CS100J 1 March, 2006

Loops, iterative statements, or repetitive statements

Start reading Sec. 2.3.8 and chapter 7 on loops.
The lectures on the ProgramLive CD can be a big help.

> **Learning without thought is labor lost. Thought without learning is perilous.**
>
> or, alternatively,
>
> **Study without reflection is a waste of time; reflection without study is dangerous.**
>
> -- Confucius

The for loop, for processing a range of integers

```
x= 0;
// add the squares of ints
// in range 2..200 to x
x= x + 2*2;
x= x + 3*3;
…
x= x + 200;
```

for each number i in the range 2..200, add i*i to x.

The for-loop:

```
for (int i= 2; i <= 200; i= i +1) {
   x= x + i*i;
}
```

loop counter: i
initialization: **int** i= 2;
loop condition: i <= 200;
increment: i= i + 1
repetend or body: { x= x + i*i; }

repetend: the thing to be repeated. The block:

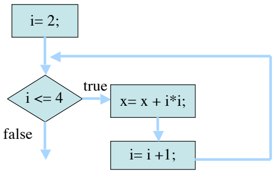
{ x= x + i*i; }

Execution of the for-loop

The for-loop:

```
for (int i= 2; i <= 4; i= i +1) {
   x= x + i*i;
}
```

loop counter: i
initialization: **int** i= 2;
loop condition: i <= 4;
increment: i= i + 1
repetend or body: { x= x + i; }



To execute the for-loop.

1. Execute initialization.
2. If loop condition is false, terminate execution.
3. Execute the repetend.
4. Execute the increment and repeat from step 2.

Called a "flow chart"

Execution of the for-loop

The for-loop:

```
for (int i= 2; i <= 4; i= i +1) {
   x= x + i*i;
}
```

loop counter: i
initialization: **int** i= 0;
loop condition: i <= 4;
increment: i= i + 1
repetend or body: { x= x + i; }

Trace execution of for-loop. We do it as shown below, rather than using a single box, for x and one for i, so that we can keep track of when events happened.

x	0		4		13		29		
i		2		3		4		5	

The pattern for processing range of integers:

range a..b-1

```
for (int k= a;  k != b;  k= k + 1) {
      Process integer k;
}
```

range c..d

```
for (int i= c;  i <= d;  i= i + 1) {
      Process integer i;
}
```

```
// Print the integers in 10..n-1
// inv: All ints in 10..k-1 been printed
for (int k= 10; k != n;  k= k +1) {
   System.out.println(k);
}
// All ints in 10..n-1 been printed
```

```
// Print the integers in 1..10
// inv: All ints in 10..i-1 printed
for (int i= 1; i <= 10;  i= i +1) {
   System.out.println(i);
}
// All ints in 10..i-1 printed
```

The pattern for processing range of integers:

range a..b-1

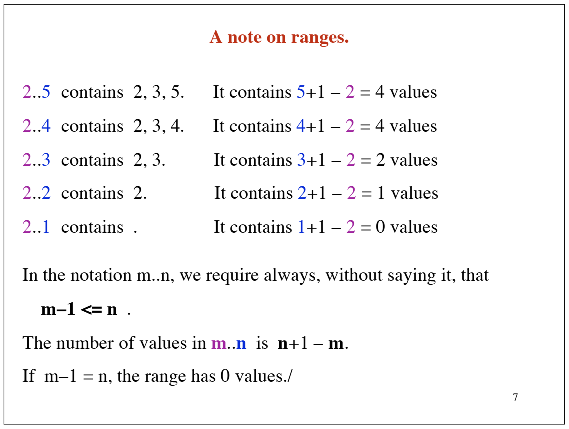
```
for (int i= a;  i != b;  i= i + 1) {
      Process integer i;
}
```

range c..d

```
for (int i= c;  i <= d;  i= i + 1) {
      Process integer i;
}
```

```
// Print the indices of all 'e's in  String s
// inv: Indices of 'e's in s[0..s.i-1]
for (int i= 0; i != s.length();  i= i +1) {
   if (s.charAt(i) == 'e')
      System.out.println(i);
}
// Indices of 'e's in s[0..s.length()-1]
// printed
```

```
// Store in double variable v the sum
//      1/1   + 1/2 + …+ 1/n
v= 0;
// inv: 1/1   + 1/2 + …+ 1/(i-1)
for (int i= 1; i <= n;  i= i +1) {
            v= v + 1.0 / i;
}
// v= 1/1   + 1/2 + …+ 1/n
```



A note on ranges.

