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:

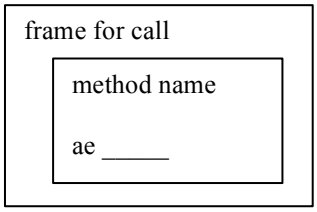
```
Calculate x;
try {
  y= 5/x;
}
catch (ArithmeticException ae) {
   System.out.println("X was 0; using 0 for 5.x");
   y= 0;
}
next statement
```

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>) <catch-block>
```

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



Now let's look at execution of a try-statement. We can 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 will show hit sin detail in later lectures.*

Execution of the try-statement begins with execution of the try-block. We have three cases to consider, depending on whether an object is thrown, and if one is thrown, whether this try-statement catches it or not.

If no object is thrown, execution of the try-statement terminates when execution of the try-block does. Execution proceeds to the next statement, following the try-statement. This is the usual case.

If an object is thrown, let's say some object a0, the try-block is abnormally terminated. What happens next depends on the catch clause that follows the try-block.

If the class of the catch-clause parameter matches the class of instance a0, a0 is assigned to the parameter and the catch-block is executed, after which execution of the try-statement terminates. 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. The catch-block in the activity window doesn't make use of ae.

We have one more case to consider. If the class of the catch-clause parameter does not match the class of object a0, the exception is thrown to another place. In other words, the exception is handled just as it would have been had there not been a try-statement. Something is guaranteed to catch the thrown object a0 and handle it. Just how this works is discussed in a later activity.