PROGRAMMATION AVANCÉE ET STRUCTURES DE DONNÉES

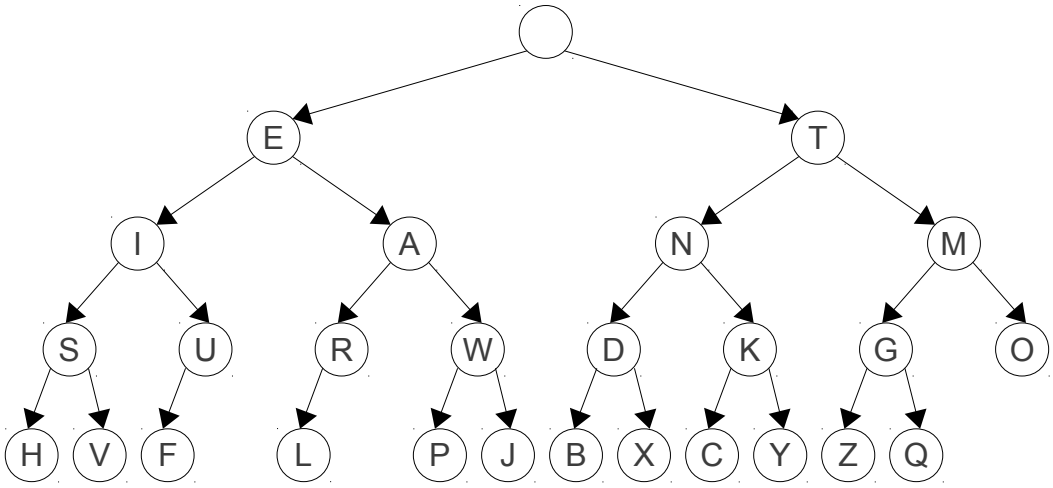
Session 2 — Juin 2015

Tous documents autorisés — durée 2 heures
Le barème indiqué est indicatif et peut légèrement varier lors de la correction. Préciser pour chaque définition de méthode la classe dans laquelle cette définition doit être présente.

Codage Morse

Le code Morse, notamment utilisé dans les communications télégraphiques dès la fin du 19ème siècle, consiste à coder des caractères par une succession d'impulsions, celles-ci pouvant être courtes ou longues. Les impulsions courtes sont notées `'.'` et les impulsions longues sont notées `'-'`. Il existe en fait plusieurs codes Morse, puisqu'il suffit de décider à quelle séquence d'impulsions correspond un caractère. L'un de ces codes Morse est standardisé, c'est le code Morse international.

Étant donné qu'il n'y a que deux types d'impulsions, un arbre binaire peut être utilisé pour décoder un message exprimé dans un codage Morse. Cet arbre binaire est construit de telle sorte qu'un parcours depuis la racine correspond à une séquence de `'.'` et de `'-'` : depuis un nœud, un `'.'` correspond à un parcours vers le fils gauche alors qu'un `'-'` correspond à un parcours vers le fils droit. Ci-dessous figure l'arbre contenant les caractères de l'alphabet dans le code Morse international : on peut y voir, par exemple, que `"--"` est le code de `'M'`, `"---"` est le code de `'O'`, `".-."` est le code de `'R'` et `"."` est le code de `'E'`.



Afin de représenter un tel arbre, nous définissons les 3 classes `NoeudAbstrait`, `Vide` et `Noeud`. `NoeudAbstrait` est une classe abstraite, super-classe des deux autres classes. `Vide` est une classe « singleton », c'est-à-dire qu'elle admet une instance unique (référencée par l'attribut `instance`) qui représente un arbre vide (l'instance unique de la classe `Vide` n'est pas représentée dans la figure ci-dessus pour des raisons de simplification). Les instances de la classe `Noeud` représentent chacune un nœud non vide, grâce aux attributs `gauche` et `droit` qui référencent respectivement le sous-arbre gauche et droit d'un nœud, et à l'attribut `caractere` qui contient le `Character` porté par le nœud, ou éventuellement `null` si le nœud ne porte aucun `Character`.

La classe `Morse` permet quant à elle de représenter un codage Morse en se basant notamment sur les 3 classes `NoeudAbstrait`, `Vide` et `Noeud`. Enfin, la classe `MorseException` est définie pour dénoter une exception spécifique au codage/décodage Morse.

