## Annexe 1

Présentation du formateur

Prénom : El Hadji

Nom : Gaye

Ville : Nantes  
 Diplômes :   
 Bac S (Lycée des Parcelles assainies de Dakar (Sénégal))

Deug de sciences et technologies pour l'ingénieur (Université de Rouen)

Licence de Physique (Université de Rouen)

Ingénieur en informatique et micro-électronique (Ecole Polytechnique de Lille)

Master d'informatique d'entreprise (Ecole Polytechnique d'Angers (ex ISTIA))

Postes occupés :

14 ans comme Lead Dev Java EE Spring C#/.Net Angular, Référent technique, Architecte, Expert E-Commerce, Chef de projet Technique et Dev Sénior.

10 ans comme Formateur Architecture Micro-Service (Conteneurisation avec Docker et orchestration Kubernetes), Devops, Java EE, .Net (C#,VB.NET et ASP.NET Core), PHP (7, 8 et 9 avec Symfony 4 et 5), Angular (2 à 16), React JS, Node Js et C, C++ ect... chez **Dawan**, **M2I Formation**,**CapGemini**, **IONIS, SupInfo**, **Epitech**, **ETNA, ORSYS ,PROFIND, Campus Académie (ex IMIE)**, **InfoTech**, **ESIA** ect...

Contact : **elhadji.gaye83@gmail.com**

## Annexe 2

**Java Standard Edition - Les fondamentaux de la programmation orientée objet Java**

**Objectif de la formation :**

Connaitre les éléments l’environnement de développement et d’exécution de Java (JVM, JDK, IDE).

Maîtriser les concepts avancés de la programmation orientée Objet avec Java

Réaliser et déployer des applications complètes, performantes et maintenables avec Java.

Utiliser des librairies et technologies pour créer une application.

**Quelques rappels algorithmiques**

* Notion de variable, de type de variable, d’opérateurs
* Notion de boucle.
* Notion de procédure et de fonctions
* Notion de tableaux, trie de tableau
* Algorithme de tri par sélection, tri par extraction, le trie par insertion, le trie
* par fusion etc…).

**Notions théoriques sur la programmation orienté objet**

* Notion de classe, notion d’objet
* Classe abstraite
* Héritage
* Polymorphisme
* Agrégation d'objets (association)

**Mise en place de l’environnement de développement**

* Les livraisons (Oracle, OpenJDK...)
* Les versions majeures de Java de 1.4 à la 17
* Structure d'une JDK
* JVM (Java Virtual Machine), JRE (Java Runtime Environment) et JIT (Just-In-Time)
* Java et Jakarta EE 9
* Les principales normes de Java et de Jakarta EE
* Les IDE du marché
* Installation d'Eclipse et fonctionnement

**Mise en œuvre des notions de Java**

* Utilisation de variables, constantes, opérateurs
* Les types de base et étendus
* Types simples et types références
* Transtypage, Wrappers
* Expression de conditions : if/else, switch, opérateur ternaire
* Utilisation de boucles : for, while, do while
* Manipulation de tableaux
* Procédures et fonctions avec Java
* Factorisation de codes avec méthodes
* Surcharge, arguments variables, récursivité
* Commenter et documenter du code

**Mise en œuvre de la POO (Programmation Orientée Objet) Java**

* Les classes
* Les packages
* L'héritage
* Super
* Protected access
* La redéfinition des constructeurs et méthodes
* Les classes abstraites
* Les interfaces en Java, default et private, apports
* Le polymorphisme
* Les sealed classes
* Les propriétés et méthodes
* Accès public et private
* Signature de méthodes
* Les constructeurs
* La surcharge
* Getter et Setter
* Les JavaBean, les POJO (Plain Old Java Object), les DTO (Data Transfer Object).
* Les records, pour remplacer les JavaBean
* Visibilité inter-classe et inter-package
* This et static
* if, switch et switch à notation fléchée, yield
* Les boucles standards et extensions de boucle for
* L'Autoboxing
* Les varargs
* Les enum
* Les classes internes
* Les classes anonymes
* Les tableaux en Java
* Les collections et énumérations en Java
* La notation fléchée et les lambdas pour les collections

**Gestion des collections**

* Classes essentielles : ArrayList, Vector, HashMap,...
* Parcourir des collections
* Trie de collections
* opérations sur des collections

**La gestion des exceptions**

* Définition, types d'exceptions
* Capturer et traiter une exception (try/catch/finally)
* Lever/Remonter une exception (throw/throws)
* Création d'exceptions

**API importantes de Java**

* L'API Optional
* L'API DateTime

**Manipulation de fichiers**

* Manipulation de flux binaires
* Lecture et écriture de fichiers
* Utilisation de buffers
* Manipulation de répertoires
* Manipulation de fichiers texte, XML et JSON.

