Premières notions de programmation

Exercices de programmation en Java

(Auteur: E. Thirion - cours.thirion.free.fr
dernière mise à jour 22/02/2019)

Les exercices suivants sont en majorité des projets à complèter. L'interface graphique de ces projets est déjà réalisée ce qui vous permettra un gain de temps important. Ces projets sont disponibles par téléchargement. Pour savoir comment y accéder cliquez ici.

D'autre part, ce document fait partie d'un ensemble de cours du même auteur (programmation procédurale, programmation objet, programmation web, bases de données) auxquels vous pouvez accéder en cliquant ici.

Exercice 1 : Afficher-Effacer

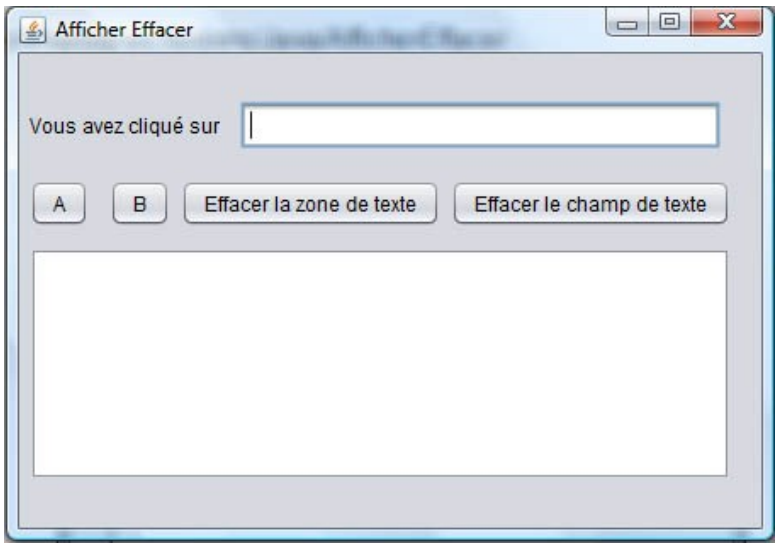
Objectif

Le but de cet exercice est de vous familiariser avec les notions de procédures évènementielles et de composant (champ de texte, boutons et zone de texte) et les instructions **es.Afficher** et **es.Effacer** .

Nous allons voir ici comment afficher ou effacer du texte dans un champ ou une zone de texte.

Projet à ouvrir : Exo-Java-Premieres-Notions/AfficherEffacer

Fenêtre de l'application



Cette fenêtre contient un champ de texte nommé **ChampTexte**, quatre boutons et une zone de texte nommée **ZoneTexte**.

Travail à réaliser

Pour l'instant, les procédures évènementielles associées aux boutons ne sont pas encore écrites.

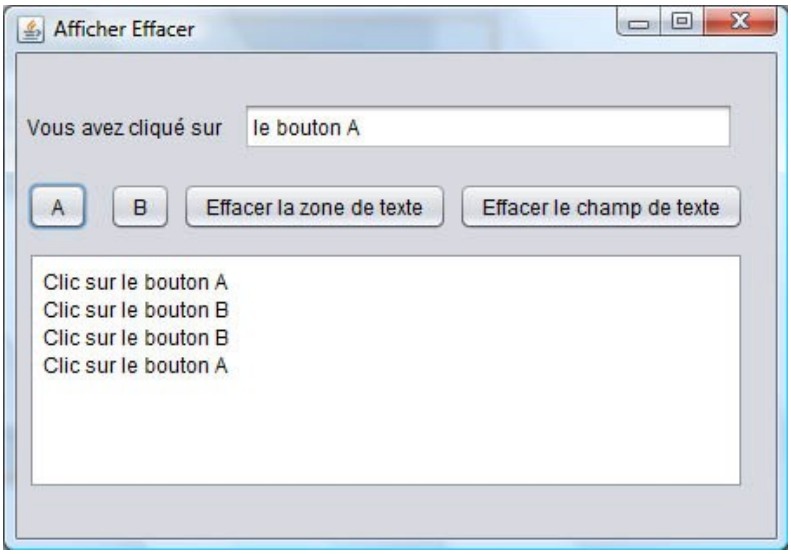
Votre travail consistera à écrire le code de ces procédures de manière à ce que

- le programme affiche dans le champ de texte et dans la zone de texte le nom du bouton qui a été

appuyé.

