Solutions De Confiance

Documentation Projet :  
DecisionTreeApp

Documentation détaillée

Ce document vise à une description détaillée du code et des divers éléments de ce projet. Cette documentation est en cours de modification.

DTS – (Document technique simplifié)

Interne

22 August 2023

**Fiche de contrôle**

|  |  |  |
| --- | --- | --- |
| Version | Date | Change Description |
| v 1.0 | 16/08/2023 | Création du document |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
| Rédacteurs | Fonction | Contact |
| **DRAOU Meryem** | Ingénieur de production | meryem.draou@orangecyberdefense.com |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
| Destinataires | Fonction | Contact |
|  |  |  |
|  |  |  |
|  |  |  |

Table des matières

**Documentation4**

**Diagramme de classes10**

**Indications et procédures d’installation11**

## **Documentation**

## **Classe DecisionTreeNode**

## **1. Introduction**

La classe **DecisionTreeNode** représente un nœud dans un arbre de décision utilisé pour une prise de décision. Chaque nœud contient des informations sur une étape de décision et ses différents états possibles.

**2. Méthodes**

**2.1 Constructeur**

**public DecisionTreeNode(String message)**

Ce constructeur initialise un nouveau nœud de décision avec le message spécifié. Le message représente l'information ou la question associée à un nœud.

**2.2 Méthode getNameNode()**

**public String getNameNode()**

Cette méthode permet de récupérer le nom du nœud actuel. Le nom est généralement le même que le message utilisé pour initialiser le nœud.

**2.3 Méthode getInput()**

**public String getInput()**

Cette méthode renvoie le message ou l'information associé au nœud actuel.

**2.4 Méthode getChild(String decision)**

**public DecisionTreeNode getChild(String decision)**

Cette méthode permet de récupérer le nœud enfant associé à une décision donnée. La décision est utilisée comme clé pour accéder au nœud enfant dans la structure de données.

**2.5 Méthode addChild(String decision, DecisionTreeNode child)**

**public void addChild(String decision, DecisionTreeNode child)**

Cette méthode permet d'ajouter un nœud enfant à ce nœud de décision. Elle associe le nœud enfant à une décision donnée en utilisant la structure de données HashMap.

**2.6 Méthode isLeaf()**

**public boolean isLeaf()**

Cette méthode vérifie si le nœud actuel est une feuille de l'arbre de décision, c'est-à-dire s'il n'a pas d'enfants. Si c'est le cas, le nœud est une feuille (« leaf » en l’occurrence).

**2.7 Méthode getChildren()**

**public HashMap<String, DecisionTreeNode> getChildren()**

Cette méthode renvoie la liste des nœuds enfants associés à ce nœud de décision. Les nœuds enfants sont stockés dans une structure de données de type HashMap, où la clé est la décision associée au nœud enfant.

**Exemple d'utilisation**

**DecisionTreeNode root = new DecisionTreeNode("Start");**

**DecisionTreeNode child1 = new DecisionTreeNode("Decision A");**

**DecisionTreeNode child2 = new DecisionTreeNode("Decision B");**

**root.addChild("A", child1);**

**root.addChild("B", child2);**

**DecisionTreeNode selectedNode = root.getChild("A");**

**if (selectedNode != null) {**

**System.out.println("SelectedNode" + selectedNode.getNameNode());**

**}**

**Attributs**

**private String input**

Attribut correspondant à l’input de l’utilisateur.

**private HashMap<String, DecisionTreeNode>**

Attribut correspondant à une structure associant une chaîne de caractère à un nœud de l’abre de décision.

**private String nameNode**

Attribut correspondant au nœud du nœud de l’arbre de décision.

## **Classe DecisionTreeRunner**

**1. Introduction**

La classe **DecisionTreeRunner** est responsable de l'exécution d'un arbre de décision interactif via une interface graphique Swing. Elle permet à l'utilisateur de traverser l'arbre de décision en répondant à des questions et affiche les choix sélectionnés dans un tableau à mesure que l’utilisateur progresse.

**2. Méthodes**

**2.1 Constructeur**

**public DecisionTreeRunner()**

Le constructeur initialise une instance de **DecisionTree** qui sera utilisée pour exécuter l’arbre de décision intéractif.

