

IUT de Lens - BUT Informatique

2023/2024

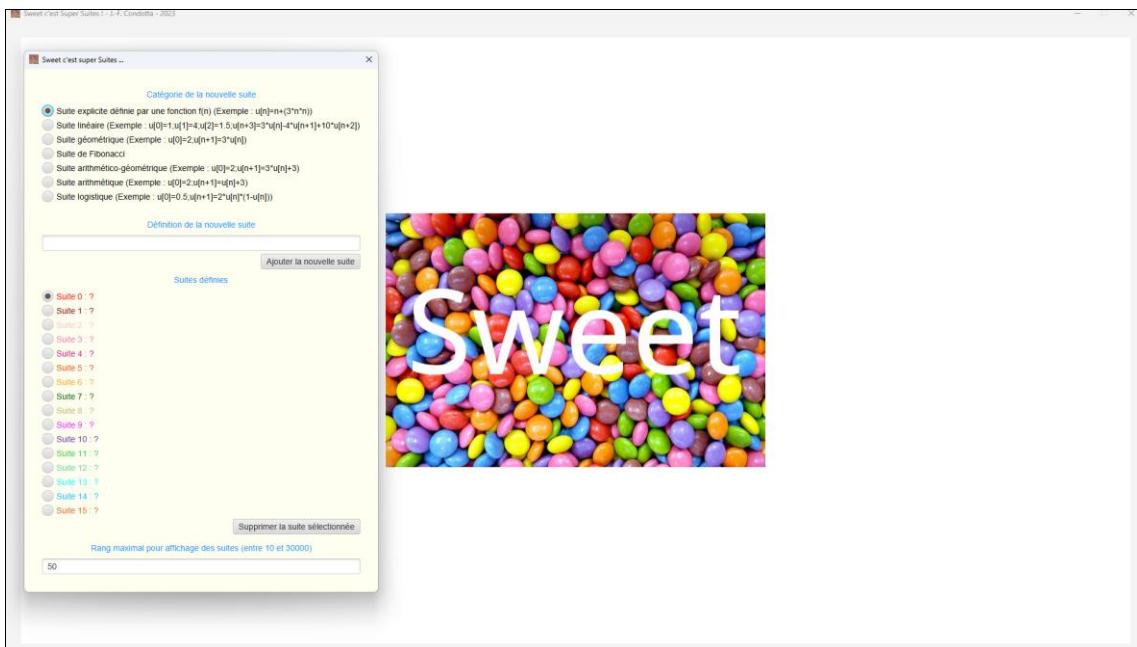
SAÉ S2.02

Exploration algorithmique d'un problème

Autour des suites : l'application Sweet

R2.01 Dév. Objets & R2.09 Méthodes numériques

Mardi 4 juin 2024 (8 h)



Contexte et Mission

Nous souhaitons réaliser une application appelée **Sweet** permettant de gérer et d'afficher des suites. Votre mission principale consistera à modéliser et planter un ensemble de classes permettant de manipuler différents types de suites. Pour développer l'application Sweet vous allez devoir réaliser des diagrammes de classe UML sous plantUML et programmer en Java sous Eclipse. Vous travaillerez sous Linux en utilisant les machines de l'IUT ou vos machines personnelles (au choix).

Pour réaliser cette SAE vous mettrez par groupe de 3 (le livrable **bilan.xlsx** devra contenir les prénoms et les noms des membres du groupe). Vous pouvez constituer les groupes comme vous le souhaitez. Dans le cas où vous n'êtes pas 3 **vous devez absolument** en référer les enseignants qui pourront éventuellement modifier la constitution des groupes.

Livrables

Les livrables demandés seront précisés dans la suite pour chacune des étapes. L'évaluation d'une étape avec un livrable manquant pourra entraîner une note de 0 à cette étape. Dans certains livrables vous pourrez indiquer que vous êtes faits aider par ou que vous avez aidé d'autres groupes. Dans le cas où il s'avèrerait que des informations soient manquantes (volontairement ou involontairement) dans ces livrables, une note de 0 sera donnée pour les aidés **et** les aidants pour l'ensemble de la SAE. Enfin, il est interdit d'utiliser **ChatGPT** ou d'une autre **IA** (une utilisation d'une IA entraînera un malus très important des évaluations des différentes étapes et donc de la SAE).

