The throw statement

We write a function that calculates x mod y, for nonzero integer y. This is the value r that satisfies

x = q*y + r   and 0 <= r < abs(y) for some q.

Note that x can be any integer and y can be any negative or positive integer. The result is directly related to the remainder operation %, but we won't investigate the relation here because it would detract from our major point, which is to investigate throwing an Exception.

Note that y should not be 0. If the caller uses 0 for y, the method should throw an ArithmeticException, just the way Java does when a division by 0 occurs. This could be done simply by allowing the division by 0 to occur during a remainder operation.

```
/** = the value r that satisfies x = q*y + r   and  0 <= r < abs(y) for some q.
  *    Throw an ArithmeticException if y = 0. */
 public static int mod(int x, int y) {
     int r= x % Math.abs(y);   // throws an ArithmeticException if y = 0
     return  r >= 0 ? r : y + r;
}
```

However, we would like to insert our own detail message into the thrown object, so that the user has more specific information as to what error occurred. For this purpose we use a throw-statement:

throw <expression> ;

The <expression> must yield a throwable object --an instance of (a subclass of) class Throwable.

We look at the specification for the constructor in class ArithmeticException and write a throw-statement that throws an ArithmeticException with the desired detail message.

```
/** = the value r that satisfies x = q*y + r   and  0 <= r < abs(y) for some q.
  *    Throw an ArithmeticException if y = 0. */
 public static int mod(int x, int y) {
     if (y == 0) throw new ArithmeticException("mod(x, 0) is undefined");
     int r= x % Math.abs(y);
     return  r >= 0 ? r : y + r;

}
```

We filled in the rest of the method, but without an explanation. More important for us here is the introduction of the throw-statement. It allows us to react to errors that our programs detect just the way that the Java runtime system and all the predefined classes react to errors that they detect.