Trouver et corriger une erreur dans NetBeans

NetBeans est un IDE (Integrated Development Environment) qui nous aide à programmer. Entre autres, le programme nous montre les erreurs de syntaxe (pendant l'édition) et d'exécution. Connaissant les différents messages d'erreur de NetBeans nous aidera à corriger rapidement nos erreurs et d'avancer plus vite.

A] Les erreurs de syntaxe et de structure.

Il s'agit d'erreurs dues au non-respect du langage de programmation. NetBeans souligne en rouge les termes/lignes qu'il n'a « pas compris » en expliquant si possible ce qui ne va pas. Le message d'erreur peut être lu en survolant la partie soulignée en rouge avec la souris.

`private int x`
`private int y;`
';' expected
 	
autre exemple :
`public void draw(Graphics g) {`
`  for (int i=0,i<trucs.size(),i++ )`
`  trucs.get(i).draw(g);`
`}`
	Le programme s'est attendu à un ';' mais a rencontré *private.*
	
	utilisation de la virgule au lieu du point-virgule

`g.fillRect (0,0,getWidth(),getHeight();` **')' expected**	Le programme s'est attendu à un ')' mais a rencontré ';'. La parenthèse finale manque.

`public Truc( int pX, int pY) {`
`  x=pX;`
`  z=pY;`
`}`
cannot find symbol
symbol : variable z
location : class Truc
 	
autre exemple :
`g.fillRect (0,0,getWidth,getHeight());`
autre exemple :
`g.setColor(Color.WHITE);`
	La variable z n'a pas été définie (comme attribut, paramètre ou variable locale), donc il ne la connaît pas.
	
	La méthode a été utilisée sans parenthèses, donc comme un attribut (qui n'existe pas).
	
	Import manque
	La classe Color doit encore être importée.

`public String toString() {`
`  return "Truc (x=" + x + ", y=" + y + ");`
`}`
unclosed string literal
	Il manque le dernier guillemet, donc la dernière chaîne de caractères n'est pas fermée.

`public Color getColor() {`
`  Color col=Color.GREEN;`
`  if (value<30) col= Color.RED;`
`  else if (value<35) col = Color.YELLOW;`
`  return col;`
`  value=value+1;`
`}`
Unreachable statement
	L'instruction à la ligne 55 ne sera jamais atteinte, car il y a un return avant.

`public String getAppreciation() {`
`  if (value<30) return "insuffisant";`
`  if (value>=30) return "suffisant";`
`}`
Missing return statement
	Le compilateur n'a pas pu déterminer si la méthode retournera toujours une valeur.
	
	Ici : il suffit de supprimer "if (value>=30)"

```
public void draw(Graphics g) {
   for (int i=0;i<trucs.size();i++ )
      trucs.get(i).draw(g, true);
}
```

```
method draw in class Truc cannot be applied to given types;
 required: Graphics
 found: Graphics,boolean
 reason: actual and formal argument lists differ in length
----
(Alt-Enter shows hints)
```

Incohérence de paramètres
Dans Big, on fait appel à une méthode draw(…) pour un objet de la classe Truc. La méthode draw(…) n'a été défini qu'avec un seul paramètre qui doit être de type Graphics, mais ici on tente d'appeler une méthode qui aurait aussi un 2e paramètre, un booléen. Une telle méthode n'existe pas.

```
if (a>0)
    a+=3;
    b=0;
else
  a=0;
```

'else' without 'if'

On écrit *else* tandis que la structure alternative *if* est déjà fermée (après le premier point-virgule, comme on a oublié de mettre les accolades).

```
public void draw(Graphics g){
    int a = Math.random();
    g.setColor(color);
    g.fillOval(x -radius, y-radius, radius*2, radius*2);
    g.setColor(Color.BLACK);
    g.drawOval(x -radius, y-radius, radius*2, radius*2);
}
```

```
incompatible types: possible lossy conversion from double to int
----
(Alt-Enter shows hints)
```

