CS100J October 07, 2003
Loops

Repetitive statements, or **iterative statements**, or **loops**

Start reading chapter 7 on loops. The lab today will continue to discuss loops.

"O! Thou hast damnable iteration and art, indeed, able to corrupt a saint." Shakespeare, *Henry IV*, Pt I, 1 ii

"Use not vain repetition, as the heathen do."
Matthew V, 48

Your "if" is the only peacemaker; much virtue if "if".
Shakespeare, *As You Like It*.

The while loop

```
System.out.println(5*5);
System.out.println(6*6);
System.out.println(7*7);
System.out.println(8*8);
```

```
int k= 5;
while ( k != 9) {
    System.out.println(k*k);
    k= k+1;
}
```

To execute the while loop:

(1) evaluate condition k != 9;

if it is false, stop execution.

(2) Execute the repetend.

(3) Repeat again from step (1).

Repetend: the thing to be repeated. The block:

```
    {
  …
}
```

The while loop

```
int k= 5;
while ( k != 9) {
    System.out.println(k*k);
    k= k+1;
}
```

To execute the while loop:

(1) evaluate condition k != 9;

if it is false, stop execution.

(2) Execute the repetend.

(3) Repeat again from step (1).

Trace execution of the loop: Section 7.1.2 shows you how to "trace" execution of a loop, showing the values of variables as you go. STUDY THIS SECTION!

The while loop: syntax

while (<*condition*>)

<*repetend*>

<*condition*>: a boolean expression.

<*repetend*>: a statement.

while (<*condition*> {

sequence of declarations

and statements

}

BUT: We always make the <*repetend*> a block.

Using assertions to understand a while loop

```
int k= 5;

while (k != 9 ) {
    System.out.println(k*k);
    k= k+1;
}
// {squares of 5..9-1 printed}
```

```
System.out.println(5*5);

System.out.println(6*6);

System.out.println(7*7);

System.out.println(8*8);
// {squares of 5..9-1 printed}
```

Using assertions to understand a while loop

```
int k= 5;
// {invariant: squares of 5..(k-1) printed }
while ( k != 9) {
   System.out.println(k*k);
   k= k+1;
}
// {k = 9}
```

```
// { squares of 5..4 printed}
System.out.println(5*5);
// { squares of 5..5 printed}
System.out.println(6*6);
// { squares of 5..6 printed}
System.out.println(7*7);
// { squares of 5..7 printed}
System.out.println(8*8);
// { squares of 5..8 printed}
```

Using assertions to understand a while loop

```
int k= 5;
// { invariant: Squares of values in 5..k-1 have been printed }
while ( k != 9) {
   System.out.println(k*k);
   k= k+1;
} // {postcondition: Squares of 5..8 have been printed}
```

Four loopy questions:

1. **How does it start?** Initialize to make invariant true?
2. **When does it stop?** Is the postcondition is true?
3. **How does it make progress?**
4. **How does repetend fix the invariant?**

Using assertions to understand a while loop

```
int k= ?;
int x= ?;
while ( ?  ) {
    ?

}
// {x =  sum of 0..3}
```

```
x= 0;

x= x+1;

x= x+2;

x= x+3;
// {x =  sum of 0..3}
```

1.How does it start?

2. When does it stop?

3. How does it make progress?

4. How does it fix the invariant?

Using assertions to understand a while loop

```
int k= 0;
int x= 0;
// { invariant: x = sum of 0..k-1 }
// inv: x =
while ( k != 4   ) {
  x=  x+ k;              //    k= k+1;
  k= k+1;                //    x= x+k;
} // { x = sum of 0..3 }
```

1. **How does the loop start?**
2. **How does it stop?**
3. **How does it make progress?**
4. **How does it fix the invariant?**

```
x= 0;
// { x = sum of 0..0 }
x= x+1;
// { x = sum of 0..1 }
x= x+2;
// { x = sum of 0..2 }
x= x+3;
// { x = sum of 0..3 }
```

Generalization:

```
// { x = sum of 0..k }
```