- les boutons d'effacement du champ de texte et de la zone de texte fonctionnent.

Voici, par exemple ce que devrait afficher votre programme, lorsque l'utilisateur clique sur le bouton A, puis deux fois sur le bouton B, puis sur le bouton A:



Remarque importante: ce programme n'utilise aucune variable. Il ne s'agit donc pas ici d'afficher des valeurs de variables, mais simplement des chaines de caractères. Si vous suivez le plan de travail, vous avez lu la partie du cours concernant la lecture et l'affichage, dans laquelle je dis que l'instruction **es.Afficher** permet d'afficher la valeur d'une variable. En réalité **es.Afficher** permet également d'afficher des chaines de caractères (nous parlerons de ceci dans la partie du cours consacrée aux expressions et plus particulièrement à l'affichage d'expression). Par exemple, pour afficher "Clic sur le bouton A" dans la zone de texte **ZoneTexte**, on écrira:

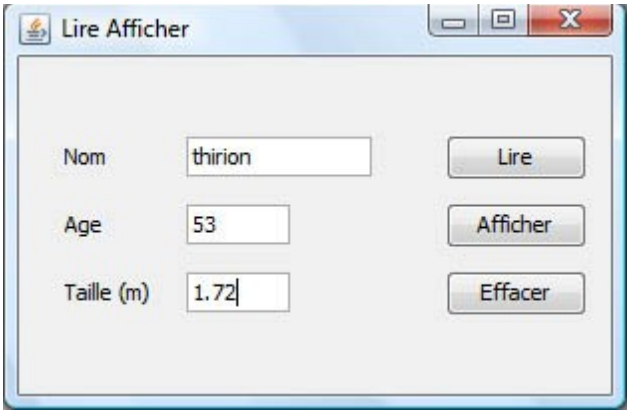
```
es.Afficher ("Clic sur le bouton A" , ZoneTexte);
```

Exercice 2 : Lire-Afficher

Parties du cours utilisées: déclaration de variables, type, la lecture et l'affichage.

Projet à ouvrir : Exo-Java-Premieres-Notions/LireAfficher

La fenêtre de l'application



Les champs de texte contenant le nom, l'age et la taille s'appellent respectivement **CT_Nom** , **CT_Age** et **CT_Taille**.

Travail à réaliser

Déclarez trois variables pour mémoriser le nom, l'age et la taille. A vous de choisir le type adéquat.

Ecrivez ensuite le code associé à chaque bouton en suivant les indications données en commentaire.

Pour tester votre programme effacer les trois champs de texte (bouton **Effacer**) juste après avoir effectué une lecture (bouton **Lire**). Si tout fonctionne bien, les trois valeurs lues doivent alors réapparaitre dans les champs de texte, preuve qu'ils ont bien été enregistrés en mémoire.