Les livrables devront être remis sous Moodle dans les temps. Les heures limites des remises des différents livrables devront être respectées. Néanmoins, plus tôt vous remettrez vos livrables, meilleure sera votre note dans le cas où les livrables demandés sont complets. Un seul étudiant par groupe remettra l'ensemble des livrables des différentes étapes.

Le sujet de la SAE proposé peut être a priori terminé à 14h30 mais les entrées pour remettre les livrables seront ouverts sous Moodle jusqu'à **17h30**.

Les enseignants sont à votre disposition pour répondre à toute question.

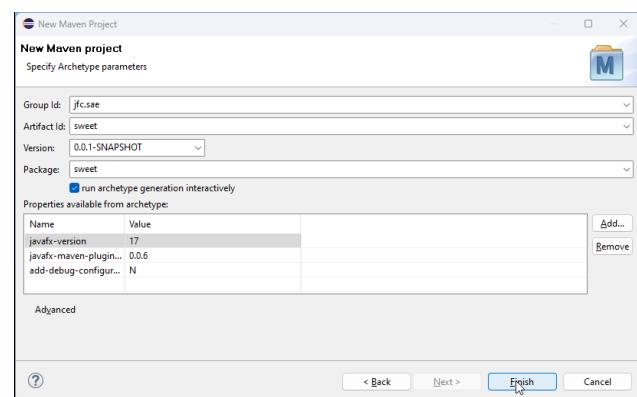
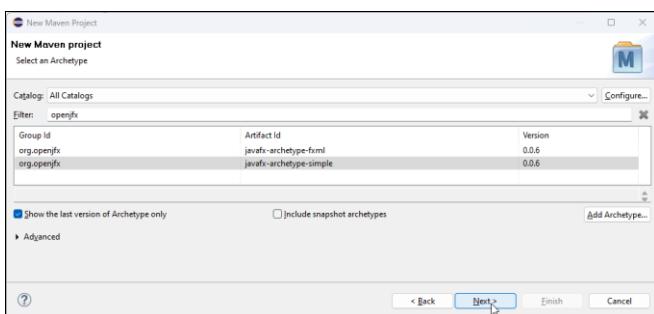
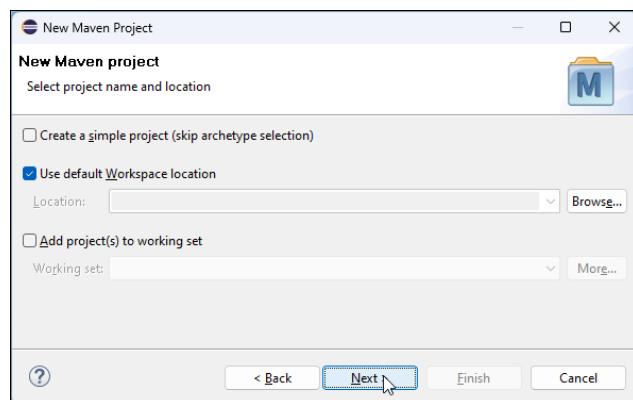
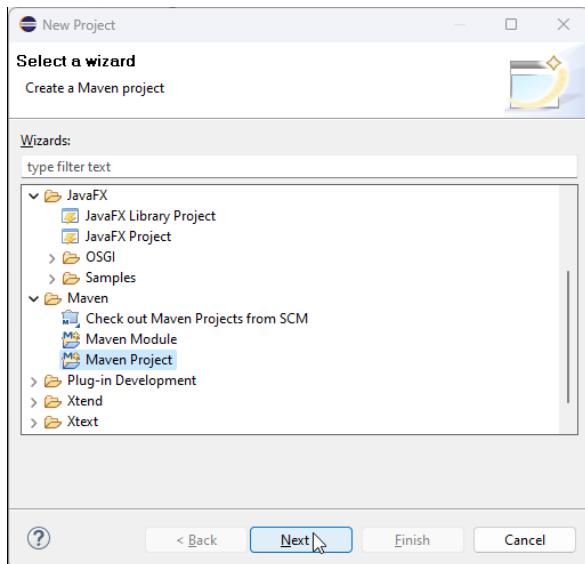
Bonne SAÉ à Vous !

