Exercise 1

Soit la classe abstraite Person, définissant la méthode abstraite greet. Créer une classe Student et Professor qui implémente la méthode greet telle que un Student dit "Good morning Professor!" et un Professor dit "Good morning students!".

```
object Main extends App{

  abstract class Person{
    def greet()
  }

  class Student extends Person{
    \\ Implement here
  }

  class Professor extends Person{
    \\ Implement here
  }
}
var p = new Student()
p.greet()
}
```

Exercise 2

Soit la solution de l'exercice 2 du TP précédent. On veut empêcher un utilisateur d'instancier les classes Animal et Mammal. Effectuer les changements nécessaires.

Exercise 3

1. Soit la classe Pet, créer une sous-classe Dog et définir la méthode 'sayHello' qui doit afficher l'attribut 'hello' suivit de 'Je suis un <this.getClass.getName>'
Créer une instance de la classe Dog et vérifier la valeur retournée par la méthode 'sayHello'

```
object Main extends App {

abstract class Pet {
  var age: Int = 0
  val hello: String = s"Hello!"
  val greeting: String = s"I like to play with you!"
  def sayHello()
  override def toString = s"$hello, $greeting"
}

\\ Implement here

}
```

2. Créer une nouvelle classe fille Cat héritant de Pet qui définie également la valeur de l'attribut abstrait 'hello'.
Cette classe re-définie l'attribut 'greeting' et ré-implémente la méthode 'sayHello'.

Exercise 4

```
abstract class Pet (name: String) {
  var age: Int = 0
  val hello: String = s”Hello”
  val greeting: String = s"I like to play with you!"
  def sayHello = { println(hello) }
  override def toString = s"$hello, $greeting"
}

trait catchball {
  \\ define trait here
}
```

On veut maintenant définir une particularités à l'aide d’un trait.

1. Définir un nouveau trait 'catchball' contenant une méthode catchBall.

2. Etendre la classe Dog avec ce nouveau trait en implémentant la méthode catchBall afin d’afficher "catch the ball!".

Exercise 5

Dans cet exercice, vous devez implémenter cinq classes représentant des types de vehicules. La première classe Vehicle, abstraite, définie les méthodes de communs. Deux sous-classes, Car et Bike, hérite de la classe Vehicle. Pour finir, les voitures définis par la classe Car sont séparés en deux groupes : electricCar et thermalCar.

1. Implémenter la classe abstraite Vehicle qui contient la méthode abstraite license et fuel. La méthode fuel affiche le comportement par défaut “Je n’ai pas besoin de fuel pour rouler”.

2. Implémenter la classe Bike, qui hérite de Vehicle. Implémenter la méthode license qui affiche “Pas de licence requise.”.

3. Implémenter la classe abstraite Car, qui hérite de Vehicle.
Implémenter la méthode licence, de telle sorte qu'elle affiche "Il faut un permis de conduire".

4. Implémenter les deux classes electricCar et thermalCar, héritant de Car. La méthode fuel doit afficher pour une instance de la classe thermalCar “J’utilise du fuel pour rouler”.