**2.2 Méthode run()**

**public void run()**

Cette méthode gère l'interface graphique Swing et la logique d'exécution de l'arbre de décision. Elle affiche des messages à l'utilisateur, recueille les entrées et met à jour l’interface graphiqe en conséquence.

**2.3 Méthode main(String[] args)**

**public static void main(String[] args)**

La méthode **main()** démarre l'exécution de l'application en invoquant la méthode **run()** dans un thread d'interface utilisateur.

## **Classe DecisionTreePanel**

La classe **DecisionTreePanel** étend **JPanel** et est utilisée pour dessiner l'arbre de décision interactif à l'aide de graphiques Swing.

**Méthodes de DecisionTreePanel**

**2.4 Méthode paintComponent(Graphics g)**

**@Override**

**protected void paintComponent(Graphics g)**

Cette méthode est responsable de dessiner les nœuds et les liens de l'arbre de décision sur le composant graphique. Elle appelle ensuite la méthode **drawTree()** pour dessiner les nœuds de l’arbre.

**2.5 Méthode drawTree(DecisionTreeNode node, int x, int y, int xOffset, int level, Graphics g)**

**private void drawTree(DecisionTreeNode node, int x, int y, int xOffset, int level, Graphics g)**

Cette méthode dessine l'arbre de décision en plaçant les nœuds et en dessinant les liens entre eux. Elle gère également la coloration des nœuds en fonction de leur type et de leur position dans l’arbre.

**Remarque**

La classe **DecisionTreePanel** est utilisée pour créer une représentation visuelle de l'arbre de décision à l'aide des graphiques Swing. Elle détermine comment les nœuds et les liens de l’arbre sont dessinés à l’écran.

La classe **DecisionTreeRunner** met en œuvre une interface utilisateur interactive pour guider les utilisateurs à travers l'arbre de décision en affichant des questions, en recueillant les réponses et en affichant les choix sélectionnés.

La classe **DecisionTreePanel** étend **JPanel** et est utilisée pour dessiner visuellement l'arbre de décision à l'aide de graphiques Swing. L'ensemble du code crée une application Swing qui permet aux utilisateurs de naviguer dans un arbre de décision et d'afficher les choix sélectionnés dans un tableau.

**Classe Main**

**1. Introduction**

La classe **Main** est une classe de base qui contient la méthode **main**(), point d'entrée pour l'exécution d'une application Java. Dans cet exemple, la classe **Main** contient une simple instruction d'impression pour afficher "Test." dans la console.

**2. Méthode main(String[] args)**

**public static void main(String[] args)**

C'est la méthode principale qui est appelée lorsque l'application Java est exécutée. Elle prend un tableau de chaînes args en tant qu'argument, qui peut être utilisé pour passer des paramètres à l'application lors de son exécution.

**3. Instructions dans main**

**System.out.println("Test.");**

Dans cet exemple, la méthode **main()** contient une seule instruction qui utilise la fonction **println()** pour afficher "Test." suivi d'un saut de ligne dans la console. Cela servira simplement à vérifier que le programme s'exécute correctement.

**Classe DecisionTree**

**1. Introduction**

La classe **DecisionTree** représente un arbre de décision avec des nœuds et des relations de décision entre eux. L'arbre est construit en utilisant des instances de la classe **DecisionTreeNode** pour chaque nœud.

**2. Attributs**

**2.1 Attribut root**

**DecisionTreeNode root**

Cet attribut maintient la référence du nœud racine de l'arbre de décision.

**2.2 Attribut pathNodes**

**ArrayList<String> pathNodes**

Cet attribut est une liste qui stocke les décisions prises par l'utilisateur en traversants l’arbre de décision.

**3. Méthodes**

**3.1 Méthode addPathNodes(String decision)**

**public void addPathNodes(String decision)**

Cette méthode ajoute la décision prise par l'utilisateur à la liste des nœuds de chemin (pathNodes) pour suivre le parcours de l’utilisateur à travers l’arbre.

**3.2 Méthode getPathNodes()**

**public ArrayList<String> getPathNodes()**

Cette méthode retourne la liste des nœuds de chemin (pathNodes) qui contiennent les décisions prises par l'utilisateur lors de son parcours dans l’arbre.

**3.3 Constructeur**

**public DecisionTree()**

Le constructeur de la classe initialise l'arbre de décision en créant les nœuds de l'arbre et en établissant les relations de décisions entre eux.