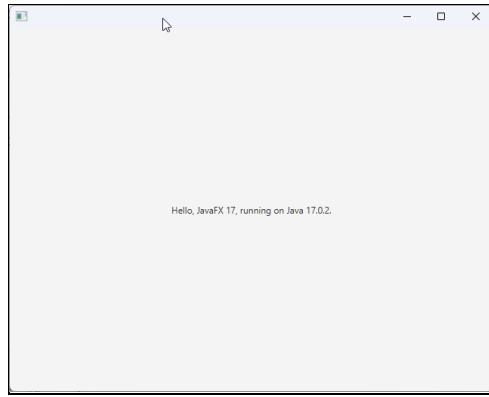
Etape 0 – Préliminaires

Vous travaillerez sous Linux, sur vos machines ou celles de l'IUT. À vous de répartir les différentes tâches à réaliser entre les différents membres du groupe.

L'application **Sweet** utilisera une interface graphique (que vous ne développerez pas) utilisant **JavaFX**. Vous allez devoir créer un projet Maven sous Eclipse afin d'importer la librairie JavaFX (et d'autres librairies à partir des dépôts distants de Maven).

- ☞ Afin de pouvoir utiliser des dépôts distants de Maven, copiez le fichier settings.xml disponible sous Moodle (fichier contenant la définition des proxies) dans un dossier réseau. Modifiez le paramètre **Window → Preferences → Maven → User Settings** afin d'indiquer le chemin vers ce fichier.
- ☞ Sous Eclipse, créez un projet Maven appelé Sweet en suivant les captures d'écran qui suivent. Après la création vous lancerez le programme **App.java** afin de tester que JavaFX est bien installé. Pour tout problème vous pouvez vous faire aider par un autre groupe.





La librairie **exp4j** (librairie permettant de définir et de calculer des fonctions mathématiques) sera également utilisée. Pour cela vous allez devoir modifier le fichier Maven **pom.xml**.

☞ Sous Eclipse, ouvrez le fichier **pom.xml** du projet **sweet** et ajoutez les lignes suivantes (**juste avant la ligne </dependencies>**) :

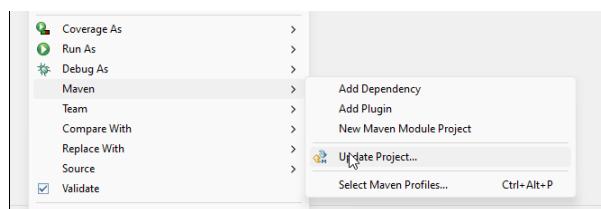
```
<dependency>
    <groupId>net.objecthunter</groupId>
    <artifactId>exp4j</artifactId>
    <version>0.4.8</version>
</dependency>
```



☞ Sous Eclipse, ajoutez la ligne **requires exp4j;** dans le fichier module-info.java.



☞ Sous Eclipse, mettez à jour le projet Maven sweet (clic droit sur le projet sweet → Maven → Update Project ...).



Remarque : Dans la suite, vous allez devoir réaliser des diagrammes de classe, pour se faire vous utiliserez l'outil plantUML.

- ☛ Récupérez sous Moodle le fichier **etudiants_etapes_0_5.zip** et faites l'extraction des différents fichiers de cette archive.
- ☛ Remplissez la première feuille du classeur Excel **bilan.xlsx** (Membres de votre groupe).

Etape 1 – La classe Suite

- ☛ Créez le paquetage **sweet.suites** et copiez dans ce paquetage la définition de la classe **Suite** (**Suite.java**).
 - ☛ Consultez le code source et la documentation de la classe **Suite**.
- La classe **Suite** est une classe abstraite qui sera la classe racine de toutes les classes représentant les différentes suites que nous considéreront par la suite.
- ☛ Complétez/modifiez le code source de la classe **Suite** afin de la compléter (les méthodes à compléter ou à modifier sont marquées par le commentaire //A compléter et/ou à modifier).
 - ☛ Créez un diagramme de classe UML dans un fichier appelé **etape_1_diagramme.puml** qui contiendra la seule classe **Suite**. **Remarque :** le type UML double sera utilisé pour la classe Double en Java (et les ArrayList<Double> seront considérés comme des tableaux de double en UML). Cette remarque vaudra pour la suite du sujet.
 - ☛ Vous déverserez les fichiers **etape_1_diagramme.puml** et **etape_1_diagramme.png** dans Moodle.
 - ☛ Complétez le fichier Excel **bilan.xlsx**.

Etape 2 – Les suites explicites

Les **suites explicites** sont des suites définies explicitement à partir de fonctions. Par exemple, la suite $u[n]=n+(3*n*n)$ est une suite explicite définie à partir de la fonction $f(n)=n+(3*n*n)$.