Erreur de **conversion implicite d'un nombre réel** vers un entier. Dans l'exemple, *Math.random()* est toujours réel et la valeur ne peut être affectée à une variable de type int.
Comme x est un *double*, x-radius est aussi un double et ne peut pas être 1er paramètre de fillOval.

```
10  public class Big {
    private ArrayList<Truc> trucs= new ArrayList<Truc>();
12
13      public void draw(Graphics g) {
14         for (int i=0;i<trucs.size();i++ )
15           trucs.get(i).draw(g);
16         }
17      }
        public Object[] toArray() {
19          return trucs.toArray();
        }
        public void clear() {
22          trucs.clear();
        }
        public boolean add(Truc pA) {
25          return trucs.add(pA);
        }
27  }
```

Lorsqu'il y a **une accolade fermante de trop**, beaucoup d'instructions dans la suite seront signalées comme des erreurs, car considérées en-dehors de la classe.
Dans cet exemple, l'accolade ouvrante a été oubliée à la ligne 14. Ainsi, l'accolade à la ligne 17 clôture la classe.

B] Les erreurs d'exécution (appelés exceptions).

Ce sont des erreurs qui ne sont pas forcément prévisible mais qui font que le programme ne peut pas exécuter le code prévu. Dans NetBeans s'affichent alors en rouge des lignes qui indiquent la nature de l'erreur ainsi que la suite d'appels de méthodes qui y ont conduit.

a) java.lang.IndexOutOfBoundsException

On a essayé d'accéder dans une ArrayList à un élément qui n'existe pas, l'indice se trouvant hors des limites (=bounds). En cas d'erreur, le programme indique l'indice qui a provoqué l'erreur et le nombre d'éléments dans la liste.

Exemple :

```
13      public void draw(Graphics g) {
14          for (int i=0;i<=trucs.size();i++ )
15              trucs.get(i).draw(g);
16      }
```

```
Exception in thread "AWT-EventQueue-0" java.lang.IndexOutOfBoundsException: Index 0 out of bounds for length 0
        at java.base/jdk.internal.util.Preconditions.outOfBounds(Preconditions.java:64)
        at java.base/jdk.internal.util.Preconditions.outOfBoundsCheckIndex(Preconditions.java:70)
        at java.base/jdk.internal.util.Preconditions.checkIndex(Preconditions.java:248)
        at java.base/java.util.Objects.checkIndex(Objects.java:372)
        at java.base/java.util.ArrayList.get(ArrayList.java:459)
        at Big.draw(Big.java:15)
        at DrawPanel.paintComponent(DrawPanel.java:28)
        at java.desktop/javax.swing.JComponent.paint(JComponent.java:1074)
        at java.desktop/javax.swing.JComponent.paintChildren(JComponent.java:907)
        at java.desktop/javax.swing.JComponent.paint(JComponent.java:1083)
        at java.desktop/javax.swing.JComponent.paintChildren(JComponent.java:907)
```

Le programme nous indique dans la première ligne qu'il s'agit d'une erreur IndexOutOfBound et qu'il y avait 0 éléments dans la liste (elle était donc vide) et qu'on a essayé d'atteindre l'élément à l'indice 0 (donc le premier élément, mais qui n'existe pas, comme la liste est vide).

Il faut ensuite trouver l'endroit de notre programme qui a provoqué l'erreur. Dans les lignes affichées il faut exclure les lignes qui commencent avec « java », ce sont des lignes qui concernent des méthodes internes de java. Si on exclut ces lignes alors il en reste deux qui nous indiquent que la méthode *paintComponent()* dans le DrawPanel a lancé dans la ligne 28 la méthode *draw()* de la classe Big et que l'erreur a été provoquée dans cette méthode *draw()*, dans la ligne 15.

Si on regarde la ligne 15 [`trucs.get(i).draw(g)`], on voit que l'erreur doit avoir été provoquée par la méthode *get()* et que c'est ***i*** qui a donc la valeur *0* au moment que la liste est vide. Comme la ligne ne doit pas être exécutée pour une liste vide, on trouve facilement la faute dans la ligne 14 au-dessus. Pour corriger l'erreur, il faut écrire `i<truc.size()` au lieu de `i<=truc-size()`.

b) java.lang.NumberFormatException

Typiquement, on a essayé de prendre un texte qui ne contient pas de nombre et de le convertir en un nombre par la méthode Integer.valueOf() ou Double.valueOf(). Après le nom de l'erreur, le programme nous indique, quel texte a provoqué l'erreur. Si le programme indique « empty String », c'est que le texte était vide.

