JAVA

1. Présentation :

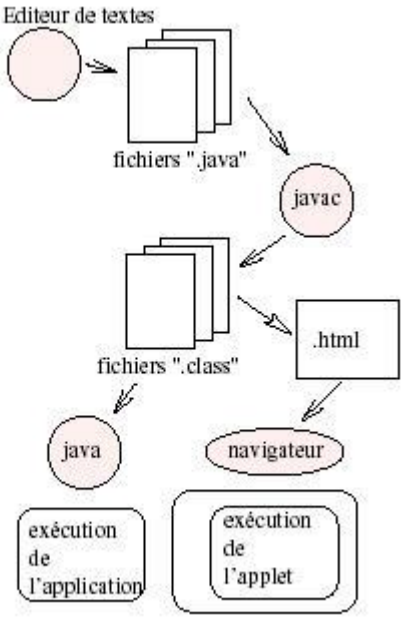
- Java est un langage totalement objet développé par la société SUN .
  L'environnement de développement de SUN se compose de :
  - **java runtime environnement** JRE composé de :
    machine virtuelle JVM
    commandes appletviewer, java, javaw, …
  - **java développement kit** JDK composé d'un ensemble de classes.
    Commandes java, javac, jar, javadoc…

- Compilation en JAVA :

Le code java compilé (byte-code) est indépendant du système sur lequel il sera interprété et exécuté. On dit que java est totalement 'portable'.
C'est la machine virtuelle (JVM) qui interprète le byte code en fonction du système.

- La plate-forme JAVA permet de construire des **applications complètes**, graphiques ou non ainsi que des **applets** (petits programmes s'exécutant sur les postes clients de sites web).

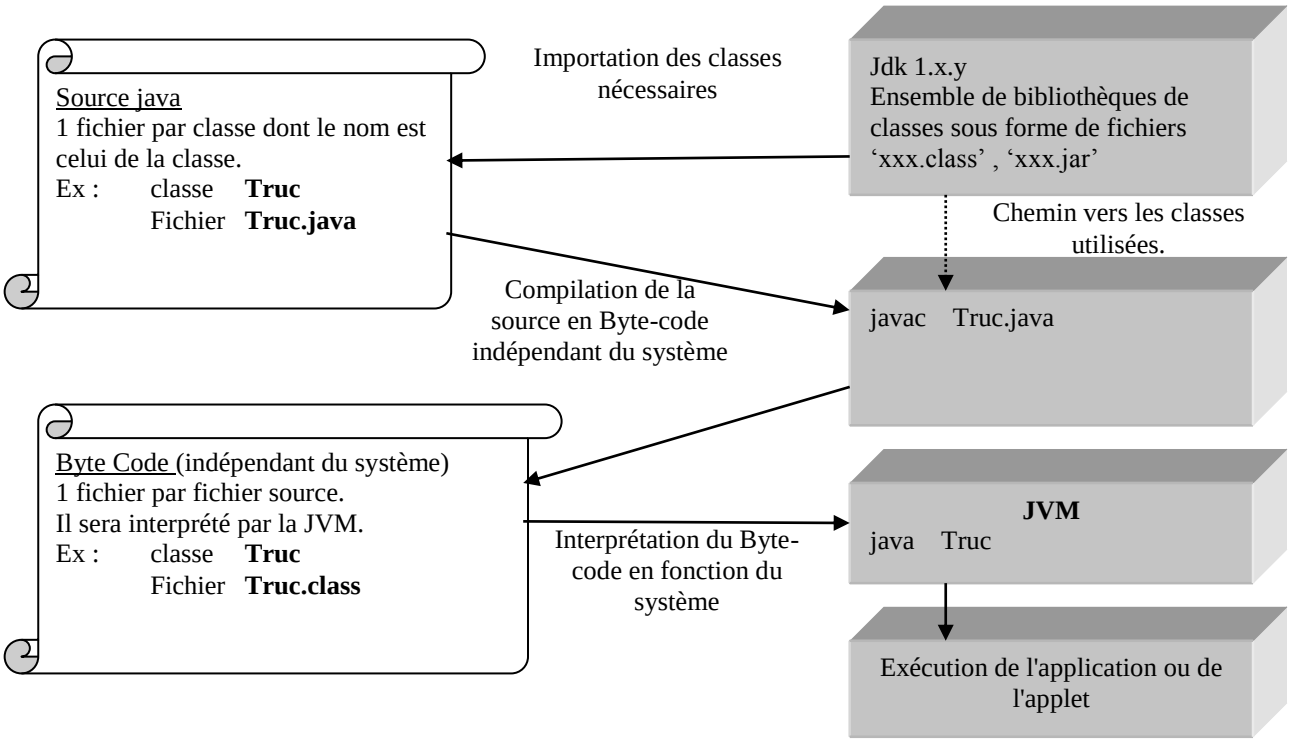
Schéma de développement :



