switch の ( ) の中には、整数型、String 型などの変数を書きますが、オブジェクトの型を判定する時は、❶のように、( ) には参照型の変数を書きます。指定している member は Member 型の変数です。

case には、❷❸❹のように、キャストできるかどうか判定する型と、変数を書きます。例えば、❷は、GeneralMember 型にキャストできるかどうか調べ、true の場合は、gem にキャストした値が入ります。-> の右側では、その変数を利用できます。

実際、❸❹では、変数を使って stm.getExpDate( ) や sem.age( ) などのメソッドを実行しています。例えば、member が StudentMember 型にキャストできる場合、実行すると次の様に表示されます。

```
学生会員です 期限日=2026-03-31
```

**ガードパターンと case の並び順**

特定の型にキャストできるかどうかだけでなく、それに何らかの条件を付けて判定することもできます。このような書き方を**ガードパターン**といいます。

例えば、SeniorMember にキャストできて、かつ、70 才以上かどうか判定したい場合は、次の❶のように書きます。

Java19以降は、&&の代わりにwhenを使うように変更されました。

```
String msg = switch(member) {
  case GeneralMember gem     -> "一般会員です";
  case StudentMember stm     -> "学生会員です 期限日="+stm.getExpDate();
  case SeniorMember sem && sem.age()>=70 -> "70才以上のシニア会員です";   ❶
  case SeniorMember  sem     -> "シニア会員です";     ❷
  default                    -> "会員ではありません";
};
```

❶の && に続けて書いた関係式がガードパターンの書き方です。

複数の「&& ＋関係式」を追加できます。また、&& (･･･) のように関係式全体を ( ) で囲うと、その中に、論理演算子や括弧を使って複雑な条件でも記述できます。

ただ、パターンマッチの switch では、case が上から下へ向かって順にチェックされるので、❶と❷はこの順序で並べる必要があります。❶と❷では、❷の方がより対象範囲が広いからです。もしも❷を先に置くと❶がチェックされないため、コンパイルエラーになります。

同じことは、case の右に書く型名についても言えます。A ⇒ B という継承関係にある場合は、case A を case B よりも先に書きます。つまり、サブクラスが先です。