1 Initialisations des nœuds (2,5 points)

Donner la définition des constructeurs des 3 classes `NoeudAbstrait`, `Vide` et `Noeud`. Le constructeur de la classe `Noeud` doit lever une `IllegalArgumentException` si l'un de ses paramètres de type `NoeudAbstrait` est `null`.

Donner la définition de la méthode `getInstance()` de la classe `Vide`.

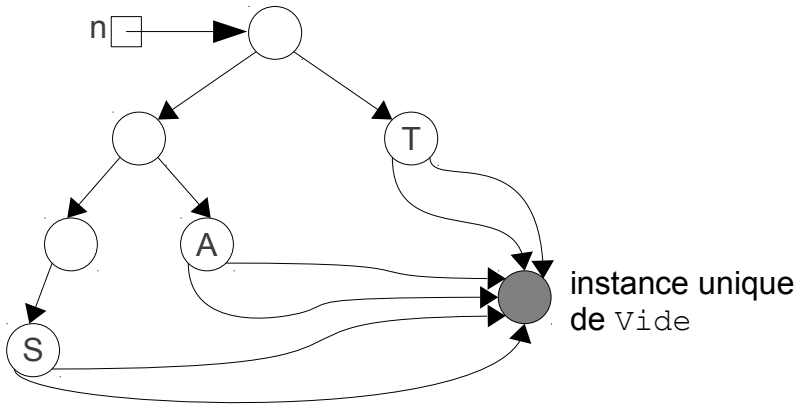
2 Ajout dans les nœuds (4 points)

Donner la définition des méthodes `ajout(String chemin, Character c)` des classes `Vide` et `Noeud`. Ces méthodes retournent la racine de l'arbre, une fois l'ajout effectué. Le chemin est toujours créé dans l'arbre, en laissant des nœuds ayant `null` comme `Character` si nécessaire.

Si `chemin` contient d'autres caractères que `'.'` ou `'-'`, une exception `MorseException` ayant pour message `"chemin invalide"` doit être levée. Si le chemin existe déjà dans l'arbre et conduit à un nœud portant un `Character` non `null`, une exception `MorseException` ayant pour message `"chemin déjà utilisé"` doit être levée.

À titre d'exemple, la séquence d'instructions suivante produit l'arbre ci-dessous (les nœuds représentés sans caractère portent un `Character` `null`).

```
NoeudAbstrait n = Vide.getInstance();
n = n.ajout(".-", 'A');
n = n.ajout("-", 'T');
n = n.ajout("...", 'S');
```



3 Décodage à l'aide des nœuds (2,5 points)

Donner la définition des méthodes `decode(String chemin)` des classes `Vide` et `Noeud`. Ces méthodes retournent le `Character` décodé grâce à l'arbre à partir de la `String` fournie en paramètre. Si la `String` fournie en paramètre ne contient pas que des `'.'` et/ou des `'-'`, une exception `MorseException` ayant pour message `"chemin non valide"` doit être levée. Si la `String` fournie en paramètre ne correspond pas à un chemin permettant d'atteindre un nœud portant un `Character` non `null`, une exception `MorseException` ayant pour message `"chemin introuvable"` doit être levée.

À titre d'exemples sur l'arbre représenté dans la question 2 :

- `n.decode(".-")` retourne une instance de `Character` correspondant à `'A'`,
- `n.decode("..")` résulte en la levée d'une `MorseException` ayant pour message `"chemin introuvable"`,
- `n.decode("--")` résulte en la levée d'une `MorseException` ayant pour message `"chemin introuvable"`,
- `n.decode(".A")` résulte en la levée d'une `MorseException` ayant pour message `"chemin non valide"`.

4 Nombre de nœuds sans caractère (1,5 points)

Donner la définition des méthodes `nbSansCaractere()` des classes `Vide` et `Noeud`. Ces méthodes retournent le nombre de nœuds ne portant pas de `Character` non `null` dans l'arbre..

5 Classe `Morse` (9,5 points)

Une instance de la classe `Morse` dénote un code Morse. L'attribut `arbre` référence un arbre semblable à celui donné dans la question 2 et l'attribut `dictionnaire` référence une `Map` associant à chaque caractère son expression en code Morse dans une `String` : `dictionnaire` est donc utile pour le codage et `arbre` pour le décodage. Il est essentiel de s'assurer que tous les `Character` présents dans `dictionnaire` sont également présents dans `arbre` et inversement.

