"The number entered is " + number);
  continueInput = false;
}
catch (InputMismatchException ex) {
  System.out.println("Try again. (" + "Incorrect input: an integer is required)");
  input.nextLine(); // Discard input
}
} while (continueInput);
}

Enter an integer: 3.5
Enter again. (Incorrect input: an integer is required)
Enter an integer: 4
The number entered is 4

When executing `input.nextInt()` (line 11), an InputMismatchException occurs if the input entered is not an integer. Suppose 3.5 is entered. An InputMismatchException occurs and the control is transferred to the catch block. The statements in the catch block are now executed. The statement `input.nextLine()` in line 22 discards the current input line so that the user can enter a new line of input. The variable `continueInput` controls the loop. Its initial value is true (line 6), and it is changed to false (line 17) when a valid input is received.

### 13.4 Exception Types

The preceding sections used ArithmeticException, FileNotFoundException, and InputMismatchException. Are there any other types of exceptions you can use? Yes. There are many predefined exception classes in the Java API. Figure 13.1 shows some of them.

![Figure 13.1](image)

**Figure 13.1** Exceptions thrown are instances of the classes shown in this diagram, or of subclasses of one of these classes.

**Note**
The class names Error, Exception, and RuntimeException are somewhat confusing. All three of these classes are exceptions, and all of the errors discussed here occur at runtime.