ICS 111
`For` Loops

- fixed number of while loop iterations
- for loops
- sentinel values
- break statements
- loop algorithms

ICS 111
While Loops: Counting to N

```
int count = 0;
while (count < N) {
  loopBody();
  count++;
}
```

- this executes loopBody() exactly N times

ICS 111
Counting to N (variant)

```
int count = 1;
while (count <= N) {
  loopBody();
  count++;
}
```

- initializing count to 1, and testing

  count <= N, also executes loopBody() exactly N times

ICS 111
Motivation for `for` loops

- in programs, it is common to want to execute a loop a fixed number of times
  - fixed in the sense that the number is known before starting the loop
  - the number could be a constant or a variable
- this can be done with a while loop
- but it is so common that many languages have a specialized mechanism, the for loop

ICS 111
Example of for loop

```
for (int count = 0; count < N; count++) {
  loopBody();
}
```

- The for statement has a three-part section that includes, in order:
  - a statement executed once before we begin to loop
  - a condition evaluated before each loop
  - a statement executed at the end of each loop
- two semicolons separate the three parts

ICS 111
Comparing for and while

```
for (A; B; C) {
  X;
}
```

- is equivalent to:

```
A;
while (B) {
  X;
  C;
}
```

- In particular, note that the for loop doesn't have to be used for counting
  - although it often is
- Ultimately, any loop can be written as either a `for` or a `while` loop

ICS 111
Choosing `for` vs. `while`

- a counting loop is usually a for loop
- if the initialization and the update are diffused in the code, generally prefer while
  - initialization in the code before the loop,
  - update in the loop body
- otherwise, free to choose
- goal: keep the code clear

ICS 111
Example 1: infinite loop

- an infinite loop:

```
for ( ; true; ) { ...
```

  - sometimes abbreviated to:

```
for ( ; ; ) { ...
```

ICS 111
Example 2: a non-counting loop

- a non-counting trivial example:

  -
    ```
    for (String s = “”;
         ! s.equals (“aaaa”);
         s = s + “a”) {
    }
    ```

ICS 111
Example 3: Counting Characters

- counting the number of characters in a string

```
int countE = 0;
String s = "...";
for (int pos = 0; pos < s.length(); pos++) {
  if ((s.charAt(pos) == 'e') ||
      (s.charAt(pos) == 'E')) {
    countE++;
  }
}
```

ICS 111
Sentinel Values

- When going through a list of values, you may want to use a special value to mark the end of the list
- For example you can prompt a user to:

  enter the next number, or -1 if done
- Here -1 is not a valid input, and so can be used to indicate something special -- in this case, it shows that the input is done
- Such a special value is called a Sentinel
  - sentinel also means sentry -- in computer science, a sentinel value means pay attention, we are doing something different here

ICS 111
Break Statements

- We have seen `break` in `switch` statements, where it means “don't execute the next statement. Instead, end execution of the switch statement”
- `break` means the same thing in loops:
  - end this loop immediately

```
for (int i = 0; i < 1000; i++) {
  int value = Scanner.nextInt();
  if (value < 0) {  // we are using any value < 0 as a sentinel
    break;              // in this case, a value < 0 means we are done
  }
  ...
}
```

ICS 111
Using Break Statements in Loops

- We can use a break statement when at least one of the loop terminating conditions is tested inside the loop body
- Especially if the condition can only be evaluated after the first part of the loop body has been executed
  - as in the previous example
- break statements give us more flexibility when loop termination is complicated

ICS 111
Loop Algorithms: Statistics

- Sum, Average/Mean, Min, Max

```
double min = Math.MAX_VALUE;     // the largest possible double
double max = - Math.MAX_VALUE;   // the most negative double
double sum = 0.0;
int count = 0;
while (in.hasNextDouble()) {
  double value = in.nextDouble();
  sum = sum + value;                  // sum += value;
  if (value < min) { min = value; }   // a new, lower minimum
  if (value > max) { max = value; }   // a new, higher maximum
  count++;
}
double mean = ((count > 0) ? sum / count : 0);
```

ICS 111
Using a Value from the Last Loop

- we are reading input from the user
- we want to report if the user enters the same string twice in a row
- we must use a variable to save the value from the last loop
- and we need a special case for the first time

ICS 111
Value from the Last Loop: check for first loop

```
String lastInput = “SENTINEL STRING”;
while (true) {
  String thisInput = in.nextLine();
  if (!lastInput.equals(“SENTINEL STRING”) &&
      lastInput.equals(thisInput)) {
    System.out.println(“same string: “ + thisInput);
  }
  lastInput = thisInput;
}
```

ICS 111
Value from the Last Loop: repeat code before the loop

```
String lastInput = in.nextLine();
while (true) {
  String thisInput = in.nextLine();
  if (lastInput.equals(thisInput)) {
    System.out.println(“same string: “ +
                       thisInput);
  }
  lastInput = thisInput;
}
```

Summary

- for loops are mostly used when counting a fixed number of loops
  - but are completely general
- sentinel values are values that are not valid, and can be used to mark something special
- break statements go to the end of a loop (or a switch statement)
- loops, with sequences and conditionals, give us the power to write interesting programs