Programmation Fonctionnelle (2024-2025) – TD 1

Exercice 1

Soit le code suivant :

```scala
object Td1 {
  case class Information(message: String, country: String, timestamp: LocalDateTime, tags: List[String], price: Int)

  def parseInformation(line: String): Information = {
    val parts = line.split("--")
    val timestamp = parts(0).trim
    val country = parts(1).trim
    val message = parts(2).trim
    val tags = parts(3).split(",").map(_.trim).toList
    val price = parts(4).split("€")(0).trim.toInt
    Information(message, country, LocalDateTime.parse(timestamp, DateTimeFormatter.ISO_LOCAL_DATE_TIME), tags, price)
  }

  @main
  def test(): Unit = {
    val rawInformation = List(
      "2017-05-08T14:39:06 -- France -- This is an information -- tag1 -- 2€",
      "2017-05-08T14:49:06 -- UK -- This is another information -- tag1,tag2 -- 4€",
      "2018-05-10T14:39:06 -- France -- This is a newer information  -- tag3 -- 8€"
    )
    val information = rawInformation.map(parseInformation(_))
    // la suite
  }
}
```

L'annotation `@main` indique que le code à exécuter est dans la méthode `test`. Vous pouvez lancer le code avec votre IDE, en ligne de commande avec `scala Td1` ou encore avec `sbt run` si vous avez correctement configuré le build.

A chaque question / item, vous commencerez par donner le type (notation du cours) puis la signature Scala, et enfin vous ferez quelques tests.

Partie 1.a Implémentez les fonctionnalités suivantes :

- `informationTimestamps` qui permet d'avoir la liste des timestamps des informations.
- `informationTag` qui permet de connaître les informations contenant un tag
- `informationTagOneOf` qui permet de connaître les informations contenant un ou plusieurs tags pris dans une liste
- `informationMessageSuchThat` qui permet de connaître les informations qui satisfont une condition donnée portant sur le message
- `informationCountry` qui permet de connaître les informations d'un pays donné

Effectuez les tests suivants :

- liste des timestamps des informations
- liste des informations contenant le mot `newer`
- liste des informations commençant par le mot `This`
- liste des informations avec le tag `tag1`
- liste des informations avec le tag `tag2` ou `tag3`
- prix total des informations concernant la France

Partie 1.b On souhaite généraliser ce que l'on vient de faire.

- Pour cela commencez par définir une méthode `selection` qui permet de sélectionner les informations en lui donnant (a) une fonction permettant de sélectionner le champ d'information concerné (ex: `i ⇒ i.tags` pour les tags, `i ⇒ i.message` pour le message, etc.), (b) une fonction permettant de conditionner la valeur de ce champ (ex: `_.startsWith("Foo")`) et (c) la liste des informations.
- Définissez des variables `messageSelecteur`, `tagsSelecteur`, `countrySelecteur` et `timestampSelecteur` correspondant respectivement aux selecteurs sur `message`, `tags`, `country` et `timestamp`. Cela doit se faire en appliquant partiellement `selection`. Quel est le type de ces variables ? Notez que l'introduction d'un paramètre de généricité se fait comme suit :

```scala
def génèrePaire[T,U](f: Int ⇒ T)(g: Int ⇒ U)(x: Int): (T,U) = (f(x), g(x))
val divisible2et3 = génèrePaire(i ⇒ i%2 == 0)(i ⇒ i%3 == 0)
```

```
List(1,2,3,4,5,6).map(divisible2et3).foreach(println)
→ (false, false), (true, false), ..., (true, true)
```

- Utilisez ces variables pour obtenir les mêmes informations qu'à l'exercice 1.a, ainsi que toutes les informations de l'année 2017.

Exercice 2

Faites la même chose en Java.