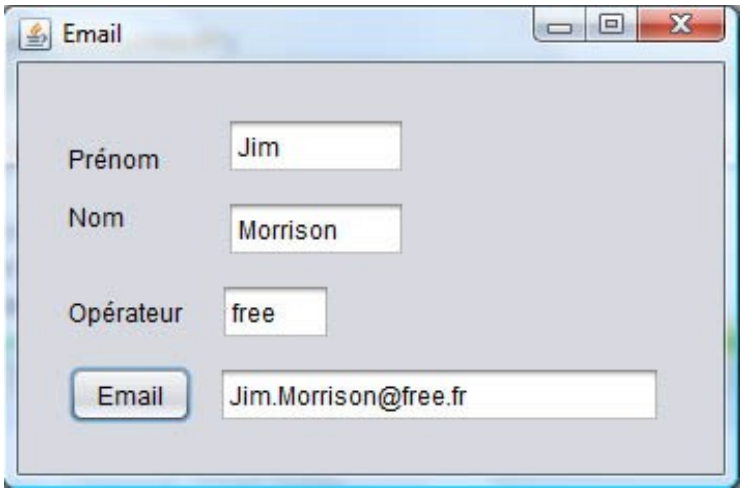
Attention: pour saisir un nombre à 'virgule', il faut utiliser le point. Par exemple pour saisir une taille de 1m72, il faudra taper 1.72 et non pas 1,72 !

Exercice 3 : Afficher un email

Parties du cours utilisées: déclaration de variables, type, lecture et l'affichage, expression, affichage d'expression.

Projet à ouvrir : Exo-Java-Premieres-Notions/Email

La fenêtre de l'application



Nom des champs de texte de haut en bas: **CT_Prenom**, **CT_Nom**, **CT_Op**, **CT_Email**.

Travail à faire

On suppose que l'adresse email est toujours de la forme *prenom**.**nom@opérateur**.fr***.

Complètez le programme de manière à ce qu'un clic sur le bouton **Email** provoque l'affichage du mail de la personne.

Exercice 4 : Calcul de salaire

Parties du cours utilisées: déclaration de variables, type, lecture et l'affichage, expression, affichage d'expression.

Projet à ouvrir : Exo-Java-Premieres-Notions/Salaire

La fenêtre de l'application

Noms des champs de texte de haut en bas: **CT_THB**, **CT_NH**, **CT_Brut**, **CT_Net**.

Travail à faire

Complètez le programme de manière à ce qu'un clic sur le bouton provoque l'affichage du salaire brut et net. Les données sont donc le nombre d'heures et le taux horaire. On supposera que ce sont des entiers.

Pour le calcul du salaire net, on prendra 75% du brut.

Pour tester votre programme, vous pouvez prendre les valeurs affichées par la copie d'écran ci-dessus.

Exercice 5 : Frais de déplacement

Parties du cours utilisées: déclaration de variables, type, lecture et l'affichage, expression, affichage d'expression, affectation, méthodologie.

Cahier des charges

Une entreprise souhaiterait disposer d'un logiciel permettant d'estimer les frais de déplacement de ses employés.

L'utilisateur précise:

- la distance parcourue entre le domicile et le lieu de travail.
- le nombre de trajets.
- la consommation de la voiture (nombre de litres pour 100 km).
- le prix du carburant utilisé (prix du litre)

En prendra également en compte l'usure des pneus que l'on supposera égale à 2 cents par kilomètre.

Dans cet exercice vous réaliserez vous même l'interface graphique. Le projet est donc à réaliser de A à Z. Reportez vous à la notice d'utilisation de Netbeans de ce cours pour savoir comment procéder.

Pour tester votre programme : vous devez obtenir des frais de déplacement de 110 € pour une distance de 20 km, une consommation de 6 litres pour 100 km, un carburant à 1.5 € / litre et 50 trajets.

