|  |  |
| --- | --- |
| **Formatif 2 – Solution**  Les erreurs dans le code et débogage | |
| **Nom, prénom DA** | * Nom, Prénom (1234567) |
| **Groupe** |  |
| **Cours** | **201 - Introduction à la programmation** |
| **Enseignant** | Robert Aubé |
| **Date** | lundi, 30 août 2021 |
| **Temps alloué** | Voir sur Léa |
| **Évaluation** | Ce travail se fait seul |

Table des matières

[Prérequis 1](#_Toc50648070)

[À lire et à faire 2](#_Toc50648071)

[Les 3 types d’erreurs en Java 2](#_Toc50648072)

[Erreur de compilation 2](#_Toc50648073)

[Erreur d’exécution 4](#_Toc50648074)

[Erreurs de logique 5](#_Toc50648075)

[Exercices théoriques sur le débogueur 5](#_Toc50648076)

[Exercice pratique sur le débogueur 7](#_Toc50648077)

[Consignes de remise du formatif 10](#_Toc50648078)

# Prérequis

Avant de vous lancer dans ce formatif, lire la section **Débogage avec IntelliJ** des notes de code :

201A21 - Environnement de programmation IntelliJ IDEA - V1.docx

# À lire et à faire

Lisez et complétez directement dans le logiciel Word ce document.