- ☛ Copiez dans le paquetage **sweet.suites** les différents fichiers **SuiteExplicite.java**, **Fonction.java** et **ExceptionFonction.java**.

La classe **Fonction** permet de définir des fonctions à partir de chaînes de caractères. La classe **ExceptionFonction** correspond aux exceptions levées lorsqu'une fonction est mal définie ou lorsqu'un problème de calcul d'une fonction survient. La classe **SuiteExplicite** permet de définir et calculer des suites explicites.

- ☛ Consultez le code source et la documentation des classes **SuiteExplicite**, **Fonction** et **ExceptionFonction**.
- ☛ Complétez/modifiez le code source de la classe **SuiteExplicite** afin de la compléter (les méthodes à compléter ou à modifier sont marquées par le commentaire //A compléter et/ou à modifier).
- ☛ Dupliquez le fichier **etape_1_diagramme.puml** en un nouveau fichier appelé **etape_2_diagramme.puml**. Vous ajouterez à ce nouveau diagramme les classes **SuiteExplicite**, **Fonction** et **ExceptionFonction**. **Remarque :** vous ne représenterez aucun attribut et aucune méthode dans la classe **Fonction**.
- ☛ Vous déverserez les fichiers **etape_2_diagramme.puml** et **etape_2_diagramme.png** dans Moodle.
- ☛ Créez le paquetage **sweet.tests** et copiez dans ce paquetage le fichier **TestSuites.java**.

- ☞ Décommentez le corps de la méthode **testEtape2** de la classe TestSuites et exécutez la classe afin de vérifier les classes Suite et SuiteExplicite. La vérification sera effectuée en réalisant une comparaison avec la trace d'affichage présente dans **trace_test_suites.pdf**.
- ☞ Complétez le fichier Excel **bilan.xlsx**.

Etape 3 – Précisions sur les nombres à virgule flottante en Java

Pour cette étape, aucun livrable n'est à fournir. Vous êtes simplement conviés à venir en amphi SP à une heure précise qui vous sera indiquée lors de la SAE (en attendant l'heure en question vous pouvez passer à l'étape 4). Une courte présentation sur les nombres à virgule flottante en Java sera effectuée. Les précisions données vous seront utiles pour la suite.

Etape 4 – Les suites récurrentes

Les **suites récurrentes** sont des suites dont les valeurs des premiers termes sont données de manière explicite. Le nombre de ces premiers termes dont la valeur est donnée explicitement est appelé **ordre** de la suite. Pour une suite récurrente, la valeur d'un terme dont le rang est supérieur ou égal à l'ordre de la suite est calculée à partir des valeurs des termes de rang inférieur. Il existe plusieurs familles de suites récurrentes, nous nous intéresserons à certaines d'entre elles dans les étapes suivantes.

- ☞ Copiez dans le paquetage sweet.suites le fichier **SuiteRecurrente.java**.

La classe **SuiteRecurrente** est une classe abstraite qui sera la classe racine des différentes classes représentant les différentes familles de classes récurrentes que nous considérerons par la suite.

- ☞ Consultez le code source et la documentation de la classe **SuiteRecurrente**.
- ☞ Complétez/modifiez le code source de la classe **SuiteRecurrente** afin de la compléter (les méthodes à compléter ou à modifier sont marquées par le commentaire **//A compléter et/ou à modifier**).
- ☞ Dupliquez le fichier **etape_2_diagramme.puml** en un nouveau fichier appelé **etape_4_diagramme.puml**. Vous ajouterez à ce nouveau diagramme la classe SuiteRecurrente.
- ☞ Vous déverserez les fichiers **etape_4_diagramme.puml** et **etape_4_diagramme.png** dans Moodle.
- ☞ Complétez le fichier Excel **bilan.xlsx**.

Etape 5 – Les suites récurrentes considérées (Modélisation)

Parmi les **suites récurrentes** nous considérerons dans la suite les familles de suites suivantes (ces familles correspondent à des classes du paquetage **sweet.suites**).

Les suites linéaires

Une **suite linéaire** récurrente u d'ordre p est définie par une relation de récurrence de la forme : pour tout $n \geq p$, le terme général $u[n+p]$ est défini par $u[n+p] = a_0*u[n] + a_1*u[n+1] + \dots + a_{p-1}*u[n+p-1]$ où a_0, a_1, \dots, a_{p-1} sont des coefficients constants. Par exemple, $u[0]=1, u[1]=4, u[2]=1.5, u[3]=3*u[0]-4*u[1]+10*u[2]$ correspond à une suite linéaire récurrente d'ordre 3.

