Checked exceptions and the throws clause

In this silly little program, method first throws an Exception but does not catch it. Thrown Exceptions are *checked*, which means that the compiler will issue an error message and refuse to compile the program.

```
public static void main(String[] pars) {
    first();
}
public static void first() {
    throw new Exception();
}
```

We can get around this problem by enclosing the throw statement in a try-statement, but that is probably not what we want. Instead, we would like to indicate that the calling method should catch this Exception. To do this, place a throws clause in the header of the method:

```
public static void main(String[] pars ){
    first();
}
public static void first() throws Exception {
    throw new Exception();
}
```

The throws clause has the form:

throws <class-name> , ... , <class-name>

where each class-name is Throwable or one of its subclasses. The occurrence of such a throws clause in the header of a method definition relieves the method of the responsibility of catching objects of the mentioned classes and places that burden on any method that calls this one.

In the above program, method main is now responsible for thrown Exceptions. It can relieve itself of this responsibility by having its own throws clause.

```
public static void main(String[] pars) throws Exception {
    first();
}
public static void first() throws Exception {
    throw new Exception();

}
```

The Java runtime system now has the responsibility of catching Exceptions, since it calls main, and it will do so.

RuntimeExceptions and Errors are *not* checked.