|  |  |
| --- | --- |
| 34 |  |
|  | Les 3 types d’erreurs en Java[[1]](#footnote-1) En programmation, on distingue 3 types d’erreurs lorsque l’on programme en **Java** :   |  |  |  | | --- | --- | --- | | Type d’erreurs | Explication | Exemple | | Erreur de compilation | Il y a une erreur dans la syntaxe du code et le programme ne compile pas et donc ne démarre même pas. | Il manque un point-virgule (;) à la fin d’une ligne | | Erreur d’exécution | Le programme compile, mais le programme arrête à l’exécution en raison d’une erreur. | Une division par 0. | | Erreur de logique | Le programme compile, ne fait pas d’erreur, mais ne donne pas le résultat escompté. | Une calculatrice qui afficherait toujours 3 comme réponse peu importe l’opération. |   Chacun de ces trois types d'erreurs est détaillé dans les prochaines sections. Erreur de compilation Il y a aura des **erreurs de compilation** s’il y a des erreurs de syntaxe dans le code source qui empêche le compilateur de pouvoir produire le code exécutable. **IntelliJ** indique ce genre d’erreur directement lorsque vous écrivez le code.  **Exemple**    Dans l’image au-dessus, on voit bien en  qu’il y a une erreur ().   1. Aussi, sur cette même image, il y a une multitude d’endroits où on indique qu’il y a une erreur. Combien en **dénombrez-vous** qui sont indiqué avec un **soulignement ondulé rouge**? Avec un **trait**? |
| 2 | Un soulignement : ***7***  Un trait : ***1*** |
|  | 1. Si dans un programme similaire, il manque un point-virgule (;) à la fin d’une ligne et que l’on place la souris directement sur le **souligné rouge** dans le code où il manque le point-virgule. Qu’indique IntelliJ ? |
| 1 |  |
|  | Si malgré l’erreur, on essaie d’exécuter le programme, **IntelliJ** ouvre une fenêtre **Build Output** qui indiquera encore l’erreur :    Dans l’image présentée, l’erreur est à nouveau indiquée dans la section **Build Output**. Il est donc **important que le programmeur lise cette section qui lui donne des indices sur les erreurs à corriger** dans son programme.   1. En  dans l’image, on indique la valeur **4**. Que signifie cette valeur? |
| 1 | ***Le numéro de ligne*** |
|  | 1. Dans la section de droite du **Build Output**, on est encore plus précis : on indique **4:47**. Qu’indique la valeur **47**? |
| 1 | ***Le numéro de colonne ou le numéro de la position du caractère dans la ligne*** |
|  | Erreur d’exécution On distingue une **erreur d’exécution** lorsque le programme compile et le compilateur ne perçoit aucune erreur. Cependant lors de l’exécution, le programme se termine de manière anormale et brutale. Souvent causé par une opération illégale (Exemple : une Division par zéro).    Dans l’image ci-haut, on voit que le programme s’est exécuté et a « **planté** » à la **ligne 7** lors d’une division par 0. Ainsi, le programme a écrit Bonjour, mais les **lignes 8**, **9** et **10** n’ont pas été exécutées. En , on voit l’explication de l’erreur : *une division par 0*. En , on voit que le programme s’est arrêté à la ligne 7 du fichier BonjourLeMonde.Java.   1. En , on indique que l’appel au constructeur de BonjourLeMonde.Java s’est fait dans quel fichier et à quelle ligne? |
| 2 | Fichier : ***PremiereClasse.java***  Ligne : ***3*** |
|  | 1. En , on indique que le processus (l’exécution du programme) s’est terminé avec un code de sortie qui vaut 1. Quel serait le code de sortie si le programme se termine normalement (donc sans erreur)? |
| 1 | ***0*** |
|  | Donc, même si votre programme ne fonctionne pas correctement, il pourrait s’arrêter subitement lors de son exécution. **Le programmeur doit alors** **observer la sortie de son programme pour trouver où s’est produite l’erreur et quelle est la nature de l’erreur**. |
|  | Erreurs de logique Les **erreurs de logique** sont les erreurs les plus difficiles à détecter et à corriger. Le programme compile et s’exécute sans problème, mais le résultat obtenu n’est pas celui escompté! (Exemple : un calcul ne donnerait pas le bon résultat)  La détection de ces erreurs est difficile : dans les débuts, on vous demande **d’exécuter le code et de bien s’assurer qu’il réponde au problème présenté**. On appelle ça **faire des tests**.  Dans les sessions suivantes, on élaborera les techniques de tests qui permettront de mieux détecter ces problèmes.  Une fois que l’on détecte que le programme n’a pas le comportement souhaité, il faut trouver ce qui cause l’erreur. Dans cette situation, le débogueur est un outil de choix : on peut suivre pas à pas pour voir le comportement du programme pour trouver ce qui ne tourne pas rond.  Bien qu’elles soient difficiles à détecter et à corriger, elles sont très satisfaisantes à résoudre par le programmeur. |
|  | Exercices théoriques sur le débogueur |
|  | 1. Répondez par **vrai** ou **faux** : « Si on ne place pas de point d’arrêt (break point)  et que l’on exécute le débogueur, le programme s’exécute au complet sans jamais s’arrêter. » |
| 1 | ***Vrai*** |
|  | 1. Répondez par **vrai** ou **faux** : « Si on place des points d’arrêt (break point) et que l’on exécute le débogueur, **si le programme ne passe pas par un point d’arrêt**, il s’exécutera sans jamais s’arrêter. » |
| 1 | ***Vrai*** |
|  | 1. Répondez par **vrai** ou **faux** : « Pour voir par exemple le contenu d’une variable dans mon programme rendu à une certaine ligne, je dois au moins placer un point d’arrêt pour que le programme s’arrête. » |
