The Try-statement

As you now know, an attempt to divide by 0 throws an ArithmeticException, which in this program causes immediate, abnormal termination. In this activity, we introduce the try-statement, which allows the programmer to catch and handle such thrown objects.

In order to "catch" the thrown Exception, enclose the statement in a try-statement that has a catch clause attached to it that catches the thrown ArithmeticException and processes it in some fashion:

```
Calculate x;
try {
  y= 5/x;
}
catch (ArithmeticException ae) {
   System.out.println("X was 0; using 0 for 5.x");
   y= 0;
}
next statement
```

The try-statement consists of keyword **try**; followed by a block, called the try-block; followed by a catch clause, which consists of keyword catch, the declaration of a parameter, and a block, called the catch-block. The class-name that gives the type of the parameter being declared must be Throwable or one of it subclasses:

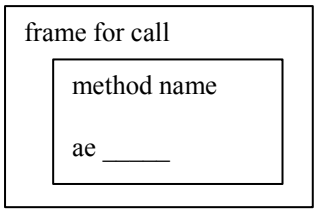
```
try <try-block>
catch (<par-dec-1>) <catch-block-1>
…
catch (<par-dec-n>) <catch-block-n>
```

where each <par-dec> has the form

```
<throwable-class   ae>      (use any variable name you want instead of ae)
```

We can see that the try-statement in the activity window has this form: Keyword **try** is followed by a block, which is followed by keyword **catch**, which is followed by a declaration of parameter ae within parentheses, which is followed by another block.

Execution of the try-statement



We describe execution of a try-statement. Assume that this statement occurs in some method that has been called and that the frame for this call contains variable ae, the parameter of the catch clause. Of course, because of the placement of the declaration of ae, ae can be referenced only within the catch-block.

Note: *Any time a method is called, a* "frame for the call" *is created, which contains the parameters and local variables of the method. After assigning arguments of the call to the parameters, the method body is executed, using the parameters and local variables that were allocated space in the frame for the call. We explain this elsewhere.*

To execute the try-statement:

Execute the try-statement. Execute the <try-block>. We have two cases to consider, depending on whether execution throws an object or not.

1. If no object is thrown, execution of the try-statement terminates when execution of the try-block does. This is the usual case.

2. If execution throws some object a0, the try-block abnormally terminates. What happens next depends on the catch clause(s) that follow the try-block.

2A. Suppose the class of some catch-clause parameter matches the class of instance a0. Choose the first one that does. Assign a0 to parameter ae and execute the catch-block. The catch-block can reference ae, so it can reference object a0. So it can look at the detail message for this thrown object and also print the call stack that is contained in the thrown object.

2B. Suppose 2A does not happen: no catch-clause catches a0. Suppose the try-statement is in some method `m`, and suppose `m` was called in a call `m(…)`. Throw object a0 to that call `m(…)` and act as if that call threw the object.