Technique de Test
TP1

Réalisé par : Stamboul Sofiane
Rovelli Hugo

1 Listing des tests

- ***TestTextAppend:***

  (Cette fonction test prend donc en argument une chaîne de caractère)

  1. Teste si la chaîne de caractère n'est pas nulle.
  2. Teste si la chaîne de caractère n'est pas vide.
  3. Teste si la chaîne de caractère ne commence pas par des espaces.
  4. Teste si la chaîne de caractère ne contient pas '\n' ou '\r'
  5. Teste si la chaîne de caractère contient des espace successifs .

- ***TestJustifyText :***

  1. Teste si le rendu est correcte (le texte est entièrement justifié).
  2. Teste si un mot est plus grand que la taille de justification
  3. Test si le texte justifié n'est pas nul.
  4. Test si le nombre de caractères pour la justification n'est pas nul.
  5. Test si le un mot n'est pas null ou composé d'espace.

Ref : JustifierTest.java

Réalisé par : Stamboul Sofiane
Rovelli Hugo

2 Test Justifier.java (Stamboul Sofiane)

- ***TestTextAppend:***

  (Cette fonction test prend donc en argument une chaîne de caractère)

  1. ***Teste si la chaîne de caractère n'est pas nulle :*** OK
  2. ***Teste si la chaîne de caractère n'est pas vide :***

     L'erreur de test est déclenché cela signifie qu'il serait nécessaire d'ajouter une exception ou ajouter une condition dans la méthode.
  3. ***Teste si la chaîne de caractère ne commence pas par des espaces.***

     L'erreur de test est déclenché cela signifie qu'il serait nécessaire d'ajouter une exception, ajouter une condition dans la méthode ou supprimer les espaces.
  4. ***Teste si la chaîne de caractère ne contient pas '\n' ou '\r'***

     L'erreur de test est déclenché cela signifie qu'il serait nécessaire d'ajouter une exception, ajouter une condition dans la méthode ou supprimer ces caractères.
  5. ***Teste si la chaîne de caractère contient des espace successifs .***

     L'erreur de test est déclenché cela signifie qu'il serait nécessaire d'ajouter une exception, ajouter une condition dans la méthode ou supprimer un des espace.

- ***TestJustifyText :***

  1. ***Teste si le rendu est correcte (le texte est entièrement justifié) :*** Ok
  2. ***Teste si un mot est plus grand que la taille de justification :*** Ok
  3. ***Test si le texte justifié n'est pas nul :*** Ok
  4. ***Test si le nombre de caractères pour la justification n'est pas nul :*** Ok
  5. ***Test si le un mot n'est pas null ou composé d'espace. :***
     Voir TestTextAppend pou explication et solution.

Ref : Justifier.java (Stamboul Sofiane)

Réalisé par : Stamboul Sofiane
Rovelli Hugo

3 Test Justifier.java (Stamboul Sofiane)

- ***TestTextAppend:***

  (Cette fonction test prend donc en argument une chaîne de caractère)

  1. ***Teste si la chaîne de caractère n'est pas nulle :*** OK
  2. ***Teste si la chaîne de caractère n'est pas vide :***OK
  3. ***Teste si la chaîne de caractère ne commence pas par des espaces:***OK
  4. ***Teste si la chaîne de caractère ne contient pas '\n' ou '\r':*** OK
  5. ***Teste si la chaîne de caractère contient des espace successifs .***

     L'erreur de test est déclenché cela signifie qu'il serait nécessaire d'ajouter une exception, ajouter une condition dans la méthode ou supprimer automatiquement un des espaces.

- ***TestJustifyText :***

  1. ***Teste si le rendu est correcte (le texte est entièrement justifié) :*** Ok
  2. ***Teste si un mot est plus grand que la taille de justification :***

     L'erreur de teste est déclencher ca signifie que ce cas la n'est pas gérer .
  3. ***Teste si le texte justifié n'est pas nul :*** Ok
  4. ***Teste si le nombre de caractères pour la justification n'est pas nul :*** Ok
  5. ***Teste si le un mot n'est pas null ou composé d'espace. :*** OK

Ref : Justifier.java (Rovelli Hugo)