**Les interfaces graphiques**

* Présentation de Swing
* Présentation de JavaFX

**JDBC (Java Database Connectivity)**

* Architecture de JDBC
* Drivers et gestionnaire de drivers
* Structuration des sources de données, les URL
* Principales interfaces et classes
* La notion de transaction, commit et rollback
* Différentes façons de connexion à une base de données
* Les interfaces Connection, Statement, PreparedStatement, ResultSet
* Mapping relationnel objet (Pattern DAO)
* Présentation de Framework ORM

**Les threads**

* Classe Thread et l'interface Runnable
* Problèmes du multithreading
* Attente et synchronisation
* Thread pools et environnements "thread safe"
* Package java.util.concurrent
* Semaphore et Lock
* Le framework Executor
* Asynchronisme en Java
* Future et Callable
* Le Fork and Join

**Gestionnaire de logs**

* Externalisation de configuration dans des .properties
* Log4j, Logback

## Annexe 3

Java est un langage de programmation orienté objet créé par James Gosling et Patrick Naughton. Ils étaient à l’époque des employés de Sun Microsystems. Sun a été racheté par la suite en 2009 par la société Oracle. Le principal avantage de java est sa portabilité. En effet, le code source Java peut être exécuté sur plusieurs systèmes d’exploitation (UNIX, Windows, Mac OS ou GNU/Linux).

Java dispose d’une machine virtuelle (JVM). Lors de la compilation, le code source java est traduit en un code intermédiaire appelé « byte code ». Ce code sera interprété par la machine virtuelle Java. Il arrive que par abus de langage la JVM soit nommée **JRE** (Java Runtime Environment).

## Annexe 4

Le JDK et le JRE (Java Runtime Environment) sont **les deux éléments qui composent la plate-forme Java SE** (Java Standard Edition).

Le **JRE** contient les bibliothèques logicielles (ou API pour Application Programming Interface), une machine virtuelle Java (**JVM** : Java Virtual Machine) et les composants nécessaires au fonctionnement d’applications développées en Java.

La machine virtuelle Java est conçue pour interpréter et exécuter le bytecode Java issu de la compilation du code source écrit lors du développement. La JVM est spécifique au type de système d’exploitation (Windows, Linux...) et de matériel associé (32-bits, 64-bits...), pour ainsi garantir un fonctionnement similaire sur ces différentes plates-formes avec le même code source écrit en Java.

Le **JDK** est une surcouche du JRE. Il se compose donc du JRE auquel il faut ajouter tous les éléments indispensables au développement d’applications Java, comme le compilateur et le débogueur.

Java SE est donc composé d’un ensemble d’API et de technologies autour de Java permettant de créer et faire fonctionner, sur les systèmes d’exploitation majeurs, des applications serveurs, des applications dites clients riches et des applets.

Une image contenant texte, capture d’écran, conception

Description générée automatiquement

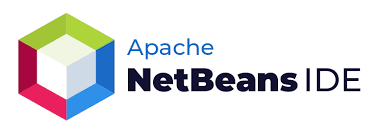
## Annexe 5

Effectuer les recherches Google :

Mot clés : logo netbeans

Une image contenant texte, diagramme, Police, logiciel

Description générée automatiquement



Mot clés : logo IntelliJ

Une image contenant texte, capture d’écran, logiciel, Icône d’ordinateur

Description générée automatiquement



Mot clés : logo Eclipse

Une image contenant texte, capture d’écran, logiciel, Icône d’ordinateur

Description générée automatiquement



Mot clés : logo Spring Tool Suite

Une image contenant texte, capture d’écran, Police, vert

Description générée automatiquement



Mot clés : logo Visual Studio Code

Une image contenant texte, capture d’écran, Police, Icône d’ordinateur

Description générée automatiquement



## Annexe 6

Ouvrir l’IDE et créer votre première application Java en cliquant sur **🡪File🡪New 🡪Java Project**

