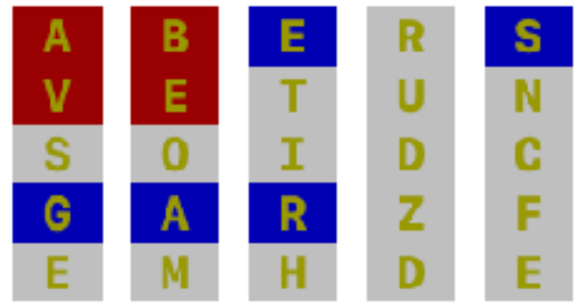
Quand construit-on un mot ? A partir de quelle information ?

```
class Grille {
  Lettre[][] lettres =
     new Lettre[5][5];
}
```

Génération automatique de la grille ?
A partir d'un fichier aussi ?

```
Grille generer()
Grille charger(String nomFichier)
String toString(Grille g)
```

Réfléchir à une représentation avec les couleurs pour les lettres capturées et aussi sur la manière pour la joueuse d'indiquer les lettres constituant son mot ! Plus direct en mode graphique, plus lourd en mode texte avec 25 cases à identifier …



a b c d e
f g h i j
k l m n o
p q r s t
u v w x y

LetterPress

Analyse des traitements

- **Quelles sont les principales étapes de ce logiciel ?**
  - Établir un premier brouillon d'algorithme principal
- **Quels sont les traitements complexes ?**
  - Identifier les fonctionnalités ne s'implémentant pas directement (les `toString` ne font pas partie de cela ;))
- **Quel serait les impacts de ces traitements sur les types et structure de données identifiées pour l'instant ?**
  - Manque-t-il des champs dans certains types ?
  - La/les SDD identifiées sont-elles adaptées aux traitements complexes ?

LetterPress

COMPLEXE
MOYEN
FACILE

Ébauche d’algorithme principal

- **Créer les joueurs/joueuses** (en dur au début)
- **Créer la grille initiale** (**fichier** et/ou **aléatoire**)
- Tant que **toutes les pièces ne sont pas capturées**
  - **Afficher la grille**
  - **Demander au joueur courant le mot qu’il souhaite jouer (nécessite d’identifier individuellement les différentes lettres !)**
  - Vérifier que **le mot est valide** (cf. **dictionnaire**, plus tard)
  - **Mettre à jour la grille avec les lettres capturées**
  - **Changer de jouer et poursuivre la partie**
- **Afficher le/la gagnant·e** une fois toutes les lettres capturées (ajouter la possibilité d’abandon avec saisie spécifique (plus tard !)

LetterPress

Ébauche d'algorithme principal

COMPLEXE
MOYEN
FACILE

- **Créer les joueurs/joueuses**
  - Classique : `newJoueur` + `toString`

```
Joueur newJoueur(String nom,
                 Couleur couleur)
String toString(Joueur j)
```

- **Créer la grille initiale** (**fichier** et/ou **aléatoire**)
  - D'abord fichier (plus simple)

Nommage plus significatif que `newGrille`

```
Grille charger(String nomFichier)
Grille generer()
```

- Tant que **toutes les pièces ne sont pas capturées**
  - Boucle du jeu : `while (!fini(grille))` → `boolean fini(Grille g)`
- **Afficher la grille**
  - Penser `toString` !

```
println(toString(grille));
```

→

```
String toString(Grille g)
```

LetterPress

Ébauche d'algorithme principal

COMPLEXE
MOYEN
FACILE

- **Demander au joueur courant le mot qu'il souhaite jouer (nécessite d'identifier individuellement les différentes lettres !)**

```
Mot saisir(Grille g,
           String message)
```

- Vérifier que **le mot est valide** (cf. **dictionnaire**, plus tard)

```
boolean valide(String mot,
               String[] dictionnaire)
```

- **Mettre à jour la grille avec les lettres capturées**

```
void capturer(Grille g, Mot m, Joueur j)
```

- **Changer de jouer et poursuivre la partie**

```
Joueur changer(Joueur actuel,
               Joueur joueur1,
               Joueur joueur2)
```

- **Afficher le/la gagnant·e** une fois toutes les lettres capturées (ajouter la possibilité d'abandon avec saisie spécifique (plus tard !)

```
Joueur gagnant(Joueur j1,
               Joueur j2)
```