Réalisé par : Stamboul Sofiane
Rovelli Hugo

4 Indexe (Code)

JustifierTest.java

```
public class JustifierTest {

	@Test
	public void testAppendText(String textToAppend) {
		assertNotNull("Erreur, textToAppend est null !",textToAppend);
		assertTrue("Erreur, Chaine de caractére vide!", !textToAppend.isEmpty());
		assertTrue("Erreur, Chaine de caractére commence par un ou plusieurs espace(s)!", !
	textToAppend.startsWith(" "));
		assertTrue("Erreur, Retour a la ligne interdit !", !textToAppend.contains("\n") && !
	textToAppend.contains("\r"));
		assertTrue("Erreur, deux espaces concécutifs!", !textToAppend.contains("  "));
	}

	@Test
	public void testJustifyText(String justifiedText, int nbChar) {

		assertNotNull("Erreur, justifiedArrayText est null !",justifiedText);
		assertNotNull("Erreur, nbChar est null !",nbChar);

		String justifiedTextTmp =
	justifiedText.replaceAll(System.getProperty("line.separator"), " ");
		String[] justifiedArrayText = justifiedTextTmp.split(" ");

		for (String it : justifiedArrayText) {
				assertTrue("Erreur, Chaine de caractére vide!", !it.isEmpty());
				assertTrue("Erreur,le mot \'"+it+"\' plus grand que
	nbChar \'"+nbChar+"\' ! ( valeur de justification )",it.length()<=nbChar);
		}

		String [] sentences =justifiedText.split(System.getProperty("line.separator"));
		for(String sentence :sentences){
			assertTrue("Erreur, nombre de caractères par ligne plus grand que la valeur
	de justification!",sentence.length()<=nbChar);
		}

	}
}
```

Réalisé par : Stamboul Sofiane
Rovelli Hugo

Justifier.java (Rovelli Hugo)

```
public class Justifier implements IBaseJustifier{

	private int nbChar=30;
	private String text="";
	private JustifierTest myJustifierTest = new JustifierTest();

	public Justifier(int nbChar){
		this.nbChar =nbChar;
	}

	@Override
	public void displayText() {
		System.out.println(text);
	}

	@Override
	public void appendText(String textToAppend) {
		if(text.isEmpty()){
			this.text += textToAppend;
		}
		else{
			this.text += " "+textToAppend;
		}
		myJustifierTest.testAppendText(textToAppend);
	}

	@Override
	public String justifyText() {
		if(!text.isEmpty()){
			if(text.length()>nbChar){
				String[] arrayText=text.split(" ");

				String justifiedText="";
				String tmpText="";
				for(int i=0; i<arrayText.length; i++){
					if(tmpText.isEmpty()){
						tmpText=arrayText[i];
					}
					else{
						if((tmpText.length()
	+arrayText[i].length())<nbChar){
							tmpText += " "+arrayText[i];
						}
						else{
							justifiedText +=
	tmpText+System.getProperty("line.separator");
							tmpText = arrayText[i];
						}
					}
				}
				myJustifierTest.testJustifyText(justifiedText, nbChar);
				return justifiedText;
			}
			return text;
		}
		return null;
	}
}
```

Réalisé par : Stamboul Sofiane
Rovelli Hugo

Justifier.java (Stamboul Sofiane)

```java
public class Justifier implements IBaseJustifier {
	private int numberForJustify;
	private String sentence;
	private JustifierTest myJustifierTest = new JustifierTest();

	public Justifier(int charToJustify) {
		this.setNumberForJustify(charToJustify);
		sentence = "";
	}

	public Justifier() {
		this.setNumberForJustify(30);
		sentence = "";
	}

	@Override
	public void displayText() {
		System.out.println(justifyText());
	}

	@Override
	public void appendText(String textToAppend) {
		myJustifierTest.testAppendText(textToAppend);
		if (!sentence.equals(""))
			this.sentence += " ";
		this.sentence += textToAppend;
	}

	@Override
	public String justifyText() {
		int cpt = 0;
		String Newligne = System.getProperty("line.separator");
		String justifiedSentence = "";
		String[] tab = sentence.split(" ");
		for (String word : tab) {
			cpt += word.length();
			if (cpt + 2 > numberForJustify) {
				justifiedSentence += Newligne + word;
				cpt = word.length();
			} else {
				if(justifiedSentence!="")
				justifiedSentence +=" ";
				justifiedSentence+=word;
			}

		}
		myJustifierTest.testJustifyText(justifiedSentence, numberForJustify);
		return justifiedSentence;
	}

	public int getNumberForJustify() {
		return numberForJustify;
	}

	public void setNumberForJustify(int numberForJustify) {
		this.numberForJustify = numberForJustify;
	}

}
```

Réalisé par : Stamboul Sofiane
Rovelli Hugo