1. أساسيات لغة الجافا

1.1 المتغيرات Variables

ستحتاج في مرحلةٍ ما إلى التعامل مع البيانات. وعند رغبتك في تخزين تلك البيانات، فإنك ستحتاج إلى شيء يقوم بتخزينها لك، وهذا هو عمل المتغيرات. يمكن تخيل المتغيرات والنظر إليها على أنها الأوعية أو الحاويات التي تحتوي وتخُزن كل ما يوضع فيها.
إذاً، يمكن النظر للمتغير على أنه أسلوب بسيط لتخزين البيانات واسترجاعها بشكل مؤقت أثناء عمل البرنامج.

تعريف المتغيرات باستخدام int

لتوضيح الفكرة لاحظ معي المثال التالي:

```
int age;
```

في المثال أعلاه قمنا بإنشاء و تعريف متغير باسم age، وذلك لحفظ قيمة العُمر بداخله. الآن دعنا نقوم بإسناد قيمة العُمر 26 له:

```
age=26;
```

لاحظ أننا قمنا باستخدام المتغيرage في هذه المرة بدون استخدام int ، وذلك لأنها تستخدم مرة واحدة فقط وهي أثناء تعريف المتغير، وبعد ذلك، سنتعامل مع المتغير بشكل مباشر من خلال اسمه فقط، وفي هذه الحالة هو age.
يمكنك أيضاً اختصار الخطوتين السابقتين في خطوة واحدة كالتالي:

```
int age=26;
```

يمكن للمبرمج استخدام age في أماكن مختلفة من البرنامج، وسيتم استبدالها بالقيمة 26. بالإضافة إلى ذلك، يمكن للمبرمج أن يقوم بتغيير قيمة المتغير أثناء البرنامج. فمثلاً، بعد تعريفنا للمتغير السابق age، يمكننا تغيير قيمته إن أردنا، ولتوضيح الفكرة لاحظ معي المثال التالي:

```
 age=30;
```

1.2 التعليقات Comments

في حالات معينة أثناء كتابة الكود، قد يحتاج المبرمج إلى وضع بعض الملاحظات أو التعليقات. فمثلاً، قد يحتاج إلى وضع ملاحظة لتذكيره بتعديل كود معين، فيقوم المبرمج حينها بكتابة بعض الملاحظات بجانب ذلك الكود للعودة إليه فيما بعد. وفي حالات أخرى، قد يعمل على الملف البرمجي أو المشروع البرمجي أكثر من شخص، وقد يحتاج أحد المبرمجين إلى أن يضع بعض الملاحظات لأعضاء الفريق، وهكذا.
توفر التعليقات في java طريقة تُساعد المبرمج على كتابة ما يود من ملاحظات في البرنامج، وبالنسبة للغة java فإنها ستتجاهل تلك التعليقات، ولن تنظر لها على أنها تعليمات ستقوم بتنفيذها.
سنتحدث في هذا الجزء عن أنواع التعليقات في java، و هي:

- Single Line Comment تعليق السطر الواحد
- Multi-Line Comment تعليق متعدد الأسطر

تعليق السطر الواحد Single Line Comment

عند رغبتنا في وضع تعليق في سطر واحد أو single-line comment، والذي سينتهي بنهاية السطر، سنستخدم //كعلامة لبداية التعليق. يوضح السطر التالي هذه الفكرة.

```
// This is a comment.
```

ليس بالضرورة أن يبدأ التعليق من بداية السطر، فقد يكون التعليق هو جزء من سطر برمجي. لتوضيح الفكرة، لاحظ معي المثال التالي:

```
int age = 25; // This is my age.
```

الأسطر متعدد تعليق Multi-line Comments

استخدام يمكننا الحالة، هذه في .سطر من أكثر على يمتد طويل، تعليق كتابة إلى نحتاج قد الحالات، بعض في */ بين الملاحظات كتابة طريق عن بذلك ونقوم .Multi-Line Comment الأسطر متعدد التعليق أسلوب .الأمر هذا التالي المثال يوضح .*/

```
/* Write your
   comments here
*/
```

1.3 التسميات Naming

للتسميات في لغة java شروط ومن غير الممكن تسمية المتغيرات او الثوابت إذا خالفة هذه الشروط.