Exemple :

```
135     private void calcButtonActionPerformed(java.awt.event.ActionEvent evt) {
136         Taxi taxi = new Taxi(
137             Double.valueOf(prixKmTextField.getText()),
138             Double.valueOf(prixBaseTextField.getText()),
139             Double.valueOf(suppPPassTextField.getText()) );
```

```
Exception in thread "AWT-EventQueue-0" java.lang.NumberFormatException: empty String
        at java.base/jdk.internal.math.FloatingDecimal.readJavaFormatString(FloatingDecimal.java:1842)
        at java.base/jdk.internal.math.FloatingDecimal.parseDouble(FloatingDecimal.java:110)
        at java.base/java.lang.Double.parseDouble(Double.java:543)
        at java.base/java.lang.Double.valueOf(Double.java:506)
        at MainFrame.calcButtonActionPerformed(MainFrame.java:137)
        at MainFrame.access$000(MainFrame.java:6)
        at MainFrame$1.actionPerformed(MainFrame.java:59)
        at java.desktop/javax.swing.AbstractButton.fireActionPerformed(AbstractButton.java:1967)
        at java.desktop/javax.swing.AbstractButton$Handler.actionPerformed(AbstractButton.java:2308)
        at java.desktop/javax.swing.DefaultButtonModel.fireActionPerformed(DefaultButtonModel.java:405)
        at java.desktop/javax.swing.DefaultButtonModel.setPressed(DefaultButtonModel.java:262)
```

En excluant toutes les lignes « système » commençant avec « java », mais en excluant aussi les autres lignes que nous n'avons pas écrites nous-mêmes, nous voyant que l'erreur est provoquée dans la ligne 137 de la classe MainFrame, dans la méthode *calcButtonActionPerformed()*.

Donc, nous avions cliqué sur le bouton *calcButton*, mais nous avons oublié de vérifier que tous les champs d'édition nécessaires pour notre calcul soient préalablement bien remplis : le champ *prixKmTextField* était vide.

c) java.lang.NullPointerException

Il s'agit ici d'une erreur fréquente. Elle intervient quand on essaye d'accéder à un attribut ou une méthode d'un objet et que ce dernier est *null*. Pour qu'un objet ne soit pas *null*, il faut le créer avec **new** ou bien lui affecter comme valeur un autre objet non *null*.

C'est la raison pourquoi on écrit habituellement dans la méthode paintComponent du DrawPanel : `if(big!=null) big.draw(g);`

En effet, si *big* est *null*, alors on ne peut pas appeler la méthode draw() car elle n'existerait pas.

Exemple dans le programme LinesDrawGUI qui fonctionne bien quand on n'utilise que le bouton gauche pour dessiner, mais qui donne une NullPointerException quand on commence le programme en tirant avec la touche droite de la souris.

```
97  private void drawPanelMousePressed(java.awt.event.MouseEvent evt) {
98      if (MouseEvent.BUTTON1==evt.getButton()) {
99          newLine = new Line(evt.getPoint(), evt.getPoint(), drawColor);
100         lines.add(newLine);
101         updateView();
102     }
103 }
104
105 private void drawPanelMouseDragged(java.awt.event.MouseEvent evt) {
106     newLine.setTo(evt.getPoint());
107     updateView();
108 }
```

```
Exception in thread "AWT-EventQueue-0" java.lang.NullPointerException: Cannot invoke "Line.setTo(java.awt.Point)" because "this.newLine" is null
        at MainFrame.drawPanelMouseDragged(MainFrame.java:106)
        at MainFrame.access$300(MainFrame.java:11)
        at MainFrame$2.mouseDragged(MainFrame.java:61)
        at java.desktop/java.awt.Component.processMouseMotionEvent(Component.java:6674)
        at java.desktop/javax.swing.JComponent.processMouseMotionEvent(JComponent.java:3407)
        at java.desktop/java.awt.Component.processEvent(Component.java:6395)
```

Selon cet affichage, l'erreur se trouve dans la ligne 106 de la méthode drawPanelMouseDragged :

```
newLine.setTo(evt.getPoint());
```

Dans cette ligne il y a deux objets qui sont susceptibles d'avoir provoqué l'erreur : *newLine* et *evt*. Comme *evt* a été créé par le système, c'est certainement *newLine* qui doit être *null*.

