1. أساسيات لغة الجافا

1.1 المتغيرات variables

أثناء عملك مع لغة java، 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 وتعريف متغير باسم age، وذلك لحفظ قيمة العُمر بداخله. الآن دعنا نقوم بإسناد قيمة العُمر 26 له

```
age=26;
```

لاحظ أننا قمنا باستخدام المتغير age في هذه المرة بدون استخدام التعبير 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;
```

نلاحظ في المثال أعلاه أنه يمكننا تعريف المتغير وإسناد القيمة إليه في آنٍ واحد.

إذاً، يُمكن للمبرمج استخدام age في أماكن مُختلفة 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 الثوابت Constants

تختلف الثوابت constants عن المُتغيرات variables في أنه **لا يمكن تغيير قيمتها بعد إسناد أول قيمة لها**،وستظل قيمة الثابت كما هي طيلة فترة البرنامج. لتعريف ثابت في java نستخدم كلمة final متبوعة بنوع المتغير.

لتوضيح الفكرة، لاحظ معي المثال التالي:

```
final int weekDays = 7;
```

أيام الأسبوع هي دائماً 7 أيام، أي أنه لا يمكننا تغيير عدد أيام الأسبوع، وعليه ، يمكن تمثيلها في ثابت في البرنامج كما سبقورأينا. ومن الأمثلة على الثوابت؛ تاريخ الميلاد، فالتاريخ الذي ولدت فيه ثابت ولن يتغير. يوضح السطر التالي تعريف ثابت لتمثيل تاريخ الميلاد.

```
final String dateOfBrith = "10/02/1990";
```

1.3 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، وهي:

- Single Line Commentتعليق السطر الواحد
- Multi-Line Commentتعليق متعدد الأسطر

Single Line Commentتعليق السطر الواحد

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.
```

Multi-line Commentsتعليق متعدد الأسطر

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

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

1.4 التسميات Names

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
```

وعوضا 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
```

1.5 أنواع البيانات DataTypes

تدعم لغة java عدداً 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

boolean	أي بيانات من هذا النوع تكون ضمن قيمتين وهما true وfalse.
char	يستخدم لتمثيل الأحرف

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;
```

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 أنه يحتوي على قيمتين فقط، أي أن أي متغير يكون نوعه boolean فستكون قيمته إما true أو false. لتوضيح الفكرة، لاحظ معي المثال التالي:

```
boolean value = true;
```

1.6 العمليات Operators

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 Operator

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 (أو المتغيرين أو الثابتين) اللذين سيتم تنفيذ العملية `op` عليهما. يوضح الجدول التالي أنواع العمليات الحسابية:

اسم العملية	رمز العملية	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 الأخرى. يوضح السطر التالي هذا الأمر:

```
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 أو الثابتين) اللذين سيتم تنفيذ العملية op عليهما. وستكون نتيجة عمليات المقارنة هي قيمة من نوع boolean، أي أن ناتج المقارنة سيكون إماtrue أو false.

يوضح الجدول التالي هذا الأمر:

Operator	Use	Description
>	x > y	x is greater than y
>=	x >= y	x is greater than or equal to y
<	x < y	x is less than y
<=	x <= y	x is less than or equal to y
==	x == y	x and op2 y equal
!=	x != y	x and y are not equal

Logical Operatorsالعمليات 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 أو الثابتين) اللذين 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 = op right
```

توضح الجداول التالية العمليات المنطقية:

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	FALSE
TRUE	FALSE	FALSE
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
```

