Comments

Comments in Java are used to provide additional information about the code that may not be clear from the code itself. Even if you do not expect other people to read and try to understand your code, you should write comments for yourself, so that when you come back to your model in a couple of months you will be able to easily remember how it works, fix, or update it. Comments keep your code live and maintainable; writing good comments as you write code must become your habit.

This is an example of a useless comment:

```
client = null; //set client to null - useless comment
```

It explains things obvious from the code. Instead, you can explain the meaning of the assignment:

```
client = null; //forget the client - all operations are finished
```

In Java there are two types of comments: *end of line comments* and *block comments*. The end of line comment starts with double slash `//` and tells java to treat the text from there to the end of the current line as comment:

```
//create a new plant
Plant plant = add_mills();
//place it somewhere in the selected region
double[] loc = region.findLocation();
plant.setXY( loc[0], loc[1] );
//set the plant parameters
plant.set_region( region );
plant.set_company( this ); //we are the owner
```

AnyLogic Java editor displays the comments in green color, so you can easily distinguish them from code. Block comment is delimited by `/*` and `*/`. Unlike the end of line comment, the block comment can be placed in the middle of a line (even in the middle of an expression), or it can span across several lines.

```
/* for sales staff we inlclude also the commission part
  * that is based on the sales this quarter
  */
amount  = employee.baseSalary + commissionRate * employee.sales + bonus;
if( amount > 200000 )
    doAudit( employee ); /* perform audit for very high payments */
employee.pay( amount );
```

When you are developing a model it may be needed to temporarily exclude portions of code from compilation, for example, for debugging purposes. This is naturally done by using comments. In the expression below the commission part of the salary is temporary excluded from the expression, for example, until the commissions get modeled.

```
amount  = employee.baseSalary /* + commissionRate * employee.sales */ + bonus;
```

If you wish to exclude one or a couple of lines of code, it makes sense to put the end of line comments at the beginning of the line(s). In the code fragment below the line with the function call `ship()` will be ignored by the compiler (note that the line already contains a comment):

```
while( ! backlog.isEmpty() ) { //repeat the code below while the backlog is not empty
    Order order = backlog.getFirst(); //pick the first order in the backlog
    if( order.amount <= inventory ) { //if enough inventory to satisfy this order
//        ship( order ); //ship
        inventory -= order.amount; //decrease available inventory
        backlog.removeFirst(); //remove the order from the backlog
    } else { //not enough inventory to ship
        break; //stop order backlog processing
    }
}
```

If many lines are excluded, it is better to use block comments:

```
/*
while( ! backlog.isEmpty() ) { //repeat the code below while the backlog is not empty
    Order order = backlog.getFirst(); //pick the first order in the backlog
    if( order.amount <= inventory ) { //if enough inventory to satisfy this order
//        ship( order ); //ship
        inventory -= order.amount; //decrease available inventory
        backlog.removeFirst(); //remove the order from the backlog
    } else { //not enough inventory to ship
        break; //stop order backlog processing
    }
}
*/
```

Be careful when using comments to exclude code: you may make undesirable changes to the code nearby . Consider the code fragment below. The original plan was to perform audit for every payment that is over $200,000. The model developer decided to temporary skip the audit and commented out the line with the `doAudit()` function call. As a side effect, the next line became a part of the if statement and the payments will be made only to those employees who earn more than $200,000.

```
amount  = employee.baseSalary + commissionRate * employee.sales + bonus;
if( amount > 200000 )
//    doAudit( employee );
employee.pay( amount );
```

Note that the accident could have been avoided if the modeler had used the block braces with the if statement:

```
amount  = employee.baseSalary + commissionRate * employee.sales + bonus;
if( amount > 200000 ) {
//    doAudit( employee );
}
employee.pay( amount );
```