Faculté des Sciences
Aix*Marseille Université

Année universitaire 2017/2018

Site : ☒ Luminy ☐ St-Charles ☐ St-Jérôme ☐ Cht-Gombert ☒ Aix-Montperrin ☐ Aubagne-SATIS
Sujet de : ☒ 1er semestre ☐ 2ème semestre ☐ Session 2 Durée de l'épreuve : 2h
Examen de : L3 Nom du diplôme : Licence d'Informatique
Code du module : SIN5U2TL Libellé du module : Programmation et conception orientées objet
Calculatrices autorisées : NON Documents autorisés : OUI, notes de Cours/TD/TP

Exercice 1 : Composition et construction de code HTML

Considérons la classe suivante qui n'est pas bien conçue. Elle permet de construire une chaîne de caractères au format HTML.

```
public class HTMLElement {
    enum Tag { TEXT, PARAGRAPH, IMAGE, DIV, HEADING }

    private final Tag tag;
    private String text;
    private int headingSize;
    private HTMLElement[] content;

    public HTMLElement(Tag tag, String text) {
        this.tag = tag; this.text = text;
    }

    public HTMLElement(Tag tag, HTMLElement... content) {
        this.tag = tag; this.content = content;
    }

    public HTMLElement(int headingSize, HTMLElement... content) {
        this.tag = Tag.HEADING;
        this.headingSize = headingSize; this.content = content;
    }

    public String toHtml() {
        switch (tag) {
            case TEXT: return text;
            case PARAGRAPH: return "<p>"+content()+"</p>";
            case DIV: return "<div>"+content()+"</div>";
            case IMAGE: return "<img src=\""+text+"\"/>";
            case HEADING: return "<h"+headingSize+">"+content()+"</h"+headingSize+">";
            default: throw new NoSuchElementException();
        }
    }

    private String content() {
        StringBuilder builder = new StringBuilder();
        for (HTMLElement contentElement : content)
            builder.append(contentElement.toHtml());
        return builder.toString();
    }

}
```

1. Qu'affiche le code suivant ?

```
HTMLElement image = new HTMLElement(HTMLElement.Tag.IMAGE, "toto.jpg");
HTMLElement title = new HTMLElement(HTMLElement.Tag.TEXT, "mon titre");
HTMLElement titleHeading = new HTMLElement(4, title);
HTMLElement text = new HTMLElement(HTMLElement.Tag.TEXT, "mon texte");
HTMLElement paragraph = new HTMLElement(HTMLElement.Tag.PARAGRAPH, text);
HTMLElement block = new HTMLElement(HTMLElement.Tag.DIV, titleHeading, paragraph);
HTMLElement article = new HTMLElement(HTMLElement.Tag.DIV, image, block);
System.out.println(article.toHtml());
```

2. Que peut-on dire de l'utilisation et du nommage de l'attribut `text` de la classe `HTMLElement` ?
3. Quels principes SOLID sont violés par la classe `HTMLElement` ? Justifiez votre réponse.
4. Donnez le diagramme de classes d'une nouvelle organisation du code qui corrige ce problème. Vous devez faire en sorte de ne pas dupliquer le code de la méthode `String content()`.
5. Implémentez le diagramme de classes donné à la question précédente.
6. Nous souhaitons générer les quatres structures de documents HTML suivantes pouvant représenter un article sur un site d'actualité. Un événement doit avoir un contenu textuel et peut avoir un titre et une image :

```
<div>                   <div>              <p>content</p></div></div>
```

```
<div><img src="image"/><div>               <p>content</p></div></div>
```

```
<div>                   <div><h4>title</h4><p>content</p></div></div>
```

```
<div><img src="image"/><div><h4>title</h4><p>content</p></div></div>
```

Proposez une implémentation de la classe `ArticleBuilder` qui permette de générer les structures précédentes de la façon suivante :

```
HTMLElement article = new ArticleBuilder("content").build();
```

```
HTMLElement article = new ArticleBuilder("content").withImage("image").build();
```

```
HTMLElement article = new ArticleBuilder("content").withTitle("title").build();
```

```
HTMLElement article = new ArticleBuilder("content")
                          .withTitle("title")
                          .withImage("image").build();
```

N'oubliez pas de faire l'exercice 2

Exercice 2 : Itérateurs de chaînes de caractères

Considérons les deux classes suivantes :

```
public class StringIterator implements Iterator<Character> {

    private final String string;
    private int position;

    public StringIterator(String string) {
        this.string = string;
        this.position = 0;
    }

    @Override
    public boolean hasNext() {
        return position < string.length();
    }

    @Override
    public Character next() {
        if (!hasNext()) throw new NoSuchElementException();
        Character character = string.charAt(position);
        position++;
        return character;
    }
}
```

```
public class IterableString extends StringIterator implements Iterable<Character> {

    public IterableString(String string) {
        super(string);
    }

    @Override
    public Iterator<Character> iterator() {
        return this;
    }

}
```

1. Qu'affiche le code suivant ?

   ```
   IterableString test = new IterableString("test");
   for (char c : test) System.out.println(c);
   for (char c : test) System.out.println(c);
   ```

2. Quel principe SOLID est violé par les classes `StringIterator` et `IterableString` ? Justifiez votre réponse.
3. Proposez le diagramme de classes d'une nouvelle organisation du code qui corrige le problème.
4. Implémentez le diagramme de classes donné à la question précédente.