- Java supporte directement les Threads. Les problèmes de multi-tâches peuvent donc être résolus indépendamment du système d'exploitation.

2. Développement en JAVA :

2.1. Structure des fichiers :



Les fichiers sources : extension **'.java'**
Ils contiennent les sources de chaque classe écrite en java.

- Le fichier porte le nom de la classe avec l'extension **.java**
- Une seule classe publique par fichier.

⇨ EX : classe **Individu** => 1 fichier **Individu.java**

Les fichiers compilés : extension **'.class'.**
Ils contiennent le résultat de la compilation du fichier source '.java' sous la forme d'un pseudo code portable sur d'autres environnements.

⇨ **Individu.class**

Les fichiers archives : extension **'.jar'**
Ils contiennent une ou plusieurs classes appartenant à une même famille logique de classes ou à des '**packages'**. Ils peuvent être exécutés ou non.

⇨ **MonProjet.jar**

2.2. Les commandes :

javac compile les fichiers sources '.java' en fichiers pseudo code '.class'
javac truc.java => construit truc.class

appletviewer permet de visualiser une applet après compilation.
appletviewer truc //ne pas écrire l'extension .class.

java permet l'exécution du fichier .class
java truc //ne pas écrire l'extension .class

javaw équivalent à java

jar permet d'archiver plusieurs classes [archives] en 1 seul fichier. Pour que ce soit un fichier exécutable, il faut ajouter un fichier manifeste nommé MANIFEST.MF contenant au moins le nom de la classe qui possède le ***main***.

Ex :

```
Main-Class: ecranClavier
Class-Path: ./projet.jar
```

javadoc construit une documentation sommaire (à compléter) au format html.

2.3. La machine virtuelle :

- La machine virtuelle est un interpréteur gérant l'exécution des fichiers compilés '.class' ou '.jar'.
- Outre l'exécution des '.class', elle gère aussi le chargement des classes, la vérification du code avant qu'il ne s'exécute (Policy) ainsi que la libération des ressources lorsqu'elles ne sont plus utiles (ramasse miettes ou Garbage Collector).
- Le pseudo code obtenu par la compilation des sources, étant le même quelque soit le système d'exploitation (le pseudo-code est portable), chaque système d'exploitation doit avoir sa propre machine virtuelle.

3] Exercices :

3.1] Afficher les paramètres de la ligne de commande :

Ces tests se font en mode console : ne pas utiliser un IDE.
Créer un répertoire nommé **java.**
Crée un fichier ecranClavier.java et y copier le code ci-dessous.
Ajouter dans votre système à la variable d'environnement le chemin vers le jdk/bin.

```
public class ecranClavier {
	//point d'entrée du programme
	public static void main(String[] args) {
		// TODO Auto-generated method stub
		System.out.println("Affichage des paramètres:");
		for(int i=0;i<args.length; i++)
			System.out.println(args[i]);
	}

}
```

- Compiler la classe :
  javac ecranClavier.java

Lire les messages dans la console et corriger les éventuelles erreurs.
Vérifier la création du fichier **ecranClavier.class.**

- Exécuter le code :
  java ecranClavier

puis

java ecranClavier toto titi tutu 12 47.25

En déduire le rôle et la nature du paramètre args.

