Module 1 –Informatique – Algorithmique et Programmation Objet
Travaux Pratiques (2), Licence 1ère Année
Types – Conversion – Précision - Débogage

Les exercices suivants ont pour but de vous familiariser à la manipulation des types de base de Java et à leur conversion. Il sera aussi souligné la limite des nombres en machines et leur imprécision. Une introduction succincte au débogage est également présentée.

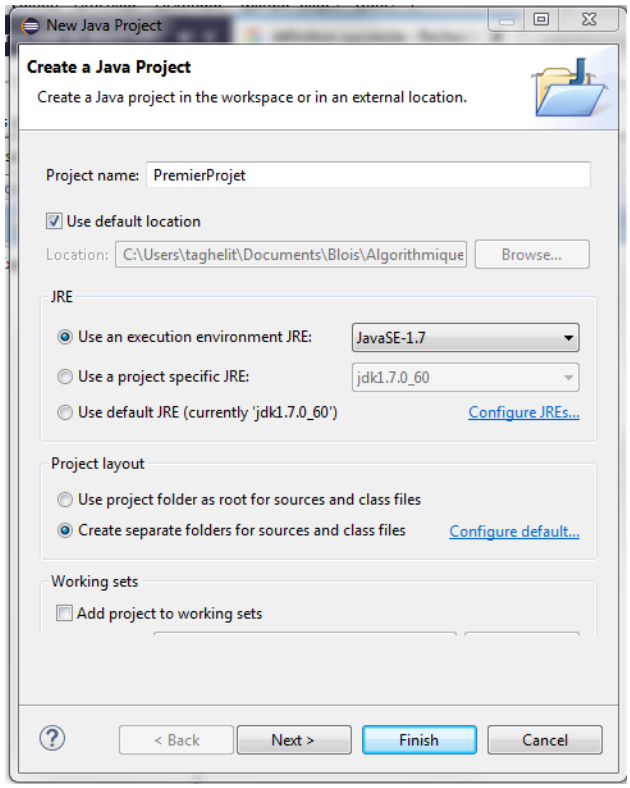
Préambule Prise en main d'Eclipse

Étape 1 Lancer Eclipse

Sur votre poste, lancez Eclipse à partir de Windows ou de la machine virtuelle Linux. Des consignes vous seront données en séance.

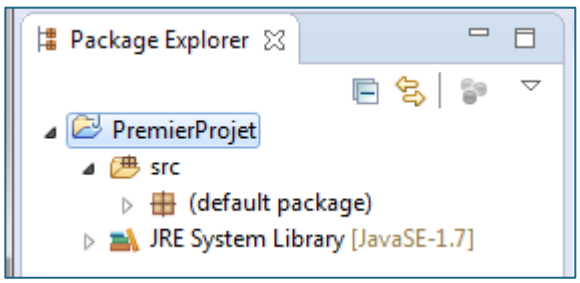
Étape 2 Créer un nouveau projet

Créez un nouveau projet (File → New → Java Project) et dans la fenêtre qui s'ouvrira donnez le nom « PremierProjet » à votre projet. Puis cliquez sur le bouton « Finish ».



Étape 3 Sélectionner votre projet

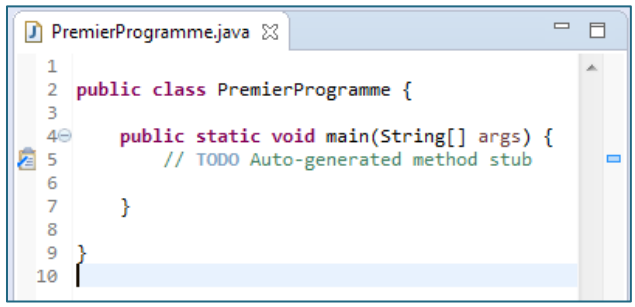
Dans la fenêtre de gauche (Package Explorer), cliquez sur le nom de votre projet.



Étape 4 Créer une classe

Créez une classe (File → New → Class) ou en cliquant sur l'icône  dans la barre du haut.
Dans la fenêtre qui s'ouvrira donner le nom « PremierProgramme » à votre classe et cochez la case qui permet d'ajouter automatiquement la méthode principale « public static void main(String[] args) ».

