DOCUMENTATION : 0/20

- Je sais décrire le contexte de mon application, pour que n'importe qui soit capable de comprendre à quoi elle sert. [sur 2 points]
  PREUVE :
  Première partie de la documentation : "Contexte"
  => 0/2

- Je sais faire un diagramme de cas d'utilisation pour mettre en avant les différentes fonctionnalités de mon application. [sur 5 points]
  PREUVE :
  Deuxième partie de la documentation : "Diagramme de cas d'utilisation"
  => 0/5

- Je sais concevoir un diagramme UML de qualité représentant mon application. [sur 7 points]
  PREUVE :
  Troisième partie de la documentation : "Diagramme UML"
  => 0/7

Je sais décrire un diagramme UML en mettant en valeur et en justifiant les éléments essentiels. [sur 6 points]
***PREUVE :**
Troisième partie de la documentation : "Diagramme UML" (en dessous du diagramme)
=> 0/6

Programmation : 0/20

- Je sais utiliser les Intent pour faire communiquer deux activités. [sur 1 point]
  PREUVE :
  L'activité de la page d'accueil (StartActivity) crée un intent afin de se diriger sur la page souhaitée lorsque le joueur clique sur un des boutons.
  => 0/1

- Je sais développer en utilisant le SDK le plus bas possible. [sur 1 point]
  PREUVE :
  Afin que tous les appareils Android puissent utiliser notre application, nous avons fait en sorte d'utiliser l'API 15.
  => 0/1

- Je sais distinguer mes ressources en utilisant les qualifier. [sur 1 point]
  PREUVE :
  Pas encore
  => 0/1

- Je sais modifier le manifeste de l'application en fonction de mes besoins. [sur 1 point]
  PREUVE :
  Nous avons modifier le manifest de l'application afin d'y ajouter des permissions pour le Bluetooth et afin d'afficher automatiquement le clavier lorsque l'on arrive sur les pages jeu en singleplayer ou multiplayer.
  => 0/1

- Je sais faire des vues xml en utilisant layouts et composants adéquats. [sur 1 point]
  PREUVE :
  Voir la page main_view.xml
  ImageView pour afficher l'image, TextView pour afficher les lettres jouées, Button pour proposer un mot, EditText pour taper les lettres à essayer, LinearLayout permettant de contenir des text_view.xml (représentant le mot avec les lettres trouvées et celles à trouver) qui sont des TextView ayant des valeurs

différentes de celles par défaut.
=> 0/1

- Je sais coder proprement mes activités, en m'assurant qu'elles ne font que relayer les évènements. [sur 1 point]
  PREUVE :
  L'activité StartActivity permet seulement de récupérer les évènements de clic sur un des boutons et de rediriger le joueur sur l'activité voulue.
  => 0/1

- Je sais coder une application en ayant un véritable métier. [sur 2 points]
  PREUVE :
  Notre métier représente le jeu du pendu (JeuPendu). Il permet de modifier notre jeu mais ne s'occupe pas de la vue.
  => 0/2

- Je sais parfaitement séparer vue et modèle. [sur 1 point]
  PREUVE :
  Les éléments de notre vue (activités) sont séparés des éléments de notre modèle qui sont eux dans un dossier modèle et qui ne modifies pas la vue mais seulement le Bluetooth.
  => 0/1

- Je maîtrise le cycle de vie de mon application. [sur 1 point]
  PREUVE :
  Lorsque l'on ferme notre application (sans la détruire), on que l'on retourne notre appareil, appel de la méthode onSaveInstanceState() qui sauvegarde les éléments de notre jeu afin de les réassignés ensuite.
  => 0/1

- Je sais utiliser le findViewById à bon escient. [sur 1 point]
  PREUVE :
  MainActivity (jeu un joueur) : Utilisation pour modifier certains éléments de la vue comme la liste des lettres tapées, les lettres trouvées, ou pour la gestion d'évènements sur certains éléments comme le clic sur le bouton pour proposer un mot.
  ex : tv_typed_letters = findViewById(R.id.tv_typed_letters);
  => 0/1

- Je sais gérer les permissions dynamiques de mon application. [sur 1 point]
  PREUVE :
  Dans le Bluetooth (BluetoothActivity permettant de se connecter à un autre appareil), vérification de la permission de l'application à utiliser le bluetooth, et si elle n'a pas le droit on demande la permission.

```
int permissionCheck =
this.checkSelfPermission("Manifest.permission.ACCESS_FINE_LOCATION");
permissionCheck +=
this.checkSelfPermission("Manifest.permission.ACCESS_COARSE_LOCATION");
if (permissionCheck != 0) {

    this.requestPermissions(new String[]{Manifest.permission.ACCESS_FINE_LOCATION,
Manifest.permission.ACCESS_COARSE_LOCATION}, 1001);
}
```