1. لا يمكن تسمية متغير يحتوي على كلمتين ،لتوضيح الفكرة، لاحظ المثال التالي:

```
String my Name = "Khalid"; //Wrong
```

و عوضا عن ذلك نقوم بتسمية المتغيرات التي تحتوي على كلمتين عن طريق استخدام اسلوب كتابة Camel case ، عن طريق كتابة اول كلمة بحرف صغير ثم أول حرف من كل كلمة يكون حرفا كبيرا. لتوضيح الفكرة، لاحظ المثال التالي:

```
String myName = "Khalid"; //Correct
```

2. لا يمكن التسمية بأسماء تحتوي في داخلها على احدى الرموز الخاصة بالعمليات مثل علامة الجمع + و علامة لطرح الى آخره، لتوضيح الفكرة، لاحظ المثال التالي:

```
String +name = "Khalid" //Wrong
```

3. لا يمكن التسمية بأسماء محجوزة في اللغة مثل كلمة final الخاصة بتعريف الثوابت:

```
String final = "Khalid" //Wrong
```

1.4 انواع البيانات Datatype

تدعم لغة Javaعدداً من أنواع البيانات. يوضح الجدول التالي هذه الأنواع.

الوصف	النوع
لتمثيل الأعداد الصحيحة.	int
يستخدم لتمثيل الارقام التي تحتوي على النقطة العشرية	double
ويستخدم لتمثيل أنواع البيانات النصية مثل characters والنصوص strings.	String
أي بيانات من هذا النوع تكون ضمن قيمتين وهما true و false.	boolean
يستخدم لتمثيل الأحرف	char

تعريف متغير من نوع String

يمكننا استخدام ثلاث طرق مختلفة للتعامل مع النصوص على النحو التالي:

```
String message = "Welcome to java";
```

طريقة دمج النصوص باستخدام علامة "+"
يمكننا دمج نصين أو أكثر ليكونا نصا واحدًا، كما يمكننا دمج متغير ونص على النحو التالي:

```
String itemTwo = "java";
String message = "Welcome to " + itemTwo;
```

ومفهوم النصوص Escape Character

تستخدم هذه العمليات داخل النص String و كل واحدة منها تقوم بعملية محددة فعلى سبيل المثال \n يجعل ما بعده على سطر جديد و منها ، يوضح الجدول التالي عمليات Escape Character:

العملية اسم	العملية رمز	وظيفتها
Horizontal Tab	\t	وضعه مكان في المسافات من عدد يضيف
Newline	\n	جديد سطر على بعده ما بجعل يقوم
Single quote	\'	وضعها مكان ' باضافة تقوم
Double quote	\”	وضعها مكان " باضافة تقوم
Backslash	\\	وضعها مكان \ باضافة تقوم

تعريف متغير من نوع Number
يُعرّف المتغير من نوع numbe كتعريف المتغير العادي، ويُسند إليه قيمة رقم. لاحظ معي المثال التالي:

```
int valueType = 2;
```

تعريف متغير من نوع Double
يُعرّف المتغير من نوع Double كتعريف المتغير العادي، ويُسند إليه قيمة رقم عشري. لاحظ معي المثال التالي:

```
double valueType = 2.0;
```

تعريف متغير من نوع Boolean
يُعرّف المتغير من نوع boolean كسائر المتغيرات، ولكن يتميز النوع boolean عن غيره من بقية الأنواع أنهيحتويعلى قيمتين فقط، أي أن أي متغير يكون نوعه boolean فستكون قيمته إما true أو false. لتوضيح الفكرة، لاحظ معي المثال التالي:

```
boolean value = true;
```

1.5 العمليات Oparators

هناك عدد من العمليات المختلفة التي يمكنك استخدامها أثناء البرمجة، مثل العمليات الرياضية وعمليات المقارنة والعمليات المنطقية وغيرها من العمليات المُختلفة. سنتحدث في هذا الجزء عن مجموعة من أهم العمليات التي توفرها لغة java.

العمليات الرياضية Arithmetic Operators
ببساطة، يمكنك تنفيذ العمليات الرياضية المختلفة باستخدام الصيغة التالية:

```
result = left op right
```

حيث يمثل op نوع العملية الرياضية المراد استخدامها، ويمثل كل من left و right القيمتين (او المتغيرين) اللذان سيتم تنفيذ العملية op عليهما. يوضح الجدول التالي أنواع العمليات الحسابية:

العملية اسم	العملية رمز	وظيفتها
Addition	+	تقوم بتنفيذ عملية الجمع
Subtraction	-	تقوم بتنفيذ عملية الطرح
Division	/	تقوم بتنفيذ عملية القسمة
Multiplication	*	تقوم بتنفيذ عملية الضرب

لتوضيح الفكرة، دعنا نقوم باستبدال op بأحد العمليات السابقة، وسنقوم هنا باختيار الجمع + كمثال يمكن تطبيقه على باقي العمليات الاخرى. يوضح السطر التالي هذا الامر:

```
int result = 5 + 2;
```

في المثال أعلاه، قمنا بتنفيذ عملية الجمع باستخدام + وسيتم تخزين ناتج العملية وهو في هذه الحالة 7 في المتغير result.