**3.4 Méthode decide()**

**public void decide()**

Cette méthode guide l'utilisateur à travers l'arbre de décision en affichant les questions et en lisant les réponses de l'utilisateur depuis la console. Elle met à jour la liste des nœuds de chemin (pathNodes) en fonction des décisions prises par l'utilisateur.

**3.5 Méthode getRoot()**

**public DecisionTreeNode getRoot()**

Cette méthode retourne le nœud racine de l’arbre de décision.

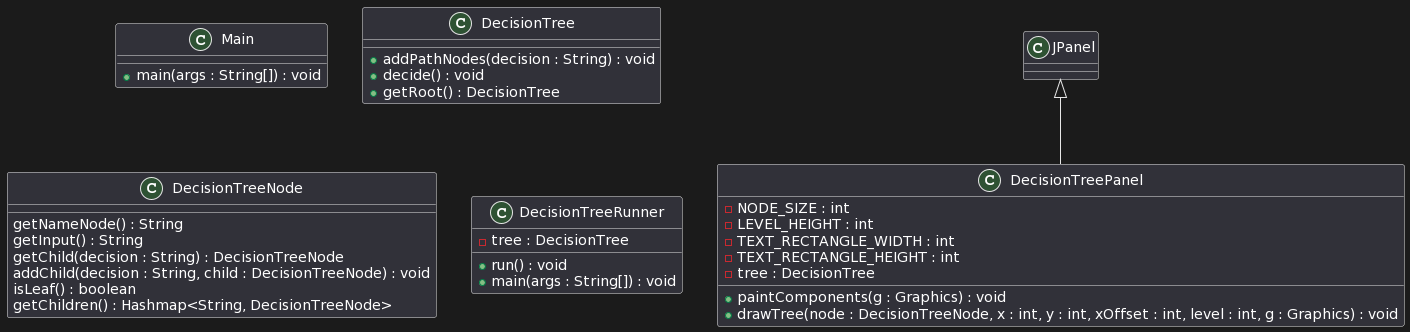
**Remarque**

La classe DecisionTree gère un arbre de décision interactif en créant des instances de nœuds de décision (DecisionTreeNode) et en les reliant pour former un arbre. Elle offre des méthodes pour guider l'utilisateur à travers l'arbre en affichant des questions et en collectant les réponses de l'utilisateur. La classe conserve également une liste des décisions prises par l'utilisateur pendant le parcours.

**Diagramme de classes**

**Projet : DecisionTreeApp**

Le diagramme de classe suivant a été codé sur **Plantuml**.



**Lien de la version du mercredi 16/08**

Sur le site suivant, il est nécessaire d’entrer dans la barre prévue à cette effet le lien suivant :

//www.plantuml.com/plantuml/dpng/dLBHQXin47pNL-pZ5Ri\_K4f8i8NjSd62SwIbfPYjjRO5kjKXwLA7ad-lvFEv8Zw\_5Ev1ExCxitFIhG\_eGbiRiJ7eFToZPlWZ1C08wdWkqEqyV8Adu3JlVlpy4CylLYiXtiHfP4eRxRNboX6bqO\_Wh0snBoUMLb5a6WpxT4oITyxXQwbz-Do9tmWXHe1AFVPDXJefd1SusuyWSOgA3DbHM4Nd8vHxHyeHgcU3VEJcwFVts9ninv28copGnU8fLHHVOftAphCRJl22cpPa62HmijT63Uz\_eIvYvYlzOzWaOc2gpmr2-v9mMoJWbxM6aCyBEFgNkCuIlnVBq-VSgcKcrrrgYFNx7vl-RClv7Vp7ctb49WDq2CJAmvUkF5ek7wPo\_RJuBgE0vX2HKZxBSZsNYzcyEeEL\_5QjLt9ItIrdfLn\_NKoh-JKo7xqUgi58JspTM2OEljZ5dfd3PgytFd-23d-daOAjee4x6iEXSnh3QtyuF6otda9V6deXqnU3Dk8jVRV4ggtDNm00

**Indications et procédures d’installation**

Procédures d’installation d’IntelliJ IDEA

Lancer le projet :

* + Ouvrir le projet DecisionTreeApp
  + Lancer le projet à partir du fichier « DecisionTreeRunner.java », la classe DecisionTreeRunner est la classe principale

