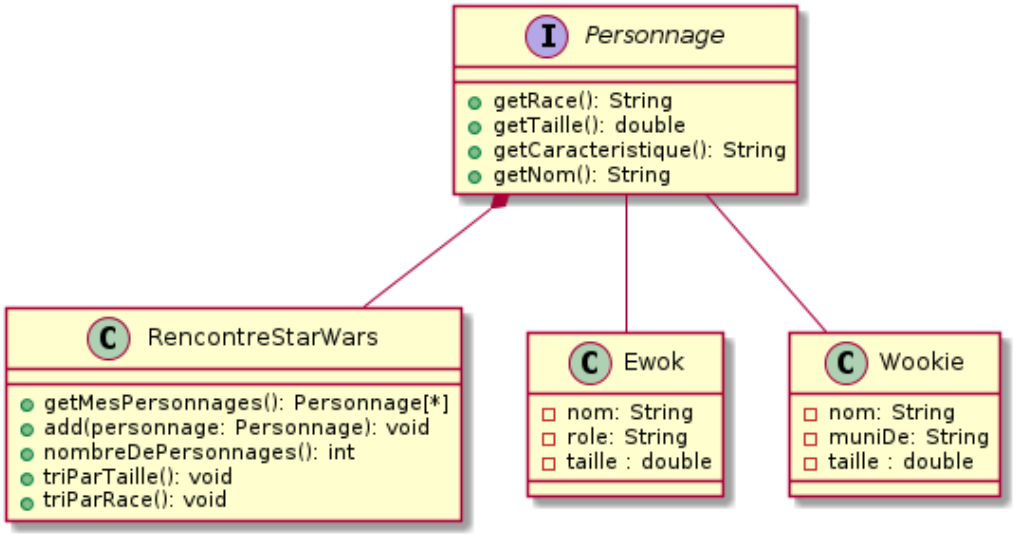
TP9

Interface (encore un peu)

Personnages de Star Wars

On veut *modéliser* des personnages de la Guerre des Etoiles. On vous donne le schéma UML suivant :



Wookie

Originaire de Kashyyyk, les Wookies sont d'immenses créatures recouvertes de fourrure. Ils pouvaient vivre jusqu'à quatre cent ans. Ils possèdent une faculté de guérison assez exceptionnelle, ce qui leur permet de se remettre assez vite d'une blessure. Les Wookies sont armés de griffes. Ces dernières ne servent qu'à grimper dans les impressionnants arbres de Kashyyk. Se battre avec elles est interdit. Tout Wookie se battant avec ses griffes, surtout contre un de ses congénères, est considéré comme fou. Les Wookies sont des individus très loyaux. Une très grande place est accordée à l'honneur et au respect de la nature. Les Wookies sont incapables de parler le Basic, même s'ils le comprennent. C'est pourquoi ils doivent être accompagnés d'un droïde traducteur s'ils veulent être compris des autres espèces.

Quelques Wookies connus : Burryaga Agaburry, Attichitcuk, Chewbacca, ...