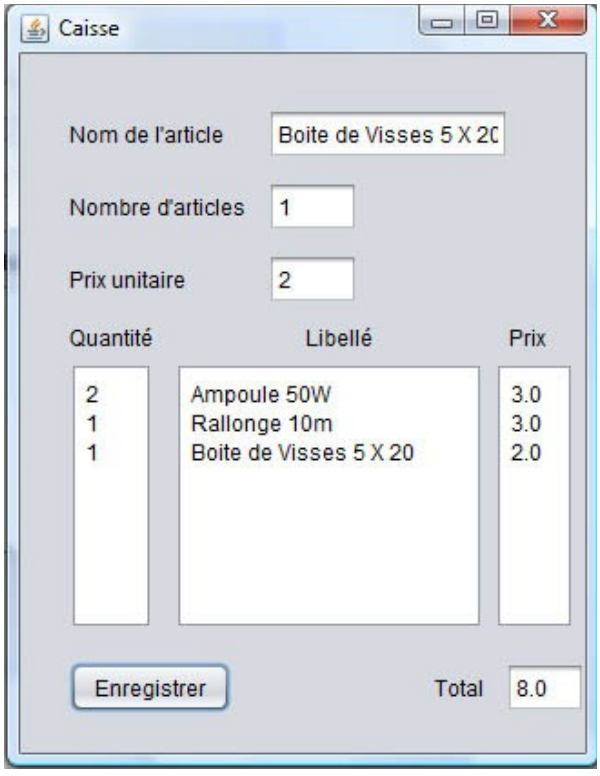
Attention: la division d'un entier par un entier est un entier. Par exemple 6 / 100 sera égal à 0 et non pas à 0.06. Pour obtenir le bon résultat, on peut par exemple écrire 100.0 à la place de 100

Exercice 6 : Caisse de magasin

Parties du cours utilisées: déclaration de variables, type, lecture et l'affichage, expression, affichage d'expression, affectation, traitement itératif.

Projet à ouvrir : Exo-Java-Premieres-Notions/Caisse

La fenêtre de l'application



Noms des composants:

- Champs de texte : **CT_Nom** (Nom de l'article), **CT_NA** (Nombre d'article), **CT_PU** (Prix Unitaire), **CT_Total** (Total)
- Zones de texte: **ZT_Quant** (Quant), **ZT_Libelle** (Libellé), **ZT_Prix** (Prix)

Utilisation du programme

Ce programme simule une caisse de magasin.

L'utilisateur saisi le nom de l'article, la quantité d'articles achetés et le prix unitaire. Lorsqu'il clique sur **Enregistrer**:

- le nombre d'articles achetés est affiché dans la zone de texte **ZT_Quant**.
- le nom de cet article est affiché dans la zone de texte **ZT_Libelle**.
- le prix à payer pour cet article (nombre d'articles X prix unitaire) est affiché dans la zone de texte **ZT_Prix**.
- le prix total à payer est affiché dans le champ de texte **CT_Total**.

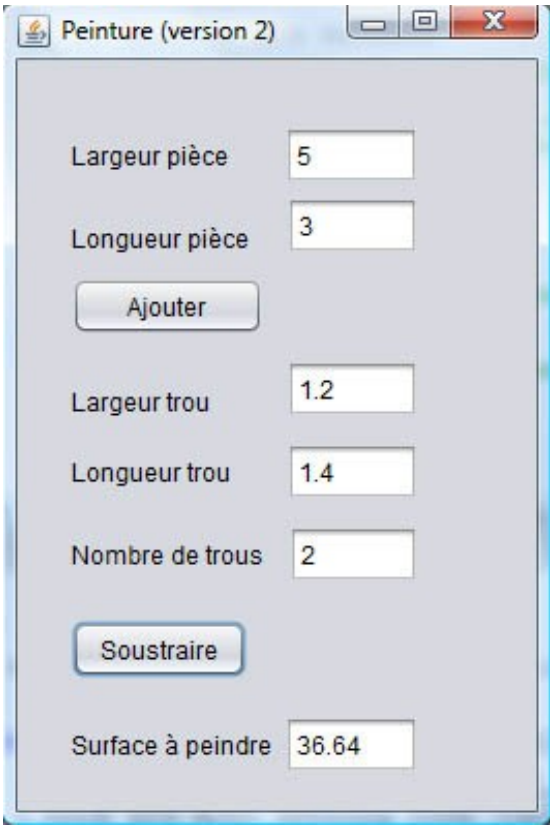
Pour réaliser ce traitement itératif, vous pouver mémoriser le prix total dans une variable qui sera mise à jour à chaque itération.