**Comment effectuer toutes les installations préliminaires**

**Étape 1 : Téléchargement de Java Development Kit (JDK)**

1. Rendez-vous sur le site officiel d'Oracle pour télécharger le JDK : <https://www.oracle.com/java/technologies/javase-jdk11-downloads.html>
2. Acceptez les termes et conditions de la licence
3. Téléchargez la version appropriée du JDK pour votre système d'exploitation (Windows ou Linux)

**Étape 2 : Installation du JDK**

1. Exécutez le programme d’installation
2. Suivez les indications
3. Choisissez l'emplacement d'installation du JDK
4. Configurez les paramètres de l'environnement si nécessaire (par exemple, la variable d'environnement JAVA\_HOME)

**Étape 3 : Téléchargement d'IntelliJ IDEA**

1. Rendez-vous sur le site officiel d'IntelliJ IDEA pour télécharger l'IDE : <https://www.jetbrains.com/idea/download/>
2. Sélectionnez la version appropriée pour votre système d'exploitation (Community Edition ou Ultimate Edition)
3. Téléchargez le fichier d'installation

**Étape 4 : Installation d'IntelliJ IDEA**

1. Exécutez le fichier d'installation téléchargé
2. Suivez les instructions de l'assistant d'installation
3. Sélectionnez les composants à installer (vous pouvez laisser les paramètres par défaut sauf si vous avez des besoins spécifiques)
4. Choisissez l'emplacement d'installation d'IntelliJ IDEA
5. Configurez les paramètres de lancement si nécessaire (par exemple, les raccourcis)

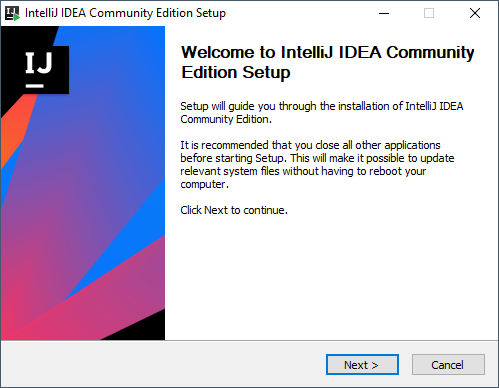
**Étape 5 : Configuration d'IntelliJ IDEA**

1. Lancez IntelliJ IDEA
2. Sélectionnez "Configure" dans la barre de menu, puis "Project Defaults", et enfin "Project Structure"
3. Dans la fenêtre qui s'ouvre, cliquez sur "SDKs" dans la colonne de gauche
4. Cliquez sur le signe "+" pour ajouter un nouveau SDK, puis sélectionnez le dossier d'installation du JDK précédemment téléchargé et installé
5. Cliquez sur "Apply" ou "OK" pour enregistrer les paramètres

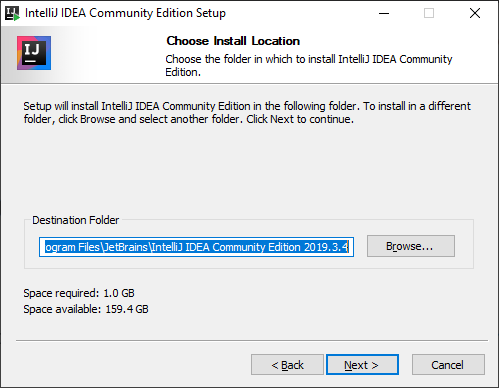
# Comment installer IntelliJ IDEA Community Edition

**Étape 1 :** Allez à la page d'IntelliJ IDEA sur Academic Software et cliquez sur le bouton 'Télécharger IntelliJ IDEA CE' pour télécharger le logiciel

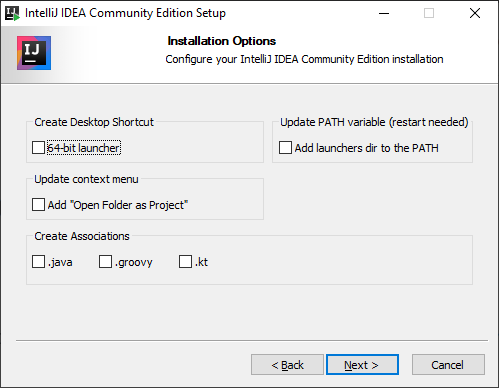
**Étape 2 :**  Ouvrez le fichier d'installation .exe dans votre dossier Téléchargements



