Mehran Sahami
CS 106A

Handout #11
October 6, 2017

Simple Programming Patterns

Based on a handout by Eric Roberts

This handout summarizes some of the most common programming patterns that you will encounter in Java. Each of these patterns is covered in more detail in Chapters 2, 3, or 4 of the textbook, but it's good to have them in a simple handy reference as well.

The first set of patterns 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 patterns you use depend on the type of value, as shown in the following table:

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

The following patterns 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 patterns, 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 pattern: (page 101)

This pattern 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 read-until-sentinel pattern: (page 102)

This pattern 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
}
```