Exercice 7 : Surface à peindre

Parties du cours utilisées: déclaration de variables, type, lecture et l'affichage, expression, affichage d'expression, affectation, traitement itératif

Projet à ouvrir : Exo-Java-Premieres-Notions/Peinture2

La fenêtre de l'application



Noms des champs de texte de haut en bas: **CT_LargP**, **CT_LongP**, **CT_LargT**, **CT_LongT, CT_NbrTrou**, **CT_Surface**.

Utilisation du programme

Ce programme est une amélioration du programme vu en cours, car il permet de calculer la surface à peindre pour plusieurs pièces avec des portes et fenêtres de dimensions variables.

Une porte ou fenêtre est considérée comme un trou rectangulaire à soustraire de la surface à peindre.

On supposera que les murs ont une hauteur de 2m50 et que toutes les pièces sont rectangulaires.

Chaque pièce augmente la surface à peindre. Le bouton **Ajouter** sert donc à ajouter la surface des murs d'une pièce à la surface à peindre.

Chaque trou diminue la surface à peindre. L'utilisateur saisi les dimensions du trou et le nombre de trous de cette dimension (utile s'il y a plusieurs fenêtres ou portes de même dimensions), puis clique sur **Soustraire**. La surface totale de ces trous est alors enlevée de la surface à peindre.

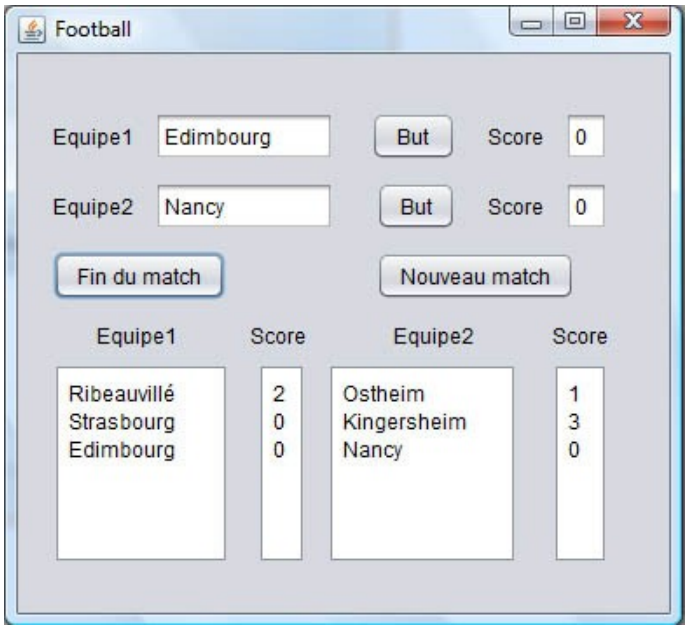
Pour réaliser ce traitement itératif, vous pouver mémoriser la surface à peindre dans une variable qui sera mise à jour à chaque itération.

Exercice 8 : Matchs de football

Parties du cours utilisées: déclaration de variables, type, lecture et l'affichage, expression, affichage d'expression, affectation, traitement itératif

Projet à ouvrir : Exo-Java-Premieres-Notions/Football

La fenêtre de l'application



Champs de texte: CT_Equipe1, CT_ScoreEquipe1, CT_Equipe2, CT_ScoreEquipe2

Boutons: BT_ButEquipe1, BT_ButEquipe2, BT_FinMatch, BT_NouveauMatch

Zones de texte: ZT_Equipe1, ZT_ScoreEquipe1, ZT_Equipe2, ZT_ScoreEquipe2

Utilisation du programme

Au début l'utilisateur lance le programme et saisi les deux noms d'équipe du premier match. Chaque fois qu'une équipe marque un but, il clique sur le bouton **But** de cette équipe. Lorsque le match est terminé, il clique sur le bouton **Fin du match**. A ce moment là, le score du match vient s'afficher dans les zone de texte.