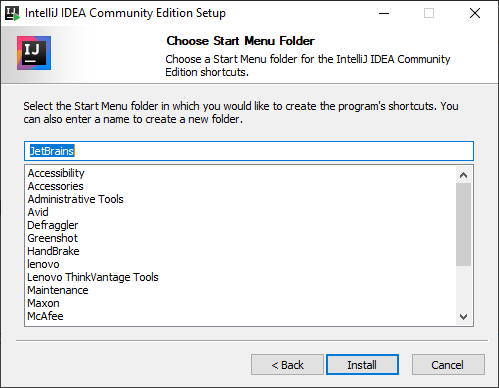
**Étape 3 :**  Cliquez sur Next pour installer le logiciel à l'emplacement par défaut. Si vous le souhaitez, vous pouvez également modifier le chemin d'installation

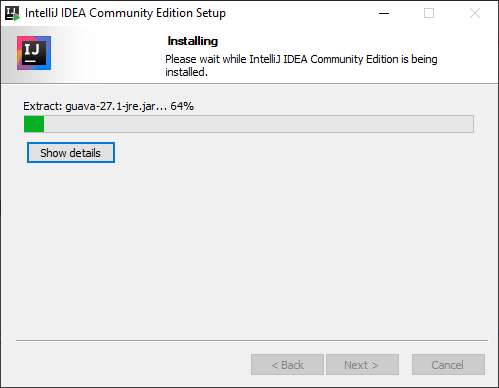


**Étape 4 :**  Choisissez des options supplémentaires telles qu'un lien sur le bureau, des associations de fichiers et un ajout au menu du clic droit

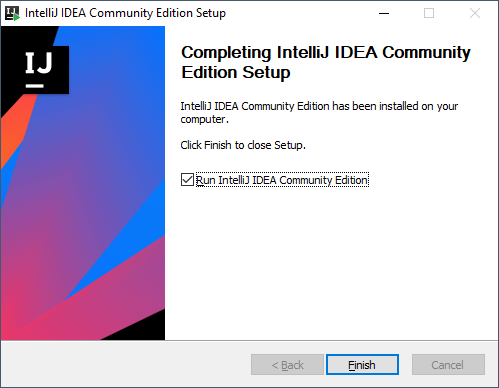


**Étape 5 :** Cliquez sur Install pour démarrer l’installation

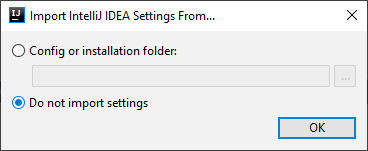




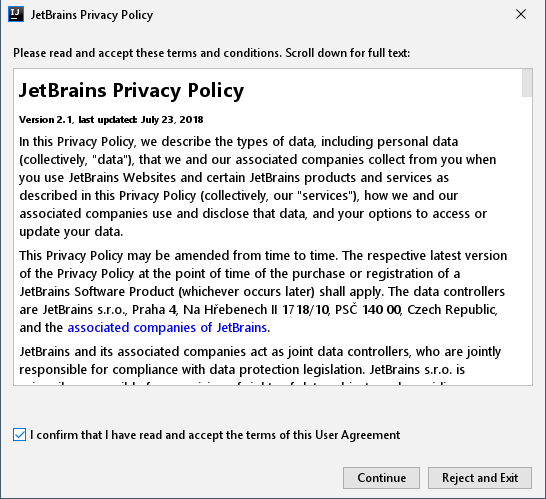
**Étape 6 :** Cochez la case pour démarrer le programma et cliquez sur Finish



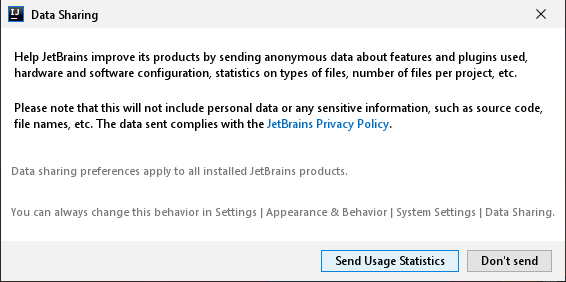
**Étape 7:**   Importez les paramètres précédents si nécessaire. Cliquez sur OK



