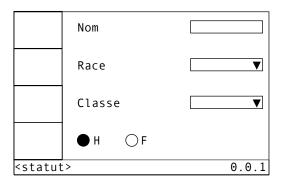
Remarque : remettre individuellement sous forme d'une archive **ZIP** les codes sources commentés de l'application au **plus tard le dimanche** qui suit la fin de la séance (3 heures).

## I. Interface graphique

On souhaite réaliser une interface graphique permettant de remplir une fiche personnage d'un Jeu de Rôle.



Cette interface sera composée :

- d'une <u>barre d'outils</u> (à gauche) contenant une grille de <u>4 boutons</u> représentant chacun une action (sous forme de texte et/ou d'image) : nouveau, charger, enregistrer, enregistrer sous...);
- d'une <u>barre d'état</u> (en bas), avec un <u>label vide</u> à gauche et la <u>version</u> de programme à droite:
- d'un <u>panneau d'information</u> (au centre), avec <u>une zone de saisie</u> (nom), <u>des listes</u> <u>déroulantes</u> (race, classe), <u>des boutons radio</u> (sexe).

Compléter les fichiers java pour obtenir cette interface. Les classes CharacterSheetListener et CharacterSheetModel ne sont pas à compléter pour le moment.

## II. Réaction aux évènements

On souhaite maintenant pouvoir définir des réactions aux différents évènements utilisateurs. En reprenant les exemples vus en cours, et en veillant à bien séparer le code de présentation des composants de celui permettant la réaction aux actions utilisateurs (c'est ici que la classe CharacterSheetListener est à compléter), afficher l'action effectuée dans le champs statut de la barre d'état. On ne s'intéresse ici qu'aux actions des boutons de la barre d'outils.

## III. Aller plus loin

Pour cette partie aucune nouvelle classe n'a besoin d'être créée.

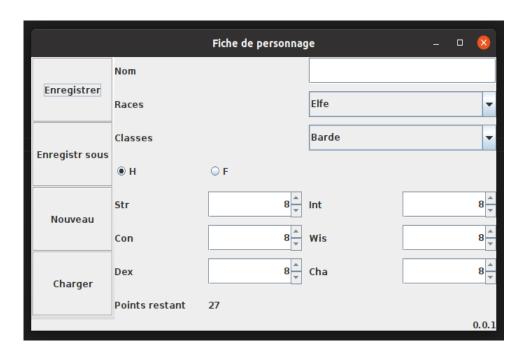
On souhaite maintenant ajouter plus d'éléments à notre fiche, notamment les caractéristiques. Le système de caractéristiques obéit à certaines règles :

- Chaque caractéristique a une valeur initiale de 8 et doit être comprise entre 8 et 15.
- Le total de points à dépenser est de 27.

Les caractéristiques à ajouter sont la Force (Str), la Constitution (Con), la Dextérité (Dex), l'Intelligence (Int), la Sagesse (Wis) et le Charisme (Cha). Vous veillerez à respecter le design pattern MVC en ajoutant les caractéristiques au Model.

On utilisera des listes déroulantes (JSpinner¹) pour modifier les valeurs.

a) Coder tout d'abord la nouvelle interface.



- **b)** Dans le modèle, il y a un attribut **changeSupport** et deux méthodes : **addPropertyChangeListener** et **removePropertyChangeListener**. À l'aide de la Javadoc<sup>2,3</sup>, expliquer le rôle de ces éléments et comment les utiliser.
- c) Ajouter finalement les réactions aux événements pour être en mesure d'appliquer les règles, en modifiant le contrôleur et en utilisant l'attribut changeSupport et PropertyChangeListener entre la vue et le modèle.

<sup>1</sup> https://docs.oracle.com/javase/8/docs/api/javax/swing/JSpinner.html

<sup>2</sup> https://docs.oracle.com/javase/8/docs/api/java/beans/PropertyChangeListener.html

<sup>3</sup> https://docs.oracle.com/javase/8/docs/api/java/beans/PropertyChangeSupport.html