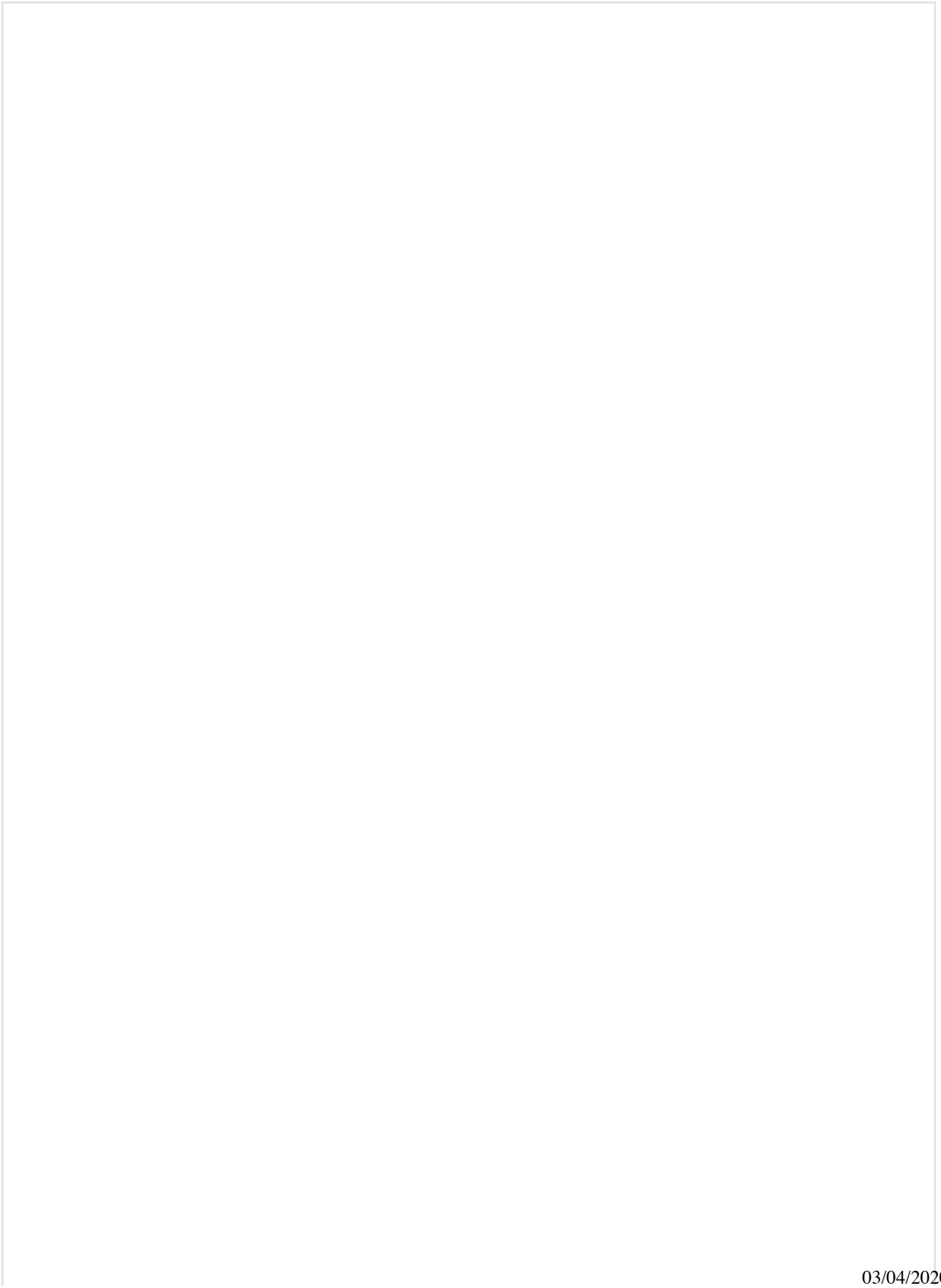
Ewok

Petites créatures avoisinant le mètre de hauteur, les Ewoks ressemblent à de petites peluches recouvertes de fourrure. Ils vivent sur Endor en tribu regroupées autour d'un chef. Bien que très curieux et courageux, les Ewoks se méfient de ce qui leur est inconnu. Ils sont également très superstitieux Les Ewoks ne disposent que d'une technologie primitive basée sur des constructions

en bois, feuille, tissu.

Nous retiendrons ici que les Wookies sont immenses alors que les Ewok mesurent à peine 1 mètre. Les Wookies sont munis d'arme spécifique (leur caractéristique) et les Ewoks ont une position dans leur village (leur caractéristique).

Quelques Ewoks connus : Wicket Warrick, Chef Chirpa, Graak ....



```
public class ExecutableSW{
  public static void main(String[] args) {
    Personnage p1 = new Ewok("Wicket Warrick",0.80, "éclaireur");
    System.out.println(p1);
    // Doit afficher :
    // Je suis un Ewok, je m'appelle Wicket Warrick, je mesure 0.80 mètre(s)
    // et je suis éclaireur.

    Personnage p2 = new Wookie("Attichitcuk", 2.29, "une arbalète");
    System.out.println(p2);
    // Doit afficher :
    // Je suis un Wookie, je m'appelle Attichitcuk, je mesure 2.29 mètre(s)
    // et porte une arbalète.

    Personnage p3 = new Wookie(" Burryaga Agaburry", 2.21, "un sabre laser");
    System.out.println(p3);
    // Doit afficher :
    // Je suis un Wookie, je m'appelle  Burryaga Agaburry,
    // je mesure 2.21 mètre(s) et porte un sabre laser.

    Personnage p4 = new Ewok("Chef Chirpa",1.0, "chef");
    System.out.println(p4);
    // Doit afficher :
    // Je suis un Ewok, je m'appelle Chef Chirpa, je mesure 1 mètre(s)
    // et je suis chef.

    RencontreStarWars starWars = new  RencontreStarWars();
    starWars.add(p1);
    starWars.add(p2);
    starWars.add(p3);
    starWars.add(p4);
    System.out.println(starWars)
    // Doit afficher :
    // StarWars [Je suis un Ewok, je m'appelle Wicket Warrick, je mesure 0.8
    // mètre(s) et je suis un éclaireur , Je suis un Wookie, je m'appelle
    // Attichitcuk, je mesure 2.29 mètre(s) et porte une arbalète , Je suis
    // un Wookie, je m'appelle  Burryaga Agaburry, je mesure 2.21 mètre(s)
    // et porte un sabre laser , Je suis un Ewok, je m'appelle Chef Chirpa,
    // je mesure 1.0 mètre(s) et je suis un chef ]

    starWars.trierParRace();
    System.out.println("Par race "+ starWars);
    // doit afficher :
    // [Je suis un Ewok, je m'appelle Wicket Warrick, je mesure 0.8 mètre(s)
    // et je suis un éclaireur , Je suis un Ewok, je m'appelle Chef Chirpa,
    // je mesure 1.0 mètre(s) et je suis un chef , Je suis un Wookie,
    // je m'appelle Attichitcuk, je mesure 2.29 mètre(s) et porte une
    // arbalète , Je suis un Wookie, je m'appelle  Burryaga Agaburry, je
    // mesure 2.21 mètre(s) et porte un sabre laser ]


    starWars.trierParTaille();
    System.out.println("Par taille " +starWars);
    // Doit afficher :
    // StarWars [Je suis un Ewok, je m'appelle Wicket Warrick, je mesure 0.8
    // mètre(s) et je suis un éclaireur , Je suis un Ewok, je m'appelle Chef
    // Chirpa, je mesure 1.0 mètre(s) et je suis un chef , Je suis un Wookie,
    // je m'appelle  Burryaga Agaburry, je mesure 2.21 mètre(s) et porte un
    // sabre laser , Je suis un Wookie, je m'appelle Attichitcuk, je mesure
    // 2.29 mètre(s) et porte une arbalète ]

   }
  }
```

