



UNIVERSITÉ  
CÔTE D'AZUR

UNIVERSITÉ CÔTE D'AZUR

---

Takenoko

<https://github.com/Mahe-Thomas/takenoko>

---

*Auteur :*

Jeremy BONSAUDO

Matthias PERCELAY

Brandon FONTANY-LEGALL

Thomas MAHE

4 novembre 2018

# Table des matières

<b>1</b>	<b>Présentation du projet</b>	<b>1</b>
1.1	Sujet . . . . .	1
1.2	Avancement du projet . . . . .	1
1.3	Problématique soulevée . . . . .	1
1.3.1	Le bon partitionnement du code . . . . .	1
1.3.2	L'injection de dépendance . . . . .	1
<b>2</b>	<b>Organisation du code</b>	<b>2</b>
2.1	Packages . . . . .	2
<b>3</b>	<b>Conception</b>	<b>3</b>
3.1	Patron de conception . . . . .	3
<b>4</b>	<b>Spring</b>	<b>4</b>
4.1	Notre Spring . . . . .	4
<b>5</b>	<b>Bilan</b>	<b>5</b>

# Chapitre 1

## Présentation du projet

### 1.1 Sujet

Le projet consiste à réaliser en Java une version numérique du jeu Takenoko créé par Antoine Bauza. Version textuel n'étant pas destinée à être jouée par des êtres humains mais par des robots jouants de façon autonome. Dans le Takenoko nous allons endosser le rôle d'un jardinier japonais affairé à répondre au mieux aux requêtes du vénérable empereur. Pour ce faire, nous aurons, durant notre tour de jeu, deux choses à faire : Regarder quel temps il fait et effectuer deux actions parmi les possibilités suivantes : agrandir le jardin, irriguer, bouger le jardinier, bouger le panda.

### 1.2 Avancement du projet

Le projet en l'état est pratiquement complet, il manque en effet le dèc qui pourra être implémenter ultérieurement car nous avons à l'heure actuelle privilégié l'injection de dépendance au dèc.

### 1.3 Problématique soulevée

Plusieurs problématiques ont été soulevées durant le projet qui sont les suivantes :

#### 1.3.1 Le bon partitionnement du code

L'organisation du code a été pour nous un assez grand problème notamment à cause de l'apparition de "God classe"<sup>1</sup> comme la classe de Jeu qui concentré un trop grand nombre de fonctions. Nous avons donc fait plusieurs ré-aménagements (Voir paragraphe Conception3)

#### 1.3.2 L'injection de dépendance

Du fait de la nouveauté de l'outils, nous avons rencontrer plusieurs problèmes lors de l'injection de dépendance. (Voir paragraphe Spring4)

---

1. La "God classe" est une classe comportant un très grand nombre de fonctions qui pourraient être dispatcher dans d'autre classes

# Chapitre 2

## Organisation du code

Intro <sup>1</sup>

### 2.1 Packages

Bla(cf. fig. 1.1)

Bla  
Bla

Bla  
Bla <sup>2</sup>

Bla(cf. ref. [?]).

---

1. Note bas de page "intro"  
2. Note bas de page "bla"

# Chapitre 3

## Conception

Intro <sup>1</sup>

### 3.1 Patron de conception

Bla(cf. fig. 1.1)

Bla  
Bla

Bla  
Bla <sup>2</sup>

Bla(cf. ref. [?]).

---

1. Note bas de page "intro"  
2. Note bas de page "bla"

# Chapitre 4

## Spring

Intro <sup>1</sup>

### 4.1 Notre Spring

Bla(cf. fig. 1.1)

Bla  
Bla

Bla  
Bla <sup>2</sup>

Bla(cf. ref. [?]).

---

1. Note bas de page "intro"  
2. Note bas de page "bla"

# Chapitre 5

## Bilan

Intro / Rappel Contexte

Nous avons donc pu en tirer la problématique suivante :

Problématique du sujet

Bla

Bla

Bla

Bla

Bla

Bla

Bla