Puis cliquez sur le bouton « Finish ».
Vous devriez obtenir à droite une fenêtre telle celle-ci-contre.

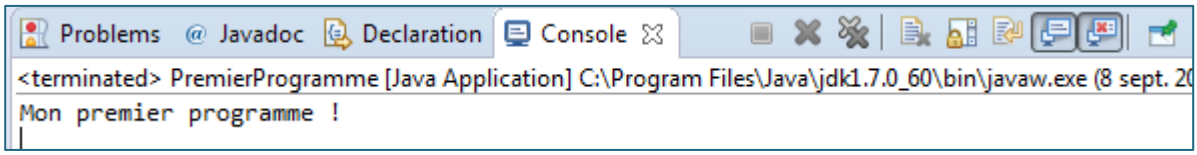


Étape 5 Écrire son premier programme

Dans le corps de la méthode main() ajouter l'instruction pour afficher « mon premier programme » : System.out.println("Mon premier programme !").
Remarque : si vous tapez syso puis « Ctrl + Espace » cela va écrire System.out.println();

Étape 6 Exécuter son programme

Exécutez votre programme (en cliquant sur l'icône  sur la barre du haut.
Vous devez voir dans la fenêtre du bas apparaître un nouvel onglet « console ».



Bravo, vous avez exécuté votre premier programme en Java sous Eclipse !

Exercice 1 Opérations sur les types

Soient les déclarations suivantes :

```
double d = 4.0;
int i1 = 10;
int i2 = 7;
```

Soient les expressions suivantes :

1. 5 / 7
2. 5. / 7
3. d + (i1 / i2)
4. (d + i1) / i2
5. d + (i1 % i2)
6. (d + i1) % i2
7. i1 % 1.6

Quel doit être le type et la valeur du résultat de l'évaluation de chacune des expressions précédentes ?
Vérifiez vos prédictions en les évaluant dans votre programme.
Note : il suffit de même chacune de ces expressions comme paramètre de l'instruction d'affichage : System.out.println(expression) ; .

Soient les déclarations suivantes :

```
int i3 = 5;
int i4 = 2;
double d1 = 5.0;
double d2 = 2.0;
```

Quel doit être le type et la valeur du résultat de l'évaluation de chacune des expressions suivantes ?

```
8. i3 / i4
9. d1 / d2
10. i3 / d2
11. d1 / i4
12. i3 % i4
13. d1 % d2
```

Vérifiez vos prédictions en les évaluant dans votre programme.

Exercice 2 Comprendre le mécanisme du cast (conversion)

Soient les déclarations suivantes :

```
int sept = 7;
int deux = 2;
int i1, i2;
float f1, f2;
```

Soit la séquence d'instructions suivante :

```
1. i1 = sept / deux;
2. i2 = deux / sept;
3. f1 = (float) (sept / deux);
4. f2 = (float) (sept / deux) + 0.5f;
5. i1 = (int) f1;
6. i2 = (int) f2;
7. f1 = (float) sept / (float) deux;
8. f2 = (float) sept / (float) deux + 0.5f;
9. i1 = (int) f1;
10. i2 = (int) f2;
```

Nous supposons que cette séquence est exécutée séquentiellement dans l'ordre indiqué des instructions.
Donnez la valeur de chacune des variables après chacune de leurs affectations.
Vérifiez vos prédictions en les exécutant dans votre programme.

Quelle est la valeur de chacune des expressions suivantes :

```
11. (double) 5 / (double) 2
12. (double) ( 5 / 2)
```

Exercice 3 Priorités des opérateurs

Java définit les priorités pour les opérateurs comme indiqué dans le tableau ci-contre (du plus prioritaire au moins prioritaire).

Les opérateurs les plus prioritaires seront évalués en premier tandis que les moins prioritaires seront donc évalués en dernier.

Les parenthèses ayant une forte priorité, l'ordre d'interprétation des opérateurs peut être modifié par des parenthèses.

Quel doit être le type et la valeur du résultat de chacune des expressions suivantes ?