2..5 contains 2, 3, 5.   It contains 5+1 – 2 = 4 values

2..4 contains 2, 3, 4.   It contains 4+1 – 2 = 4 values

2..3 contains 2, 3.   It contains 3+1 – 2 = 2 values

2..2 contains 2.   It contains 2+1 – 2 = 1 values

2..1 contains .   It contains 1+1 – 2 = 0 values

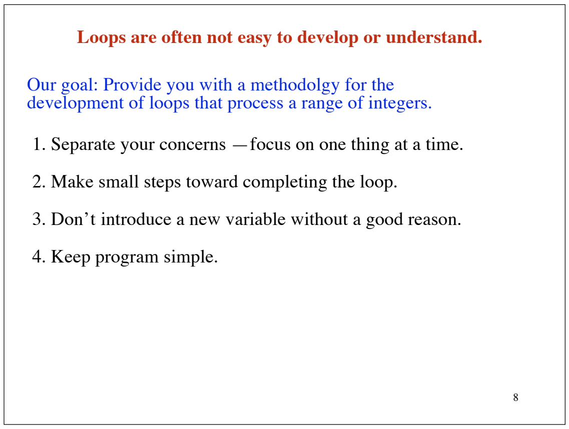
In the notation m..n, we require always, without saying it, that

m–1 <= n .

The number of values in **m**..**n** is **n**+1 – **m**.

If m–1 = n, the range has 0 values./

7

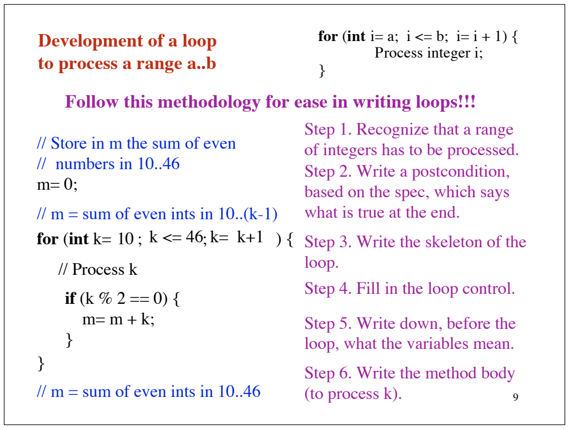


Loops are often not easy to develop or understand.

Our goal: Provide you with a methodolgy for the development of loops that process a range of integers.

1. Separate your concerns — focus on one thing at a time.
2. Make small steps toward completing the loop.
3. Don't introduce a new variable without a good reason.
4. Keep program simple.

8



Development of a loop to process a range a..b

```
for (int i= a;  i <= b;  i= i + 1) {
              Process integer i;
}
```

Follow this methodology for ease in writing loops!!!

```
// Store in m the sum of even
//  numbers in 10..46
m= 0;
// m = sum of even ints in 10..(k-1)
for (int k= 10 ;  k <= 46; k=  k+1   ) {
     // Process k
      if (k % 2 == 0) {
          m= m + k;
      }
}
// m = sum of even ints in 10..46
```

Step 1. Recognize that a range of integers has to be processed.

Step 2. Write a postcondition, based on the spec, which says what is true at the end.

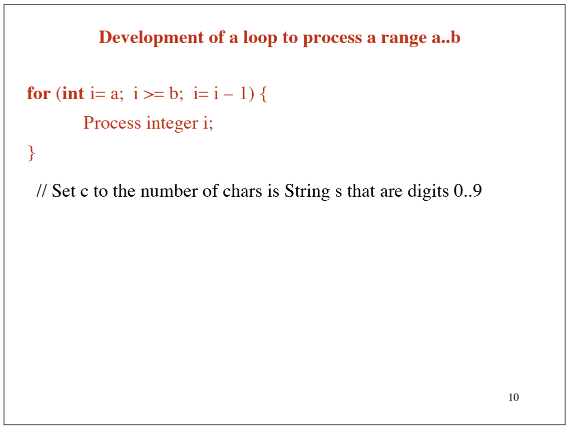
Step 3. Write the skeleton of the loop.

Step 4. Fill in the loop control.

Step 5. Write down, before the loop, what the variables mean.

Step 6. Write the method body (to process k).

9

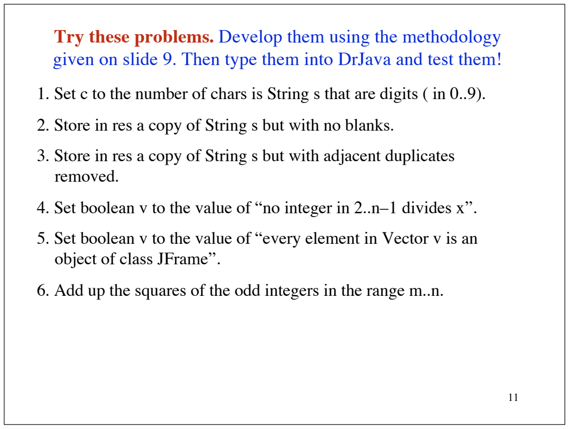


Development of a loop to process a range a..b

```
for (int i= a;  i >= b;  i= i – 1) {
          Process integer i;
}
```

// Set c to the number of chars is String s that are digits 0..9

10



Try these problems. Develop them using the methodology given on slide 9. Then type them into DrJava and test them!

1. Set c to the number of chars is String s that are digits (in 0..9).
2. Store in res a copy of String s but with no blanks.
3. Store in res a copy of String s but with adjacent duplicates removed.
4. Set boolean v to the value of "no integer in 2..n–1 divides x".
5. Set boolean v to the value of "every element in Vector v is an object of class JFrame".
6. Add up the squares of the odd integers in the range m..n.

11