Une image contenant texte, capture d’écran, nombre, logiciel

Description générée automatiquement

Une image contenant texte, capture d’écran, logiciel, nombre

Description générée automatiquement

Une image contenant texte, capture d’écran, logiciel, nombre

Description générée automatiquement

Une image contenant texte, logiciel, Icône d’ordinateur, Page web

Description générée automatiquement

Une image contenant texte, Police, capture d’écran

Description générée automatiquement

Créer la classe **com.cours.MyClasse** :

Une image contenant texte, Police, nombre, logiciel

Description générée automatiquement

Une image contenant texte, capture d’écran, logiciel, nombre

Description générée automatiquement

Une image contenant texte, Police, capture d’écran, logiciel

Description générée automatiquement

Créer la classe de démarage **com.cours.main.MainApp** :

Une image contenant texte, Police, nombre, logiciel

Description générée automatiquement

Une image contenant texte, capture d’écran, logiciel, nombre

Description générée automatiquement

N’oublier pas de cocher sur **public static void main(String[] args)**.

On obtient :

Une image contenant texte, capture d’écran, logiciel, Police

Description générée automatiquement

Réalisons maintenant un programme Java qui affiche : « **Bonjour JAVA** »

La méthode **com.cours.main.MainApp.main** aura pour contenu :

package com.cours.main;

public class MainApp {

    public static void main(String[] args) {

        System.out.println("Bonjour JAVA");

    }

}

On obtient comme résultat :

Une image contenant texte, capture d’écran, Police, logiciel

Description générée automatiquement

Une image contenant texte, Police, ligne, logiciel

Description générée automatiquement

Une image contenant texte, capture d’écran, Police, logiciel

Description générée automatiquement

Pour fermer un projet Java il suffit de faire :

Une image contenant texte, capture d’écran, affichage, Police

Description générée automatiquement

Une image contenant texte, capture d’écran, Police, nombre

Description générée automatiquement

Une image contenant texte, capture d’écran, logiciel, affichage

Description générée automatiquement

Surtout ne pas cocher « **Delete project contents on disk** »

Pour ouvrir un projet Java existant il suffit de faire :

Une image contenant texte, capture d’écran, Police, logiciel

Description générée automatiquement

Une image contenant texte, ligne, capture d’écran, Police

Description générée automatiquement

Mettre dans **Directory** : **C:/Users/El Hadji/eclipse-workspace/app-java-samples** puis cliquer sur **Finish**.

Une image contenant texte, capture d’écran, logiciel, Icône d’ordinateur

Description générée automatiquement

## Annexe 7

*Le type int*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Type** | **Plage** | **Taille** | **Objet correspondant** | **Valeur par défaut** |
| int | -2 147 483 648 à 2 147 483 647 | 4 octets | java.lang.Integer | 0 |

*Le type float*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Type** | **Plage** | **Taille** | **Objet correspondant** | **Valeur par défaut** |
| float | -2-31 à 2+31 -1 | 4 octets | java.lang.Float | 0 |

*Le type long*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Type** | **Plage** | **Taille** | **Objet correspondant** | **Valeur par défaut** |
| long | –9,223,372,036,854,775,808 à 9,223,372,036,854,775,807 | 8 octets | java.lang.Long | 0 |

*Le type double*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Type** | **Plage** | **Taille** | **Objet correspondant** | **Valeur par défaut** |
| double | -1.7 x 10-308 à 1.7 x 10+308 | 8 octets | java.lang.Double | 0 |

*Le type byte*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Type** | **Plage** | **Taille** | **Objet correspondant** | **Valeur par défaut** |
| byte | -128 à +127 | 1 octet | java.lang.Byte | 0 |

*Le type short*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Type** | **Plage** | **Taille** | **Objet correspondant** | **Valeur par défaut** |
| short | -32768 à +32767 | 2 octets | java.lang.Short | 0 |

*Le type char*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Type** | **Plage** | **Taille** | **Objet correspondant** | **Valeur par défaut** |
| Char | Caractère Unicode (65536 caractères possibles) | 2 octets | java.lang.Character |  |

*Le type boolean*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Type** | **Valeurs** | **Taille** | **Objet correspondant** | **Valeur par défaut** |
| boolean | true ou false | 1 octet | java.lang.Boolean | false |