Decrement و Increment نظرة على

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 ـ 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)
```

2. الشروط Conditions

2.1 تعريف الشروط Conditions

الشروط (conditions) تستخدم لتحديد طريقة عمل البرنامج نسبةً للمتغيرات التي تطرأ على الكود.

كمثال بسيط, يمكنك بناء برنامج لمشاهدة المسلسلات, عند الدخول إليه يطلب من المستخدم في البداية أن يدخل عمره لكي يقوم بعرض أفلام تناسب عمره.

يمكنك وضع العدد الذي تريده من الشروط في البرنامج الواحد، و تستطيع وضع الشروط بداخل بعضها البعض أيضاً.

2.2 تعريف الشروط Conditions

وظيفتها	اسم الجملة
نستخدمهم إذا كنا نريد تنفيذ كود معين في حال تحقق الشرط أو مجموعة من الشروط التي وضعناها.	if - else - else if statements

نستخدمها إذا كنا نريد إختبار قيمة متغير معين مع لائحة من الإحتمالات نقوم نحن بوضعها, و إذاتساوت هذه القيمة مع أي إحتمال وضعناه ستتنفذ الأوامر التي وضعناها في هذا الإحتمال فقط.	switch statement

طريقة الكتابة

```
if (condition)
   {
إذا كان الشرط صحيحا نفذ هذا الكود        //
   }

else if (condition)
   {
إذا كان الشرط صحيحا نفذ هذا الكود        //
   }

else
   {
نفذ هذا الكود في حال لم يتم التعرف على الكود في أي شرط        //
   }
```

2.2.1 جملة الشرط if

تعني باللغة العربية "اذا".و هي تستخدم فقط في حال كنت تريد تنفيذ كود معين حسب شرط معين.

```
int number = 1;

if( number < 6 )
{
   System.out.print("number is smaller than 6");
}
```

المخرجات

```
number is smaller than 6
```

2.2.2 جملة الشرط else

في اللغة العربية تعني "أي شيء آخر". وهي تستخدم فقط في حال كنا نريد تنفيذ كود معين في حال كانت نتيجة جميع الشروط التي قبلها تساوي false، يجب وضعها دائماً في الأخير, لأنها تستخدم في حال لم يتم تنفيذ أي جملة شرطية قبلها. إذاً، إذا نفذ البرنامج الجملة if أو else if فإنه سيتجاهل الجملة else. وإذا لم ينفذ أي جملة من الجمل if و else if فإنه سينفذ الجملة else

```
int age = 11;

   if( age == 10 ) {
   System.out.print("age is equal to 10");
}

   else {
   System.out.print("age is not equal to 10");
}
```

المخرجات

```
age is not equal to 10
```

2.2.3 جملة الشرط else if

جملة else if تستخدم إذا كنت تريد وضع اكثر
من احتمال (شرط) جملة او جمل الelse if
يوضعون في الوسط بين else و if

```
int number = 1;

if( number == 0 ) {
   System.out.print("Zero");
}
else if(  number == 1 ) {
   System.out.print("One");
}
 else {
   System.out.print("Negative Number");
}
```

المخرجات

```
One
```

2.3 تعريف Switch

نستخدمها إذا كنا نريد إختبار قيمة متغير معين مع لائحة من الإحتمالات نقوم نحن بوضعها فيها, و إذا تساوت هذه القيمة مع أي إحتمال وضعناه ستتنفذ الأوامر التي وضعناها في هذا الإحتمال فقط.

كل إحتمال نضعه يسمى case.

```
switch(expression) {

   case value:
      // Statements
      break;
   case value:
      // Statements
      break;
   default:
      // Statements
      break;
}
```

بعض القواعد المهمة Switch statements :

- غير مسموح بقيم Case المكررة
- يجب ان تكون قيمة Case من نفس نوع المتغير
- يجب ان تكون قيمة Case ثابتة او حرفية
- مسموح أن يتم استخدام Break داخل switch لانهاء تسلسل العبارة
- عبارة break اختيارية إذا تم حذفها، فسيستمر التنفيذ في الحالة التالية
- العبارة الافتراضية ويمكن ان تظهر في اي مكان داخل Switch block في حالة اذا لم تكن في النهاية فيجب الاحتفاظ بتعليمة break بعد العبارة الافتراضية لحذف تنفيذ case التالية:

المخرجات

```
Saturday
```