1. 7 + 2 * 3
2. 1 - 2 - 3
3. 1 + 2 * 3 / 4
4. 3 * 5 - 2 + 1 - 8 / 2 * 3
5. -9 >= 6 && true || 7 * 3 < 2

Vérifiez vos prédictions en les évaluant dans votre programme.

Donnez pour chacune des expressions précédentes leurs équivalents « parenthésés ».

les parenthèses	**( )**
les opérateurs d'incrémentation	**++ --**
les opérateurs de multiplication, division et modulo	*** / %**
les opérateurs d'addition et soustraction	**+ -**
les opérateurs de décalage	**<< >>**
les opérateurs de comparaison	**< > <= >=**
les opérateurs d'égalité	**== !=**
l'opérateur OU exclusif	**^**
l'opérateur ET	**&**
l'opérateur OU	**|**
l'opérateur ET logique	**&&**
l'opérateur OU logique	**||**
les opérateurs d'assignement	**= += -= *= /= %=**

Exercice 4 Limite et (Im)Précision des nombres

Écrivez dans votre programme les trois instructions suivantes :

```
byte b = 112;
b += 20;
System.out.println(b);
```

Quelle valeur va être affichée pourquoi ?

Écrivez dans votre programme les instructions suivantes :

```
float x1 = (float) 10.2, y1 = (float) 3.101, resu1;
resu1 = x1 / y1;
System.out.println(resu1);
System.out.println( (double)x1 / (double)y1 );
double x2 = 10.2, y2 = 3.101, resu2;
resu2 = x2 / y2;
System.out.println(resu2);
```

Quelles valeurs vont être affichées, pourquoi les résultats sont différents ?

Écrivez dans votre programme les instructions suivantes :

```
float     a = 3.1f, b = 2.3f, c = 1.5f;
float     x, y;
x = ( a * b ) * c;
y = a * ( b * c );
System.out.println("( " + a + " * " + b + " ) * " + c + " = " + x);
System.out.println(a + " * ( " + b + " * " + c + " ) = " + y);
```

Quelles valeurs vont être affichées, pourquoi les résultats sont différents ?

Exercice 5 Débogage

On considère la méthode **sequence()** suivante :

```
public static void sequence() {
      int a;
      a = 2;
      double b = 3.5;
      int c = 8;
      String d = "toto";
      a = a + c;
      b = 3 * b + a;
      c = (int)b - c;
      d = d + a;
      a = 3;
      b = 4;
      c = 5;
      boolean b1 = (a <= b) && (b < c);
      boolean b2 = (a > b) || (b >= c);
      boolean b3 = !b1;
      boolean b4 = b2 == b3;
}
```

Le tableau ci-dessous contient autant de colonnes qu'il y a de variables dans le programme précédent et autant de ligne qu'il y a d'instructions dans le programme précédent. Complétez le tableau de façon qu'il contienne pour chaque variable l'évolution de sa valeur au cours des instructions. Lorsqu'une variable n'existe pas encore, vous lui affecterez la valeur x et sinon, si la variable existe et qu'aucune affectation n'a été réalisée, vous utiliserez sa valeur par défaut. En cas d'erreur, signalez-la et corrigez-la de façon à pouvoir continuer l'exécution du programme.

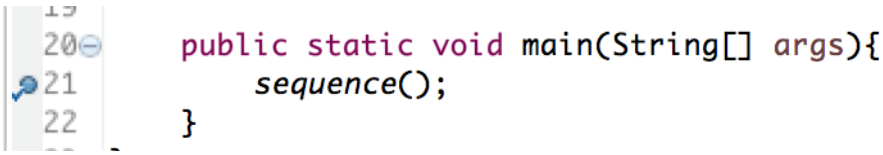
Note : la première ligne, correspondant à la première instruction, est déjà remplie.