La classe **SuiteLineaire** représentera les suites linéaires dans notre programme. Cette classe possèdera l'attribut et le constructeur suivant :

```
private double coefficients[ ];  
public SuiteLineaire(String chaineSuite,double valPremiersTermes[],double coefficients[ ]) {...}
```

Les suites géométriques

Une **suite géométrique** est une suite linéaire u d'ordre 1. Par exemple, $u[0]=1.5$ et $u[n+1]=3*u[n]$ correspond à une suite géométrique de raison 3 (la raison est le coefficient de $u[n]$).

La classe **SuiteGeometrique** représentera les suites géométriques dans notre programme. Cette classe ne possèdera pas d'attribut et possèdera le constructeur suivant :

```
public SuiteGeometrique(String chaineSuite,double valPremierTerme,double raison) {...}
```

Les suites arithmétiques

Une suite arithmétique est une suite récurrente u d'ordre 1 dont le terme général est défini par $u[n+1]=u[n]+r$ avec r une constante appelée la raison de la suite. Par exemple, $u[0]=1.5$ et $u[n+1]=u[n]+5$ correspond à une suite arithmétique de raison 5.

La classe **SuiteArithmetique** représentera les suites arithmétiques dans notre programme. Cette classe ne possèdera pas d'attribut et possèdera le constructeur suivant :

```
public SuiteArithmetique(String chaineSuite,double valPremierTerme,double raisonArithmetique) {...}
```

Les suites arithmético-géométriques

Une **suite arithmético-géométrique** est une suite récurrente u d'ordre 1 dont le terme général est défini par $u[n+1]=q*u[n]+r$ avec q et r deux constantes que nous appellerons respectivement raison géométrique et raison arithmétique. Par exemple, $u[0]=1.5$ et $u[n+1]=3*u[n]+5$ correspond à une suite arithmético-géométrique de raison géométrique 3 et de raison arithmétique 5.

La classe **SuiteArithmeticoGeometrique** représentera les suites arithmético-géométriques dans notre programme. Cette classe possèdera les attributs et le constructeur suivant :

```
private double raisonGeometrique;  
private double raisonArithmetique;  
public SuiteArithmeticoGeometrique(String chaineSuite,double valPremierTerme,double  
raisonGeometrique,double raisonArithmetique) {...}
```

La suite de Fibonacci

La **suite de Fibonacci** est la suite linéaire d'ordre 2 définie par $u[0]=0$, $u[1]=1$ et $u[n+2]=1*u[n]+1*u[n+1]$.

La classe **SuiteFibonacci** représentera la suite de Fibonacci dans notre programme. Cette classe ne possèdera pas d'attribut et possèdera le constructeur suivant :

```
public SuiteFibonacci() {...}
```

Remarque 1 : nous avons fait le choix de prendre une classe pour représenter la suite de Fibonacci (c'est un choix !).

Remarque 2 : La suite de Fibonacci sera toujours à la chaîne de caractères " $u[0]=0;u[1]=1; u[n+2]=1*u[n]+1*u[n+1]$ " pour la représenter, c'est pour cela que le constructeur n'a pas de paramètre String chaineSuite.

Les suites logistiques

Une **suite logistique** est une suite récurrente u d'ordre 1 dont le terme général est défini par $u[n+1]=A*u[n]*(1-u[n])$ avec A une constante qui doit être comprise entre 0.0 et 4.0. La valeur du terme $u[0]$ doit être comprise entre 0.0 et 1.0. Par exemple, $u[0]=0.5$ et $u[n+1]=2.0*u[n]*(1-u[n])$ correspond à une suite logistique avec 2.0 pour valeur de A .

La classe **SuiteLogistique** représentera les suites logistiques dans notre programme. Cette classe possèdera l'attribut et le constructeur suivant :

```
private double valA;  
public SuiteLogistique(String chaineSuite,double valPremierTerme,double valA) throws ExceptionSuite {...}
```

Remarque : La classe **ExceptionSuite** est une sous-classe de la classe **Exception** avec comme seul élément le constructeur public **ExceptionSuite(String message)** { super(message); }. Une **ExceptionSuite** est levée lorsque les paramètres du constructeur de **SuiteLogistique** ne sont pas valides.