Pour ajouter un nouveau match, il clique sur le bouton **Nouveau match**, ce qui a pour effet d'effacer les noms des équipes du match précédent et d'afficher un score nul.

Pour réaliser ce traitement itératif, vous pouver mémoriser les scores des deux équipes dans deux variables qui seront mise à jour à chaque itération.

Exercice 9 : Réalisation d'une calculatrice

Parties du cours utilisées: déclaration de variables, type, lecture et l'affichage, expression, affichage d'expression, affectation, traitement itératif.

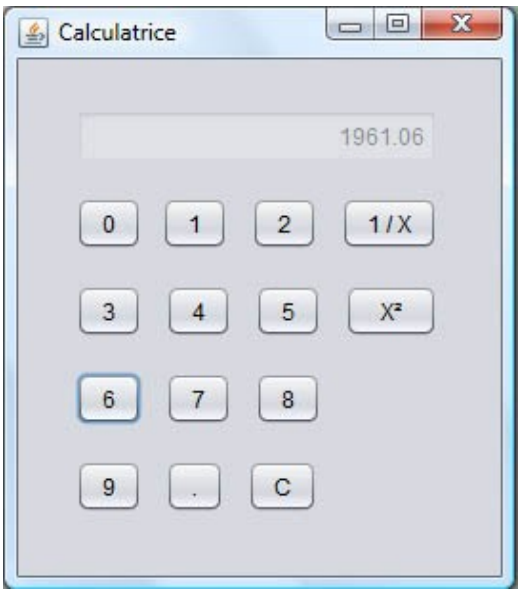
Intérêt pédagogique : cet exercice vous montrera quelques aspects des limitations d'un ordinateur dans les calculs et comment java les gère. Vous verrez également comment des calculs peuvent provoquez des erreurs d'exécution. Cet exercice vous aidera aussi à mieux comprendre la différence entre les types **int** et **double**.

Objectif

Il s'agit de réaliser une calculatrice avec clavier numérique permettant de calculer l'inverse ou le carré d'un nombre.

Projet à ouvrir : Exo-Java-Premieres-Notions/Calculatrice

Fenêtre de l'application



Il comporte 14 boutons et un champ de texte.

Les boutons des trois colonnes de gauche constituent le clavier numérique et comportent:

- 10 boutons étiquetés par les chiffres 0 à 9.
- un bouton étiqueté par un point, pour saisir les nombres non entiers.
- un bouton étiqueté **C** permettant d'effacer l'affichage.

Les deux derniers boutons sont des touches de calcul:

- la touche étiquetée **1/X** permet de calculer l'inverse du nombre affiché.
- la touche étiquetée **X ^{2}** permet de calculer son carré.

Le champ de texte pour l'affichage des chiffres est nommée **CT_Nombre**. Il a été défini de manière à ce que l'affichage des chiffres soit automatiquement cadré à droite et qu'on ne puisse pas y entrer du texte.

Travail à Faire

1) Clavier et Affichage

Il s'agit ici de faire fonctionner la saisie et l'affichage d'un nombre à l'aide des touches du clavier numérique.

Lorsque l'utilisateur clique sur un chiffre ou le point, celui-ci doit venir s'ajouter à la fin de la chaine de caractères déjà affichée. Par exemple, si le nombre actuellement affiché est 666 et qu'il clique sur 1, il obtiendra l'affichage du nombre 6661.

Pour réaliser ceci, vous utiliserez uniquement la variable **Affichage** de type **String**.

Indication : pour ajouter un chiffre à la fin du nombre affiché, utilisez une affectation récursive.

2) Touches de calcul

Toutes les opérations de calcul se feront à l'aide de la variable **X** (de type **double**) déjà déclarée. N'utilisez aucune autre variable.

Lorsque l'utilisateur clique sur une des deux touches de calcul, il doit se passer la chose suivante:

- le nombre affichée est affecté à **X**.
- le résultat de l'opération sur **X** est affecté à **X** (affectation récursive)
- la valeur de **X** est affichée.