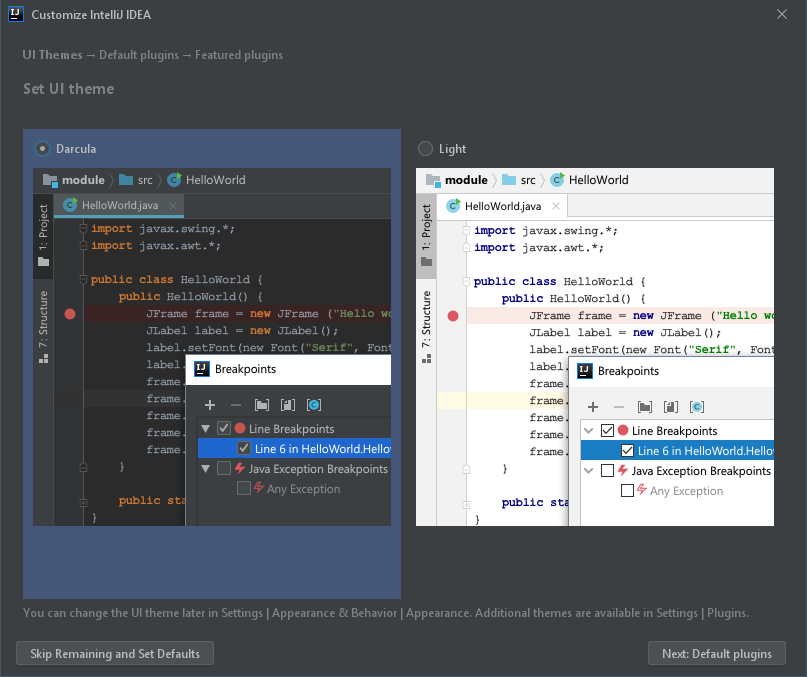
**Étape 8 :** Cochez la case et cliquez sur Continue pour accepter les conditions de la licence



**Étape 9 :** Choisissez si vous souhaitez envoyer des données d'utilisateur anonymes



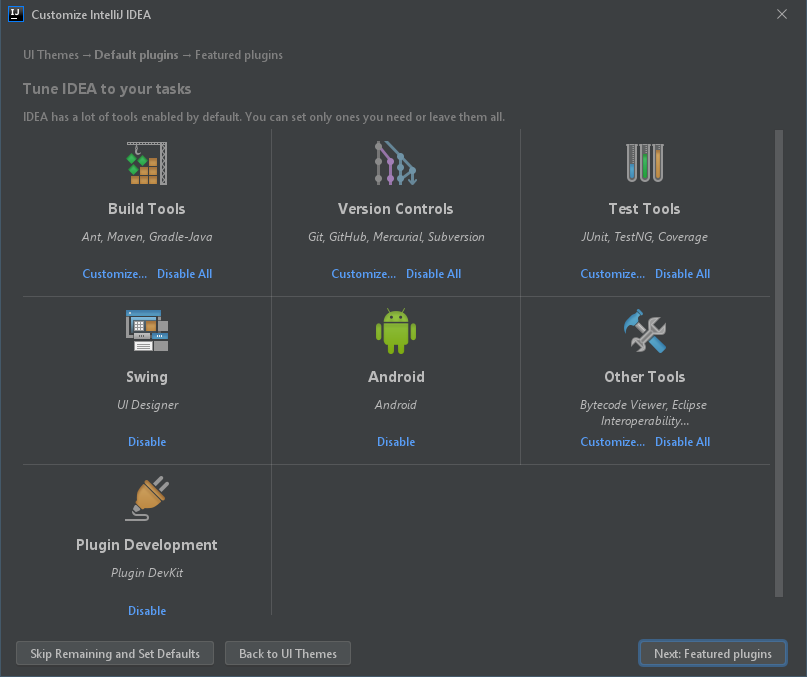
**Étape 10 :** Choisissez votre thème. Un thème sombre est plus doux pour les yeux, surtout dans l'obscurité



**Remarque**

Pour démarrer le logiciel avec les paramètres par défaut, vous pouvez cliquer sur 'Skip Remaining and Set Defaults' en bas à gauche. Sinon, cliquez sur Next

**Étape 11 :** Choisissez ici pour modifier ou supprimer des plugins



**Étape 12 :** Choisissez ici pour installer les plugins supplémentaires. Cliquez ensuite sur 'Start using IntelliJ IDEA'. Le logiciel est maintenant installé et prêt à l'emploi