- ☞ Dupliquez le fichier **etape_4_diagramme.puml** en un nouveau fichier appelé **etape_5_diagramme.puml**. Vous ajouterez à ce nouveau diagramme les classes correspondant aux différentes familles de suites considérées en ne prenant en compte que les informations données à cette étape (les classes n'auront que les constructeurs et éventuellement les attributs). Votre diagramme devra également contenir les relations d'héritages entre les différentes classes vous semblant les plus pertinentes.
- ☞ Vous déverserez les fichiers **etape_5_diagramme.puml** et **etape_5_diagramme.png** dans Moodle.
- ☞ Complétez le fichier Excel **bilan.xlsx** et déversez le dans Moodle.

Attention ! Vous avez jusqu'à 11 h 30 pour déverser l'ensemble des délivrables issus des étapes 0 à 5 !

Etape 6 – Les suites récurrentes considérées (Modélisation - Suite)

- ☞ Récupérez sous Moodle le fichier **etudiants_etapes_0_10.zip** et faites l'extraction des différents fichiers de cette archive.
- ☞ Copiez dans Eclipse les sources des classes suivantes : **SuiteLineaire**, **SuiteArithmetique**, **SuiteGeometrique**, **SuiteArithmeticoGeometrique**, **SuiteFibonacci**, **SuiteLogistique** et **ExceptionSuite**.
- ☞ Dupliquez le fichier **etape_5_diagramme.puml** en un nouveau fichier appelé **etape_6_diagramme.puml**. A partir des sources et de la documentation modifiez/rectifiez le diagramme afin d'obtenir un diagramme de classe UML correct par rapport à ces sources et cette documentation.
- ☞ Vous déverserez les fichiers **etape_6_diagramme.puml** et **etape_6_diagramme.png** dans Moodle.

Etape 7 – Les suites récurrentes (Implantation)

- ☞ Complétez/modifiez le code source des différentes classes importées sous Eclipse à l'étape précédente (les méthodes à compléter ou à modifier sont marquées par le commentaire **//A compléter et/ou à modifier**). Au fur et à mesure de votre implantation vous effectuerez des vérifications en décommentant les parties adéquates dans la classe **TestSuites**. L'affichage doit être exactement que celui se trouvant dans **trace_test_suites.pdf**.
- ☞ Complétez le fichier **bilan.xlsx** en inspectant la trace générée par l'exécution de la classe **TestSuites**.

Etape 8 – Ensemble de suites

☞ Copiez dans Eclipse le source de la classe **EnsembleSuites** dans le paquetage **sweet.suites**.

La classe **EnsembleSuites** permet de gérer un ensemble de suites dans un tableau.

☞ Dupliquez le diagramme de classe **etape_6_diagramme** en un nouveau diagramme de classe appelé **diagramme_final**. Ajoutez la classe EnsembleSuites à ce diagramme.

☞ Vous exporterez le diagramme **diagramme_final** au format .png et déverserez **diagramme_final.png** dans Moodle.

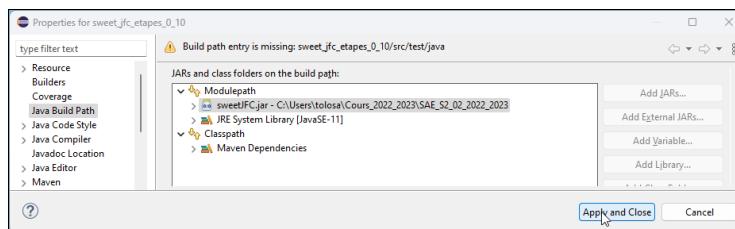
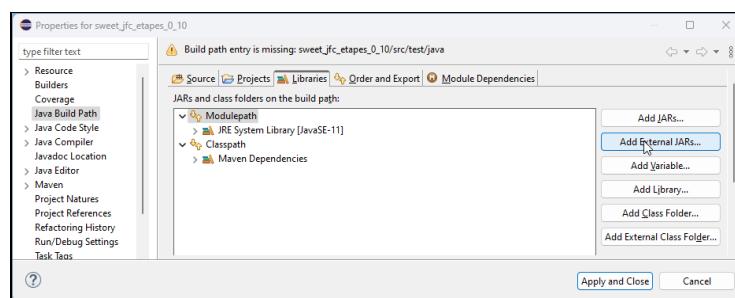
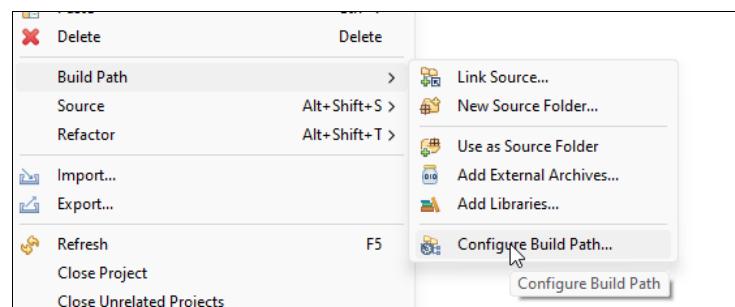
☞ Complétez/modifiez la classe **EnsembleSuites** (les méthodes à compléter ou à modifier sont marquées par le commentaire **//A compléter et/ou à modifier**).