3) Expériences amusantes et enrichissantes

Lorsque votre calculatrice fonctionnera correctement, je vous propose de faire quelques expériences qui vous ferons comprendre les limites de la représentation des nombres dans un ordinateur et également ce qu'est une erreur d'exécution.

a) Vers l'infiniment petit

Calculez l'inverse de 2. Si votre calculatrice fonctionne correctement vous obtenez 1/2 = 0.5. Elevez ensuite ce nombre au carré, puis à la puissance 4, puis à la puissance 8, etc en cliquant sur la touche x^2 . Vous obtiendrez ainsi, des nombres de plus en plus petits ($(1/2)^4$ =1/16, $(1/2)^8$ = 1/256, $(1/2)^{16}$=1/65536, ...). Au bout d'un certain temps vous constaterez que l'affichage du nombre passe en mode scientifique. Lorsque le nombre devient si petit qu'il dépasse la limite de précision du type double, le résultat est assimilé à 0.

b) Vers l'infiniment grand

Partez du nombre 2 et élevez le au carré, puis à la puissance 4, puis à la puissance 16 .... etc, en continuant à cliquer sur la touche x^2. Vous obtiendrez ainsi des nombres de plus en plus grands (2^4 = 16, 2^8 = 256, 2^{16}= 65536, ...). Au bout d'un certain temps vous constaterez que l'affichage du nombre passe en mode scientifique. Lorsque le nombre devient si grand qu'il dépasse la limite de précision du type **double**, surprise ! Le résultat affiché est **Infinity**. Cela signifie que le résultat est assimilé à l'infini.

Vous constaterez également qu'il est possible de continuer les calculs avec **Infinity**. L'inverse de **Infinity** vous donnera 0. Le carré de **Infinity** vous donne **Infinity**.

Il est également intéressant de constater que l'inverse de 0 donne **Infinity**.

Dans d'autre langages de programmation ces calculs (dépassement de capacité, division par 0) auraient

produits une erreur d'exécution.

c) Erreur d'exécution sur le format des nombres

A l'aide du clavier numérique de votre calculatrice, vous pouvez très bien former une chaine de caractères qui ne représdente pas un nombre. Par exemple 06.06.1961. Si vous saisissez un tel nombre et que vous effectuez un calcul dessus, vous obtiendrez une erreur d'exécution et la JVM vous afffiche un message d'erreur ressemblant à celui ci:

```
Exception in thread "AWT-EventQueue-0" java.lang.NumberFormatException: multiple points
```

d) Expérience avec le type int

Changez le type de X en **int**. et modifiez également les instructions de lecture du nombre X pour les adapter aux nombres entiers. Vous constaterez alors que les calculs avec le type **int** se comportent de manière très différentes que les calculs avec le type **double**.

- Inversez le nombre 2 : vous obtenez 0 ! Explication: la division 1 / 2 est ici interprétée comme une division euclidienne.

- Inversez le nombre 0. Vous n'obtenez pas **Infinity** comme avec les **double**. Cette fois ci, vous avez une erreur d'exécution avec un message du type:

  ```
  Exception in thread "AWT-EventQueue-0" java.lang.ArithmeticException: / by zero
  ```

- Elevez 2 au carré, puis à la puissance 4, puis à la puissance 8, .. etc ... comme précédemment. Vous constaterez que lorsque vous atteignez la puissance 32, le résultat est 0 ! Explication: 2 puissance 32 n'est pas représentable avec le codage du type int (nombres entiers signés sur 32 bits). Lorsque vous dépassez ce nombre, les résultats des calculs sont faux. On peut même obtenir des nombres négatifs en refaisant la même expérience en partant de 3 ! Ici aussi Java se distinguent d'autre langage en de produisant pas d'erreur d'exécution pour ces calculs erronés.