عمليات المقارنة Comparison Operators

يمكنك تنفيذ عمليات المقارنة المختلفة باستخدام الصيغة التالية (مع التنبيه على أنه يمكنك استخدامها في سياقات برمجية أخرى دون إسنادها إلى قيمة، مثل استخدامها كشرط مع جملة if كما سنرى لاحقًا):

```
result = left op right
```

يمثل op نوع عملية المقارنة المُراد استخدامها ويمثل كل من left و right القيمتين (أو المتغيرين) اللذان سيتم تنفيذ العملية op عليهما. وستكون نتيجة عمليات المقارنة هي قيمة من نوع boolean ، أي أن ناتج المقارنة سيكون إما true أو false.
يوضح الجدول التالي هذا الأمر:

العملية	استخدامها	الوصف
>	x > y	x اكبر من y
>=	x >= y	x اكبر من او يساوي y
<	x < y	x اصغرمن y
<=	x <= y	x اصغرمن او يساوي y
==	x == y	x يساوي y
!=	x != y	x لا يساوي y

Logical Operators العمليات المنطقية

هناك ثلاث عمليات منطقية، اثنتان منهما تكتب بالصيغة التالية (مع التنبيه على أنه يمكنك استخدامها في سياقات برمجية أخرى دون إسنادها إلى قيمة، مثل استخدامها كشرط مع جملة if كما سنرى لاحقًا):

```
result = left op right
```

يمثل op نوع العملية المنطقية المُراد استخدامها ويمثل كل من left و right القيمتين أو المتغيرين اللذان سيتم تنفيذ العملية op عليهما .وستكون نتيجة العمليات المنطقية هي قيمة من نوع boolean ، أي أن ناتج المقارنة سيكون إما true or false .
بالنسبة للعملية المتبقية، أي العملية الثالثة، فهي تكتب بالصيغة التالية :

```
result = left op right
```

المنطقية العمليات التالية الجداول توضح:

A	B	A&&B
FALSE	FALSE	FALSE
FALSE	TRUE	FALSE
TRUE	FALSE	FALSE
TRUE	TRUE	TRUE

A	B	A||B
FALSE	FALSE	FALSE
FALSE	TRUE	TURE
TRUE	FALSE	TRUE
TRUE	TRUE	TRUE

توضح الأسطر التالية استخدام العمليات السابقة برمجياً:

```
boolean first = true;
boolean second = false;
boolean andResult = first && second; // false
boolean orResult = first || second; // true
boolean notResult = !(5 == 10); // true
```

على نظرة Increment و Decrement

من العمليات المتكررة أثناء البرمجة، عملية زيادة واحد على قيمة المتغير الحالية أو إنقاص واحد منها. تسمى عملية الزيادة في هذه الحالة Increment وتُسمى عملية الإنقاص Decrement. لتوضيح الفكرة العامة، لاحظ معي الأسطر التالية:

```
int number = 5;
number = number + 1; // 6
number = number - 1; // 5
```

في السطر الثاني، قمنا بزيادة واحد على قيمة number، لتصبح القيمة 6 ، وهذا هو المقصود بمفهوم Increment.
وقمنا في السطر الثالث بإنقاص واحد من قيمة number ، لتصبح 5 ، وهذا هو المقصود بمفهوم Decrement.
توفر لغة java طريقة مُختصرة لتنفيذ كلتا العمليتين السابقتين، وذلك من خلال استخدام معامل الزيادة ++ لزيادة واحد على قيمة المتغير، ومعامل الإنقاص -- لإنقاص واحد من قيمة المتغير. لتوضيح الفكرة، دعنا نقوم بإعادة كتابة المثال السابق بالطريقة المُختصرة في المثال التالية:

```
int number = 5;
number++; // number = number + 1 (increment)
number--; // number = number - 1 (decrement)
```