Etape 9 – Affichage de suites

Nous allons maintenant incorporez le programme principal (la classe **Sweet**). Afin de le faire fonctionner vous devrez incorporez les classes correspondant à l'interface graphique (l'archive externe **sweetJFC.jar**).

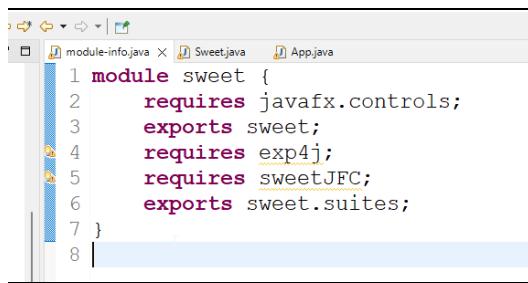
☞ Copiez dans le paquetage **sweet** la classe **Sweet** (le fichier **Sweet.java**). Tant que l'archive externe **sweetJFC.jar** n'est pas incorporée des erreurs seront présentes.

☞ Incorporez l'archive externe **sweetJFC.jar** au projet sweet. Pour cela réalisez un clic droit sur le projet sweet puis **Build Path → Configure Build Path ...** et suivez les captures d'écran suivantes.



☞ Modifiez maintenant le fichier **module-info.java** en y ajoutant les lignes suivantes :

```
requires sweetJFC;  
exports sweet.suites;
```



```
module sweet {  
    requires javafx.controls;  
    exports sweet;  
    requires exp4j;  
    requires sweetJFC;  
    exports sweet.suites;  
}
```

⇨ Vous pouvez maintenant exécuter la classe Sweet ☺.

⇨ Lancez Sweet et réalisez l'affichage des suites suivantes :

Suite 0 : la suite géométrique $u[0]=20; u[n+1]=1.1*u[n]$

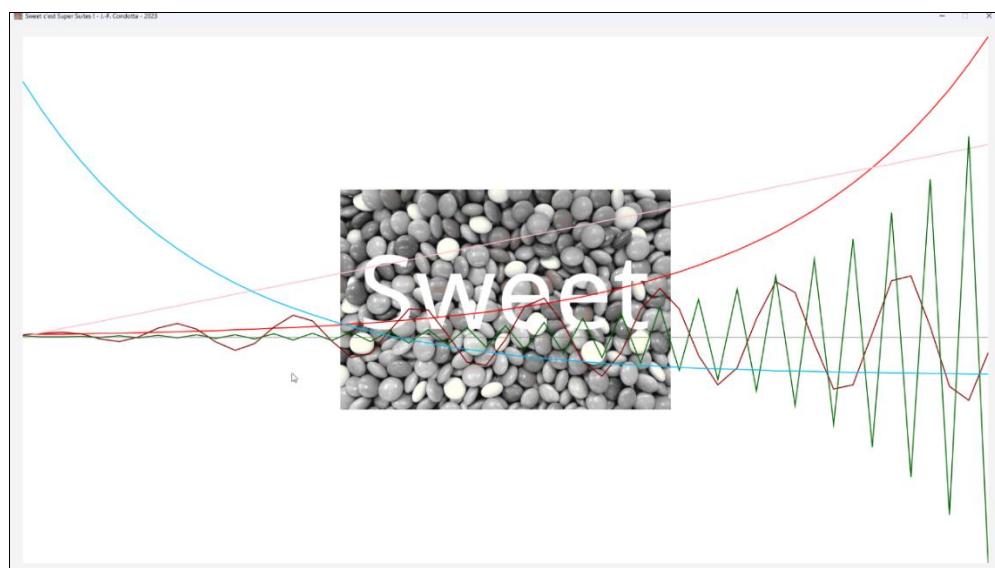
Suite 1 : la suite explicite $u[n]=20+11*\sin(n)*n$