1. Ecrire les classes qui permettent de répondre à la classe `ExecutableSW` et fournisse les affichages proposés.

Exceptions

Rappels de TD sur les Exceptions

On vous donne la classe Tableau suivante :

```
import java.util.List;
import java.util.ArrayList;
import java.lang.Math;
public class Tableau {
   private List<Integer> tab;
   public Tableau () {
       tab = new ArrayList<Integer>();
   }
   public void remplir() {
       int nb = (int)(Math.random()* 10);
       for(int i = 0; i< nb; ++i)
           tab.add((int)(Math.random()* 50));

   }
   @Override
   public String toString() {return tab.toString();}
   public List<Integer> getTableau() {return this.tab;}
}
```

1. Dans un exécutable, utilisez la méthode min de Collections pour extraire le minimum du tableau. Traitez l'exception NoSuchElementException de façon à afficher "Il n'y a pas de minimum: le tableau est vide !" dans le cas où Collections.min lève l'exception NoSuchElementException.
2. Ajoutez une méthode get qui prend en entrée un indice (entier) et renvoie l'élément à cet indice si il existe et lève NoSuchElementException sinon. Testez cette méthode et traitez le cas d'exception dans votre exécutable.
3. Ajoutez une méthode getMax à votre classe. Vous l'implémenterez en faisant appel à la méthode max de Collections. Que se passe-t-il en cas d'exception ?

Exception personnalisée

Dans cet exercice, on va reproduire nous même le comportement de la méthode min de Collections en écrivant notre propre exception 'PasDeTelElementException'.

1. Écrivez une classe PasDeTelElementException qui étend Exception.
2. Définissez une méthode **getMin()** déterminant la valeur du minimum du tableau. Vous devrez lever une exception PasDeTelElementException si le tableau est vide.
3. Associez à cette classe une classe exécutable permettant de déterminer la valeur du minimum d'une liste en utilisant la classe Tableau et en gérant le cas où celui-ci est vide.

L'Entier plus petit plus grand qu'un entier donné

On suppose l'existence du tableau suivant : **[99, 38, 10, 49, 27, 74, 81, 60, 87]**. On désire rechercher dans ce tableau le plus petit entier plus grand qu'un entier donné et l'on veut que cet entier appartienne au tableau.

- Ainsi 90 n'appartient pas au tableau (traitement d'erreur).
- Mais 74 appartient au tableau, le plus petit entier plus grand que 74 est 81
- De meme 99 appartient au tableau mais il n'existe pas de plus petit élément plus grand que 99.

Définissez une classe Bibliotheque possédant une méthode **plusPetitPlusGrand** prenant un tableau d'entiers (ArrayList<Integer>) et un entier et répondant aux critères ci-dessus.

Animaux et zoo (Bonus)

Si vous avez le temps

1. On vous demande de définir une class **Animal** possédant deux attributs : un nom et un booléen indiquant s'il est ou non blessé.
2. Définissez une classe **Zoo** ayant comme attributs un nom et un tableau d'animaux.
3. Définissez une méthode *accueillir* permettant d'accueillir des animaux dans ce zoo.
4. Définissez une méthode **soigner** prenant un Animal en argument et permettant de soigner cet animal. Attention vouxs envisagerez les cas où l'animal n'est pas dans le zoo et celui où l'animal n'est pas blessé.
5. Définissez une classe **ExecZoo** contenant l'exécutable de vos classes.