| 1 | ***Vrai*** |
|  | 1. Indiquez brièvement comment placer un point d’arrêt (break point) par exemple à la ligne 8 d’un programme. |
| 2 | ***Il faut cliquer en marge du code à la ligne 8*** |
|  | Soit les actions suivantes d’exécution d’un programme avec **IntelliJ** :   1. Le programme s’exécute en mode debug 2. Le programme s’exécute jusqu’à la fin sans tenir compte des points d’arrêt du débogueur. Bref, il s’exécute normalement. 3. En indiquant l’une des 2 lettres précédentes, associez la conséquence de l’action du programmeur dans **IntelliJ** (Ici, comme réponse, la même lettre sera utilisée plusieurs fois) : |
| 5 | |  |  | | --- | --- | | Que se produit-il si… | Lettre de l’action associée | | On appuie sur l’icônesous la barre de menu? | ***B*** | | On appuie surqui se trouve dans la marge vis-à-vis l’une des lignes d’un programme? | ***B*** | | On appuie sur l’icônesous la barre de menu? | ***A*** | | On fait la combinaison des touches **ALT-Maj+F9**? | ***B*** | | On fait la combinaison des touches **ALT-Maj+F10**? | ***A*** | |
| 8 | Soit les actions suivantes liées à l’exécution en mode débogueur :   1. Désactiver les points d'arrêt pour le reste de l'exécution 2. Exécuter l'instruction suivante, mais sans faire un appel à une méthode si c'est le cas 3. Exécuter l'instruction suivante, même si celle-ci est à l'intérieur d'une méthode 4. Poursuivre l'exécution du programme jusqu'à la position du curseur dans le programme 5. Poursuivre l'exécution jusqu'à la fin du programme ou un prochain point d'arrêt 6. Poursuivre l'exécution jusqu'à la sortie de la méthode 7. Terminer immédiatement l'exécution du débogueur 8. Terminer le programme puis recommencer l’exécution en mode débogueur 9. Associez l’action à la bonne icône :  |  |  | | --- | --- | | Icône | Lettre de l’action associée | |  | ***E*** | |  | ***G*** | |  | ***C*** | |  | ***B*** | |  | ***F*** | |  | ***H*** | |  | ***D*** | |  | ***A*** | |
|  | Exercice pratique sur le débogueur Dans un projet quelconque, créer une classe java dont le nom est FormatifDebogueur et coller exactement le code suivant :   |  | | --- | | public class FormatifDebogueur {  public FormatifDebogueur() {  permutation();  }   public void permutation() {  *// Déclaration des variables* int nb1 = 10;  int nb2 = 15;  int temp = 0;   *// Afficher les 2 nombres avant la permutation* System.*out*.println("Avant la permutation: ");  System.*out*.println("nb1 = " + nb1);  System.*out*.println("nb2 = " + nb2);   *// Permuter les 2 nombres* temp = nb1;  nb1 = nb2;  nb2 = temp;   *// Afficher les 2 nombres après la permutation* System.*out*.println("Après la permutation: ");  System.*out*.println("nb1 = " + nb1);  System.*out*.println("nb2 = " + nb2);  }   public static void main(String[] args) {  new FormatifDebogueur();  } } | |
|  | Une fois que vous avez copié le code, vous devriez avoir **exactement** ceci :    **ATTENTION** : Assurez-vous que vos numéros de ligne concordent bien avec ceux de l’image précédente sinon vous n’arriverez pas aux bons résultats.   1. Exécutez en mode débogueur puis donnez la valeur de chaque variable à chacune des lignes indiquées pour le fichier FormatifDebogueur.java. Si la valeur d’une variable n’est pas définie, indiquez simplement un trait. Les lignes 29 et 10 sont données à titre d’exemple et sont exactes. |
| 8 | |  |  |  |  | | --- | --- | --- | --- | | No ligne | nb1 | nb2 | Temp | | 29 | *-* | *-* | *-* | | 3 | *-* | *-* | *-* | | 8 | *-* | *-* | *-* | | 9 | *10* | *-* | *-* | | 10 | *10* | *15* | *-* | | 13 | *10* | *15* | *0* | | 18 | *10* | *15* | *0* | | 19 | *10* | *15* | *10* | | 20 | *15* | *15* | *10* | | 26 | *15* | *10* | *10* | |
|  | 1. Répondez par **vrai** ou **faux** : « Lorsque le débogueur s’arrête sur une ligne. L’état présenté est celui sans considérer cette ligne. » |
| 1 | ***vrai*** |
|  | 1. Répondez par **vrai** ou **faux** : « On peut placer un break point sur un commentaire. » |
| 1 | ***faux*** |
|  | 1. Répondez par **vrai** ou **faux** : « On peut placer un break point sur une ligne blanche. » |
| 1 | ***faux*** |
|  | 1. Une autre façon de procéder au débogage d’une application est de faire une trace manuelle… En considérant que chacune des lignes suivantes se suivent dans une même méthode. Indiquez le contenu des variables. Indiquez **un trait** si la variable n’est pas encore définie et un ? si la variable existe en mémoire mais qu’elle n’a pas encore une valeur attribuée : |
| 1 | |  |  |  |  | | --- | --- | --- | --- | | Nom->  Type-> | x  int | y  int | z  int | | int x; | ***?*** | ***-*** | ***-*** | | int y; | ***?*** | ***?*** | ***-*** | | int z; | ***?*** | ***?*** | ***?*** | | x = 0; | ***0*** | ***?*** | ***?*** | | y = 10; | ***0*** | ***10*** | ***?*** | | z = y; | ***0*** | ***10*** | ***10*** | | y = y + 1; | ***0*** | ***11*** | ***10*** | | z = z \* y; | ***0*** | ***11*** | ***110*** | | x = x – 1; | ***-1*** | ***11*** | ***110*** | | x = x \* -2 + 3; | ***5*** | ***11*** | ***110*** | | x = 3 + x \* 2; | ***13*** | ***11*** | ***110*** | |
|  | Consignes de remise du formatif  1. Imprimez :  * Ce formulaire complété dans Word * Déposez ce fichier **Word** sur Léa.   Et brochez le tout dans l’ordre.   1. Remettre ce document imprimé à votre enseignant |

1. Pour plus de détails sur les types d’erreurs, suivez le lien suivant :

   <http://www.ukonline.be/cours/java/apprendre-java/chapitre1-4> [↑](#footnote-ref-1)