Essential Idioms

Your programming instincts will be helped by learning the idioms and patterns that implement particular problem-solving strategies. These idioms give you a considerable amount of power without requiring you to learn too many of their component details. This sheet summarizes the most important idioms, each of which is covered in more detail in the text.

The first set of idioms involves getting data in and out of the computer, which provide the necessary support for the input and output phases of a typical programming task. The idioms you use depend on the type of value, as shown in the following table:

Type	Declaration	Input idiom
Integer	`int` *var* `=` *value*`;`	*var* `= readInt("`*prompt*`");`
Floating-point	`double` *var* `=` *value*`;`	*var* `= readDouble("`*prompt*`");`
String	`String` *var* `=` *value*`;`	*var* `= readLine("`*prompt*`");`

The following idioms are useful in calculations:

English	Java (long form)	Java (shorthand form)
Add *y* to *x*.	`x = x + y;`	`x += y;`
Subtract *y* from *x*.	`x = x - y;`	`x -= y;`
Add 1 to *x* (increment *x*).	`x = x + 1;`	`x++;`
Subtract 1 from *x* (decrement *x*).	`x = x - 1;`	`x--;`

The most helpful idioms, however, encompass programming operations on a larger scale and help you establish the overall strategy of a program. The most important ones are described in the next few sections.

The repeat-N-times idiom:

This idiom is the same as it was in Karel and is used for the same purpose.

```
for (int i = 0; i < N; i++) {
    statements to be repeated
}
```

In Java, however, you are allowed to use the value of the index variable `i` in the body of the loop.

The repeat-until-sentinel idiom:

This idiom is useful whenever you need to read in values until the user enters a particular value to signal the end of input. Such values are called **sentinels**.

```
while (true) {
    prompt user and read in a value
    if (value == sentinel) break;
    rest of body
}
```