Suite 2 : la suite arithmétique $u[0]=5; u[n+1]=u[n]+30$

Suite 7 : la suite linéaire $u[0]=10; u[1]=2; u[2]=3; u[n+3]=0.75*u[n]+1*u[n+1]-0.83*u[n+2]$

Suite 14 : la suite arithmético-géométrique $u[0]=2000; u[n+1]=0.9*u[n]-30$

⇨ Réalisez une capture d'écran dans un fichier image appelé **capture_1_etape_9.png** et déverserez ce fichier sous Moodle.



⇨ Stoppez et relancez Sweet et réalisez l'affichage des suites suivantes :

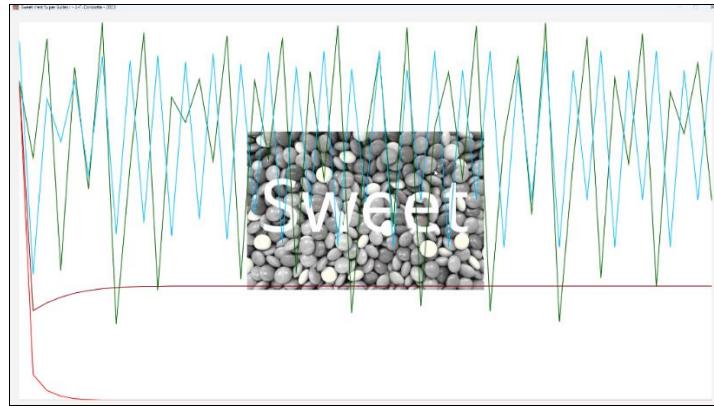
Suite 0 : la suite logistique $u[0]=0.8; u[n+1]=0.4*u[n]*(1-u[n])$

Suite 1 : la suite logistique $u[0]=0.8; u[n+1]=1.4*u[n]*(1-u[n])$

Suite 7 : la suite logistique $u[0]=0.8; u[n+1]=3.8*u[n]*(1-u[n])$

Suite 14 : la suite logistique $u[0]=0.9; u[n+1]=3.5*u[n]*(1-u[n])$

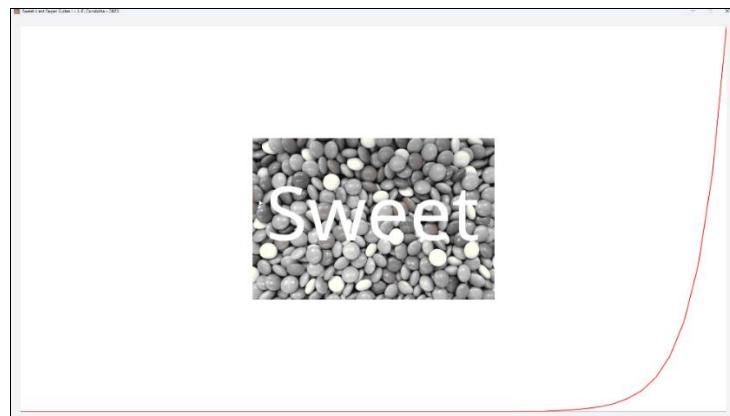
⇨ Réalisez une capture d'écran dans un fichier image appelé **capture_2_etape_9.png** et déverserez ce fichier sous Moodle.



☞ Stoppez et relancez Sweet et réalisez l'affichage de la suite de Fibonacci :

Suite 0 : la suite logistique de Fibonacci

☞ Réalisez une capture d'écran dans un fichier image appelé **capture_3_etape_9.png** et déversez ce fichier sous Moodle.



Etape 10 – Pour finir ...

☞ Sous Eclipse exportez les sources de l'ensemble des classes de l'application sweet dans un fichier appelé **sources_sweet.zip**. Déversez ce fichier dans Moodle.

☞ Complétez le fichier **bilan.xlsx** et déversez le dans Moodle.

☞ Une fois que le tout a été déversé sous Moodle appelez un des enseignants pour effectuer une démonstration.

Le temps optimal de fin de la SAE est fixé à 14h30 mais vous pourrez toujours déverser vos délivrables jusqu'à 17h30 (heure limite).

Bon courage à Tous !