- Modifier le code afin d'afficher le nombre de paramètres ainsi que leur valeur précédée d'un texte renseignant leur position tel que :

⇨ java -classpath ./java1 ecranClavier totot jghghgh kdkdkd
Argv[0]= totot
Argv[1]= jghghgh
Argv[2]= kdkdkd

3.2] Saisie et affichage :

Remplacer le code ci-dessus par celui-ci-dessous :

```
//import java.io.IOException;

public class ecranClavier {

	public static void main(String[] args)
	{
	byte tab[] = new byte[10];
	System.out.println("Entrer un texte:");
	try {
		int i = System.in.read(tab);
		String text = new String(tab,0,i);
		System.out.println("Texte:" + text);
		System.out.println("nb caracteres:" + i);
		System.out.println("Taille tableau:" + tab.length);
		System.out.println("Taille texte:" + text.length());
		for(int j=0;j<i;j++)
			System.out.print(tab[j]+" ");
		} catch (IOException e) {
			// TODO Auto-generated catch block
			e.printStackTrace();
			}
	}
}
```

- Compiler, corriger et exécuter ce code (taper un texte de 5 lettres)

⇨ En déduire le rôle de la ligne : **`import java.io.IOException;`**

- Analyser les affichages produits en fonction des frappes au clavier.

⇨ Pourquoi i est-il toujours = nb caractères frappés + 2 ?

⇨ A quoi correspondent ces deux caractères supplémentaires.

⇨ Que se passe-t-il si on frappe plus de 10 caractères et pourquoi ?

3.3] Archivage et exécution :

- Dans le même répertoire, créer le fichier correspondant au code ci-dessous :

```
public class Individu {
	public int age;
	public String nom;
	public void setAge(int a){
		age = a;
	}
	public void setNom(String n){
		nom = n;
	}
	public void visu(){
		System.out.println("Individu: \n\tnom: "+nom+"\n\tage="+age);
	}
}
```

- Puis, modifier la classe **ecranClavier** de la façon suivante : et créer le fichier manifest suivant(ci-dessous à droite) :

```
	public static void main(String[] args) {
		// TODO Auto-generated method stub
		Individu in1 = new Individu();
		Individu in2 = new Individu();
			in1.setNom("Grincheux");
			in1.setAge(12);
			in2.setNom("Simplet");
			in2.setAge(42);
			in1.visu();
			in2.visu();

	}
```

```
Main-Class: ecranClavier
Class-Path: ./projet.jar
```

- Compiler les deux sources.
- créer une archive contenant les deux classes et le fichier manifeste ci-dessus :

jar cvmf manifest.mf projet.jar *.class

- Ouvrir le fichier projet.jar avec un outil de décompression classique (winzip, winrar, 7zip, gunzip…) et vérifier le résultat.
- Exécuter l'archive :

java –jar projet.jar

3.4] Création d'une documentation :

Se rendre sur le site officiel de java pour connaître tous les tags et les règles d'écriture.

- Modifier la classe Individu de la façon suivante :

```
package premier;

/**
 * classe test de gestion d'un individu
 * @author IRIS AGORA
 * @version 1.0
 */
public class Individu {
	public int age;
	public String nom;
	/**
	 * Assesseur
	 * @param a le paramètre est un entier correspondant à l'age
	 */
```

```
	public void setAge(int a){
		age = a;
	}
	/**
	 * Assesseur
	 * @return age l'age de l'individu
	 */
	public int getAge(){
		return age;
	}
	public void setNom(String n){
		nom = n;
	}
	public void visu(){
		System.out.println("Individu: \n\tnom: "+nom+"\n\tage="+age);
	}
}
```

- Créer la documentation relative aux tags et commentaires insérés dans le source. Puis visualiser dans un navigateur les fichiers crées.

javadoc Individu.java