5.1 Constructeur sans paramètres (0,5 point)

Donner la définition du constructeur sans paramètres de la classe `Morse`.

5.2 Ajout (2 points)

Donner la définition de la méthode `ajout(String chemin, Character c)` de la classe `Morse` qui permet d'ajouter un nouveau caractère au code associé à son `chemin` qui est son expression en Morse. Si l'un des paramètres est `null`, une exception `IllegalArgumentException` doit être levée. Si le caractère est déjà présent dans le code, une exception `MorseException` ayant pour message `"ajout impossible : caractère déjà présent"` doit être levée.

5.3 Constructeur avec paramètre (1 point)

Donner la définition du constructeur à un paramètre de la classe `Morse`. Ce constructeur permet d'initialiser un code Morse comptant comme caractères ceux présents dans la `Map` fournie en paramètre dans laquelle ils sont associés à leur expression en Morse.

5.4 Nombre de chemins vides (0,5 point)

Donner la définition de la méthode `nbCheminsVides()` de la classe `Morse`, qui retourne le nombre de chemins permettant d'atteindre dans arbre un nœud sans caractère. On considère que la chaîne vide `""` n'est pas un chemin valide.

5.5 Codage (2,5 points)

Donner la définition de la méthode `code(String message)` de la classe `Morse`, qui retourne le codage en Morse de la `String` fournie en paramètre. Les codages de chaque caractère de `message` dans la `String` résultat sont séparés par un espace `' '`. Les espaces `' '` présents dans `message` sont reportés tels quels dans la `String` résultat. Si `message` contient, à l'exception de l'espace `' '`, un caractère absent du code Morse, une exception `MorseException` ayant pour message `"caractère inconnu"` doit être levée.

Par exemple, pour un code ayant pour arbre celui représenté dans la question 2, `code("SAS T")` retourne `"… .- …  - "` et `code("AN")` résulte en la levée d'une `MorseException` ayant pour message `"caractère inconnu"`.

5.6 Décodage (3 points)

Donner la définition de la méthode `decode(String message)` de la classe `Morse`, qui retourne le décodage d'un message codé en Morse. La `String message` peut contenir plusieurs espaces `' '` consécutifs mais chaque expression Morse d'un caractère est séparée de la suivante par un espace `' '`.

Par exemple, pour un code ayant pour arbre celui représenté dans la question 2, `decode(" … .- …  - ")` retourne `" SAS T"`.

7 Annexes : Rappels, indications et squelettes de classe

- Le constructeur `Character(char c)` permet de construire un `Character` dont la valeur correspond à `c`. La méthode d'instance `charValue()` de la classe `Character` permet d'obtenir la valeur d'un `Character` considérée sous la forme d'un `char`.

- `IllegalArgumentException` est une sous-classe de `RunTimeException`.

```
package morse;
public abstract class NoeudAbstrait {
	public NoeudAbstrait() {…}

	public abstract NoeudAbstrait ajout(String chemin, Character c);
	public abstract Character decode(String chemin);
	public abstract int nbSansCaracteres();
}
```

```
package morse;
public class Vide extends NoeudAbstrait {
	private static Vide instance = new Vide();

	private Vide() {…}

	public static Vide getInstance() {…}

	public NoeudAbstrait ajout(String chemin, Character c) {…}

	public Character decode(String chemin) {…}

	public int nbSansCaracteres() {…}
}
```

```
package morse;
public class Noeud extends NoeudAbstrait {
	private NoeudAbstrait gauche, droit;
	private Character caractere;

	public Noeud(Character c, NoeudAbstrait g, NoeudAbstrait d) {…}

	public NoeudAbstrait ajout(String chemin, Character c) {…}

	public Character decode(String chemin) {…}

	public int nbSansCaracteres() {…}
}
```

```
package morse;
public class MorseException extends RuntimeException {
	public MorseException() {}

	public MorseException(String message) {
		super(message);
	}
}
```

```
package morse;
import java.util.Map;
public class Morse {
	private NoeudAbstrait arbre;
	private Map<Character, String> dictionnaire;

	public Morse() {…}
	public Morse(Map<? extends Character, ? extends String> m) {…}

	public void ajout(String chemin, Character c) {…}

	public int nbCheminsVides() {…}

	public String code(String message) {…}

	public String decode(String message) {…}
}
```