Instructions	a	b	c	d	b1	b2	b3	b4
`int a;`	0	x	x	x	x	x	x	x

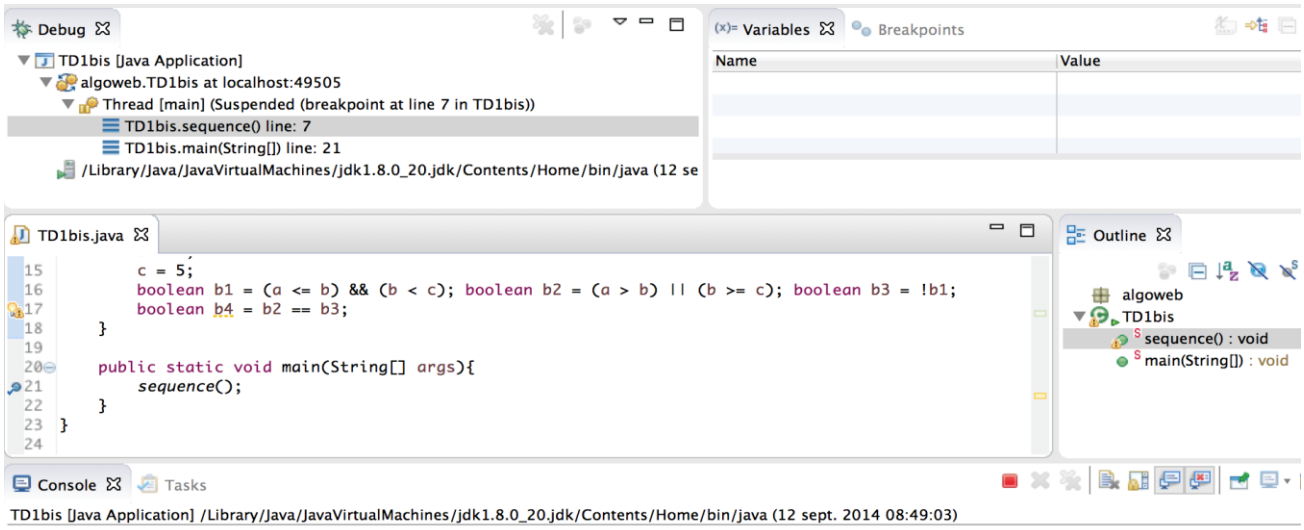
Nous allons maintenant rejouer la séquence précédente sur ordinateur mais au lieu de l'exécuter, nous allons la déboguer. Pour cela, il faut préalablement écrire une méthode **`main()`** qui permet d'exécuter le programme comme suit :

```
public static void main(String[] args){
      sequence();
}
```

Il faut ensuite ajouter un point d'arrêt (**breakpoint**) en double cliquant dans la marge de la ligne à laquelle on souhaite commencer à déboguer le programme comme illustré dans la figure suivante :



Ensuite, il faut cliquer sur l'icône  (**debug**) qui permet de changer l'affichage à l'écran pour passer en mode débogage. La nouvelle fenêtre se présente comme suit :



En haut à gauche on trouve dans l'onglet **Debug**, le détail de la pile des appels des méthodes Java. En haut à droite, l'onglet **Variables** liste les variables utilisées dans le programme ainsi que leur valeur. Cette information va nous permettre de vérifier que l'exécution de notre programme est conforme à ce que nous avions prévu initialement.

Ensuite, un onglet au centre et à gauche permet de voir avec une ligne en surbrillance où on se trouve dans l'exécution du programme et à droite l'onglet **Outline** présente les propriétés et les méthodes de la classe qui contient le code qui est parcouru actuellement.

Enfin, le dernier onglet **Console** en bas de l'écran permet de suivre les entrées / sorties en mode texte du programme au fur et à mesure de son débogage.

En conclusion, pour déboguer votre programme il faut :

1. mettre un point d'arrêt à l'endroit que vous souhaitez
2. passer en mode ***debug***,
3. utiliser le menu **Run** de Eclipse ou les raccourcis pour :
    - rentrer dans le prochain appel de fonction (pas à pas détaillé) (**Step Into**) : touche F5
    - aller à la ligne suivante sans rentrer dans le détail des appels de fonctions de la ligne courante (**Step Over**) : touche F6
    - sortir de la fonction courante (car il n'y a plus rien à vérifier) : touche F7
    - exécuter jusqu'au prochain point d'arrêt (ou la fin) : touche F8
    - en cas de boucle infinie ou de problème : touche F2