Recall our example from last time:

```
public class F2Cwhile {
    public static void main(String[] args) {
    double fahr;
    double celsius;

    do{
        System.out.print("Enter Temperature in Fahernheit : ");
        fahr=IO.readDouble();
        if (fahr<-459.67){
           System.out.println("Temperature must be >= 459.67 ");
        }
    }
    while (fahr<-459.67);
    celsius=(fahr-32)*5/9;
    IO.outputDoubleAnswer(celsius);
    }
}
```

Question:

Is it possible to achieve the same effect using a while as a do-while?

The answer is YES. Let us try it on the Faherenheit to Celsuis conversion program.

If we see this do-while as pseudocode, we get:

```
do{
    read fahr
    if fahr<-459.67{
        bad input
    }
}
while (fahr < -459.67)
celsius=(fahr-32)*5/9
```

Notice that the condition circled in red repeats!
If something repeats in a program we have to be very careful to modify both places in case we have to alter the code!

Now using while

```
read fahr
while (fahr < -459.67){
        bad input
        read fahr
}
celsius=(fahr-32)*5/9;
```

Again there is repeated code.

But the way this program looks from the user's point of view is the same as using do-while

Sentinel

Suppose that we want to prompt the user for several values but the program doesn't know how many values the user will enter.

We use a special value called *"sentinel"* so that when the user enters that value, the program stops prompting for more.

Version 1 (do-while)

```
do{
   read value
   if (value != sentinel){
      DO SOMETHING WITH
      CURRENT value
}
while (value != sentinel)
```

Version 2 (while)

```
read value
while (value != sentinel){
   DO SOMETHING WITH
   CURRENT value
   read value
}
```

Summary variables

So far we are able to prompt the user repeatedly for a value until the user types the *sentinel*, this will allow us to use the values that the user typed to compute *summary* values based on the values given by the user, those *summary* values include:

- Count
- Sum
- Average
- Maximum (largest)
- Minimum (smallest)

We use summary variables to hold summary values. In order to compute summary variables we must follow the following pattern:

Summary variables (continued)

We use summary variables to hold summary values. In order to compute summary variables we must follow the following pattern:

1) Outside the loop: initialize the variable to an initial value consistent with our summary variable (usually 0).
2) Update the value of the summary variable in the loop using the current value given by the user.
3) Display or use the summary variable after the loop.

Example: compute the sum of the values given by the user

We will use variable `sum` as our summary variable.

1) Initialize summary variable (before the loop): `sum=0;`
2) Update the summary variable in the loop by adding the current value given by the user to the current sum: `sum=sum+value;`
3) Print `sum`

The idea is that at any point during execution sum will have the accumulated sum of all the previous values, so that to update it, we just have to add the last value read.

Putting it all together:

```
public class Sum
{
    public static void main(String[] args)
    {
       double sentinel,value,sum;

       System.out.print("Enter sentinel: ");
       sentinel=IO.readDouble();

       System.out.print("Enter value: ");
       value=IO.readDouble();
       sum=0;
       while (value!=sentinel){
          sum=sum+value;
          System.out.print("Enter value: ");
          value=IO.readDouble();
       }
       System.out.println("sum="+sum);
    }
}
```

Sample execution:

```
Enter sentinel: -1
Enter value: 4
Enter value: 8
Enter value: 10
Enter value: -1
sum=22.0
```

Computing the average of a list of numbers

To compute the average of a list we need two summary variables:

- `sum`
- `count`

`count` keeps track of the number of values (not including the sentinel) that the user has entered:

- Initialization: `count=0;`
- Update: `count++;`

After all the values have been entered, the average can be computed using (after the loop):

```
average = sum/count;
```

We just have to make sure that count is not zero

Program to compute the average of a list of numbers:

```
double sentinel,value,sum,average;
int count;
System.out.print("Enter sentinel: ");
sentinel=IO.readDouble();
System.out.print("Enter value: ");
value=IO.readDouble();
sum=0;
count=0;
while (value!=sentinel){
    sum=sum+value;
    count++;
    System.out.print("Enter value: ");
    value=IO.readDouble();
}
if (count==0){
    System.out.println("There is no data");
}
else{
    average=sum/count;
    System.out.println("average="+average);
}
```

Sample execution:

```
Enter sentinel: -1
Enter value: 5
Enter value: 6
Enter value: 7
Enter value: 6
Enter value: 5
Enter value: -1
average=5.8
```