=> 0/1

- Je sais gérer la persistance légère de mon application. [sur 1 point]
  PREUVE :
  On sauvegarde l'état du jeu puis on le restore :

  @Override

```
protected void onSaveInstanceState(Bundle outState) {
    if (jeu.getWord() != null) {
        outState.putString("currentWord", jeu.getWord());
        outState.putInt("nbLettersFound", jeu.getFound());
        outState.putInt("nbError", jeu.getError());
        outState.putStringArrayList("lettersList", jeu.getListOfLettersString());
    }
    super.onSaveInstanceState(outState);
}

@Override
protected void onRestoreInstanceState(Bundle savedInstanceState) {
    super.onRestoreInstanceState(savedInstanceState);
    jeu.setInstanceState(savedInstanceState.getString("currentWord"),
savedInstanceState.getInt("nbLettersFound"),
savedInstanceState.getInt("nbError"),
savedInstanceState.getStringArrayList("lettersList"));
}

@Override
protected void onCreate(@Nullable Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    jeu = new JeuPendu(this.getApplicationContext());
    if (savedInstanceState != null) {
        jeu.setInstanceState(savedInstanceState.getString("currentWord"),
savedInstanceState.getInt("nbLettersFound"),
savedInstanceState.getInt("nbError"),
savedInstanceState.getStringArrayList("lettersList"));
    }
```

=> 0/1

- Je sais gérer la persistance profonde de mon application. [sur 1 point]
  PREUVE :
  on sauvegarde tous les mots dans un fichier txt qui est dans le fichier assets/pendu_liste.txt
  => 0/1

- Je sais afficher une collection de données. [sur 1 point]
  PREUVE :
  Il faut utiliser un inflater.
  Exemple : affichage de la liste des appareils Bluetooth découvrables :

```
public DeviceListAdapter(Context context, int tvResourceId,
ArrayList<BluetoothDevice> devices){
    super(context, tvResourceId,devices);
    this.mDevices = devices;
    mLayoutInflater = (LayoutInflater)
context.getSystemService(Context.LAYOUT_INFLATER_SERVICE);
    mViewResourceId = tvResourceId;
}
```

  Dans l'activité de la vue de connexion Bluetooth :

```
mDeviceListAdapter = new DeviceListAdapter(context, R.layout.device_adapter_view,
mBTDevices);
lvNewDevices.setAdapter(mDeviceListAdapter);
```

  => 0/1

- Je sais coder mon propre adaptateur. [sur 2 points]
  PREUVE :
  Utilisation de ArrayAdapter pour l'adaptation d'un array par exemple (DeviceListAdapter)
  => 0/2

- Je maîtrise l’usage des fragments. [sur 2 points]
  PREUVE :
  Pas encore
  => 0/2

- Je maîtrise l’utilisation de Git. [sur 1 point]
  PREUVE :
  Nous avons utilisé Git tout au long du projet afin de toujours garder la dernière version du projet en ligne et de pouvoir travailler de notre côté à tout moment.
  => 0/1

Application : 0/20

- Mon application présente un intérêt à être publié sur le store. [sur 5 points]
  PREUVE :
  De nombreux jeux de pendu existes déjà sur le store. Cependant, nous n'en avons trouvé aucun permettant de jouer avec vos amis sur plusieurs appareils en même temps grâce à une connexion Bluetooth.
  => 0/5

- Mon application fonctionne de manière à être utilisée par le public. [sur 5 points]
  PREUVE :
  C'est un jeu simple que tout le monde connaît et peut utiliser que ce soit seul ou à plusieurs sans perdre de temps à chercher les fonctionnalités car elles sont intuitives.
  => 0/5

- Mon application utilise des contraintes spécifiées lors du choix du projet. [sur 5 points]
  PREUVE :
  Nous utilisons une connexion Bluetooth afin de jouer en multijoueur et cela fait partie des contraintes spécifiées lors du choix du projet.
  => 0/5

- Mon application utilise les contraintes à bon escient. [sur 5 points]
  PREUVE :
  La connexion n'est utilisée que lorsque le joueur souhaite jouer en multijoueur avec ses amis et nous n'avons pas ajoutés d'autres contraintes comme l'accéléromètre, le gyroscope, l'appareil photo ou la localisation car nous n'en avions aucunement besoin pour notre application et cela l'aurait rendu plus lourde et donc moins simple d'utilisation alors que nous cherchons à atteindre le plus de personnes possibles comme nous l'avons fait avec le choix du SDK le plus bas possible.
  => 0/5