Effectivement, comme on a tiré avec le bouton droit, l'objet *newLine* n'a pas été créé dans la ligne 99 de la méthode *drawPanelMousePressed*.

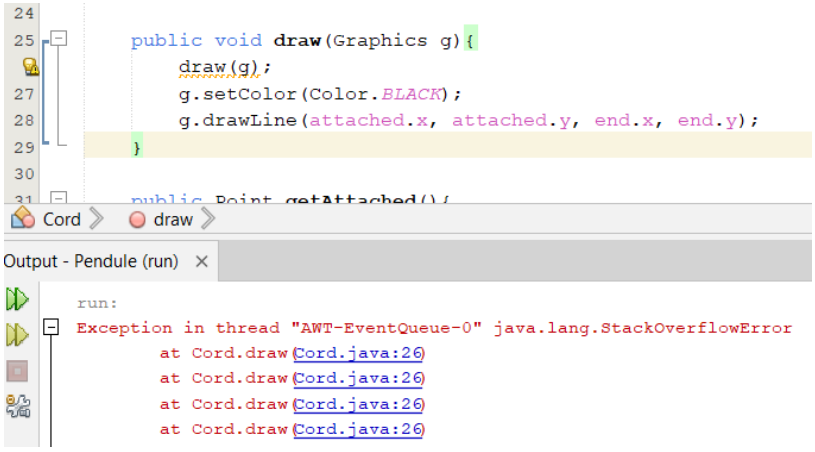
Dans cet exemple, on peut corriger l'erreur en mémorisant par un attribut la touche de la souris dans *drawPanelMousePressed*, puis de vérifier dans *drawPanelMouseDragged* si la touche est bien la touche gauche. Ou bien il faut vérifier dans *drawPanelMouseDragged* si *newLine* n'est pas vide avant d'appeler sa méthode *setTo()* et faire en sorte que *newLine* devienne vide dans *drawPanelMousePressed* si ce n'est pas la touche gauche.

d) java.lang.ArithmeticException: / by zero

Il y a eu une division (arithmétique) par zéro.

e) java.lang.StackOverflowError

Il y a un dépassement de mémoire, causé souvent par le fait qu'une boucle infinie use de plus en plus de mémoire, par exemple si une méthode s'appelle elle-même.



C] Les erreurs sémantiques

Ce sont des erreurs que le compilateur ne peut pas détecter. Elle a lieu là où le programme ne fait pas ce qui est attendu de lui. Il y a faute de programmation ou d'algorithme. Dans ce cas il faut réanalyser l'énoncé. Un outil important pour trouver ces erreurs peut être le debugger.

Le debugger

Le debugger est une fonctionnalité dans NetBeans qui sert à analyser de manière détaillée le déroulement d'un programme avec possibilité d'avancer instruction par instruction et d'afficher les états actuels des attributs et variables. Il peut être très utile dans la recherche d'une erreur.

Placer un **break point** : clic sur numéro de ligne

```
34
35      public void untroc() {
            for (int i=alTrucs.size()-1; i>=0; i--)
37              if (alTrucs.get(i) instanceof Troc)
38                  alTrucs.remove(i);
39      }
40
```

Lancer le programme en mode *debug* : par le bouton à droite du bouton "run" ou Ctrl-F5

Quand le Programme rencontre un break point, il se met en pause, la ligne est de couleur verte :

```
35      public void untroc() {
            for (int i=alTrucs.size()-1; i>=0; i--)
37              if (alTrucs.get(i) instanceof Troc)
38                  alTrucs.remove(i);
39      }
```

On a alors plusieurs possibilités pour agir : 

Finish (Shift-F5) : Stop stopper le programme définitivement (en laissant les break points)

Pause : mettre le programme en pause (après l'avoir fait continuer)

Continue (F5) : faire continuer le programme

Step Over (F8) : continuer jusqu'à la ligne suivante

Step Over expression (Shift-F8) : continuer jusqu'à l'expression suivante (si plusieurs expressions dans la même ligne)

Step into (F7) : entrer dans la méthode appelée (ex. : dans ligne 38, enter dans remove())

Step out (Ctrl-F7) : continuer jusqu'à sortir de la méthode actuelle

Run to Cursor (F4) : continuer jusqu'au curseur (qu'il faut préalablement placer dans une ligne)