ALGONQUIN COLLEGE

LOOP STATEMENTS IN JAVA

while statement
do while statement
for statement

Kumari Gurusamy
Source Credit: Howard Rosenblum

Repetition

- Loops are used to repeat parts of program

Repetition

- Loops are used to repeat parts of program
- Often the programmer doesn't know how many times to repeat

Repetition

- Loops are used to repeat parts of program
- Often the programmer doesn't know how many times to repeat
- Need to control number of repetitions – use a loop control variable (LCV)

Repetition

- Loops are used to repeat parts of program
- Often the programmer doesn't know how many times to repeat
- Need to control number of repetitions – use a loop control variable (LCV)
  - initial value

Repetition

- Loops are used to repeat parts of program
- Often the programmer doesn't know how many times to repeat
- Need to control number of repetitions – use a loop control variable (LCV)
  - initial value
  - how the value changes

Repetition

- Loops are used to repeat parts of program
- Often the programmer doesn't know how many times to repeat
- Need to control number of repetitions – use a loop control variable (LCV)
  - initial value
  - how the value changes
  - cessation value (to stop loop - so it doesn't go forever)

Repetition

- Loops are used to repeat parts of program
- Often the programmer doesn't know how many times to repeat
- Need to control number of repetitions – use a loop control variable (LCV)
  - initial value
  - how the value changes
  - cessation value (to stop loop - so it doesn't go forever)
  - final value of LCV can often be valuable

While Statement

- basic loop statement

While Statement

- basic loop statement
- Syntax:

```
while (boolean expression) {
    statement(s);
}
```

While Statement

- basic loop statement
- Syntax:
```
while (boolean expression) {
    statement(s);
}
```
- Process:
  - Check boolean expression
    - if false continue processing at statement after }
    - if true, execute all statement(s) in the loop body in order, and then go back and evaluate the boolean expression

Example

- Want to write a loop that displays “numOfStars” asterisks to screen
- Figure out LCV - not numOfStars…use count

```
int numOfStars,  count;

System.out.print (“\nEnter number of stars to display:”);
numOfStars = input.getInt();

count = 0;                                          // initialize LCV
 System.out.println (“\nHere are stars: )”;
 while (count < numOfStars) {   // LCV part of test to end loop
            System.out.print ( “*”);
            count ++;                  // LCV changes in loop body
 }
```

Example Analysis

```
int numOfStars,  count;

  System.out.print ("\nEnter number of stars to display:");
  numOfStars = input.getInt();
```

Let's say we enter 3 …

Memory has numOfStars<---->3

Example Analysis

```
int numOfStars,  count;

 System.out.print ("\nEnter number of stars to display:");
 numOfStars = input.getInt();

count = 0;                                  // initialize LCV
System.out.println ("\nHere are stars: )";
while (count < numOfStars) {          // LCV part of test to end loop
          System.out.print ( "*");
          count ++;                         // LCV changes in loop body

}
```

Memory has numOfStars < -- >3
count < -- >0

Example Analysis

```
int numOfStars,  count;

  System.out.print ("\nEnter number of stars to display:");
  numOfStars = input.getInt();

 count = 0;                                        // initialize LCV
System.out.println ("\nHere are the stars: )";
while (count < numOfStars) {   // LCV part of test to end loop
          System.out.print ( "*");
          count ++;                // LCV changes in loop body
}
```

Memory has numOfStars < -- >3
count < -- >0

Screen has Here are the stars:

Example Analysis

```
int numOfStars,  count;

  System.out.print (“\nEnter number of stars to display:”);
  numOfStars = input.getInt();

 count = 0;                                          // initialize LCV
 System.out.println (“\nHere are the stars: )”;
 while (count < numOfStars) {   // LCV part of test to end loop
            System.out.print ( “*”);
            count ++;                 // LCV changes in loop body
 }
```

Memory has numOfStars < -- >3
count < -- >0

Screen has Here are the stars:

Example Analysis

```
int numOfStars,  count;

System.out.print ("\nEnter number of stars to display:");
numOfStars = input.getInt();

count = 0;                                          // initialize LCV
System.out.println ("\nHere are the stars: )";
while (count < numOfStars) {          // LCV part of test to end loop
        System.out.print ( "*");
        count ++;                     // LCV changes in loop body
}
```

Memory has numOfStars < -- >3
count < -- >0

Screen has Here are the stars:
*

Example Analysis

```
int numOfStars,  count;

 System.out.print (“\nEnter number of stars to display:”);
 numOfStars = input.getInt();

count = 0;                                              // initialize LCV
System.out.println (“\nHere are the stars: )”;
while (count < numOfStars) {          // LCV part of test to end loop
            System.out.print ( “*”);
            count ++;                          // LCV changes in loop body
}
```

Memory has numOfStars < -- >3
count < -- >1

Screen has Here are the stars:
*

Example Analysis

```
int numOfStars,  count;

  System.out.print ("\nEnter number of stars to display:");
  numOfStars = input.getInt();

 count = 0;                                              // initialize LCV
 System.out.println ("\nHere are the stars: )";
 while (count < numOfStars) {          // LCV part of test to end loop
              System.out.print ( "*");
              count ++;                          // LCV changes in loop body
 }
```

Memory has numOfStars < -- >3
count < -- >1

Screen has Here are the stars:
*

Example Analysis

```
int numOfStars,  count;

 System.out.print ("\nEnter number of stars to display:");
 numOfStars = input.getInt();

count = 0;                                              // initialize LCV
System.out.println ("\nHere are the stars: )";
while (count < numOfStars) {          // LCV part of test to end loop
            System.out.print ( "*");
            count ++;                          // LCV changes in loop body
}
```

Memory has numOfStars < -- >3
count < -- >1

Screen has Here are the stars:
**

Example Analysis

```
int numOfStars,  count;

  System.out.print ("\nEnter number of stars to display:");
  numOfStars = input.getInt();

 count = 0;                                              // initialize LCV
 System.out.println ("\nHere are the stars: )";
 while (count < numOfStars) {          // LCV part of test to end loop
            System.out.print ( "*");
            count ++;                  // LCV changes in loop body
 }
```

Memory has numOfStars < -- >3
count < -- >2

Screen has Here are the stars:
**

Example Analysis

```
int numOfStars,  count;

 System.out.print ("\nEnter number of stars to display:");
 numOfStars = input.getInt();

count = 0;                                              // initialize LCV
System.out.println ("\nHere are the stars: )";
while (count < numOfStars) {          // LCV part of test to end loop
            System.out.print ( "*");
            count ++;                          // LCV changes in loop body
}
```

Memory has numOfStars < -- >3
count < -- >2

Screen has Here are the stars:
**

Example Analysis

```
int numOfStars,  count;

 System.out.print ("\nEnter number of stars to display:");
 numOfStars = input.getInt();

count = 0;                                              // initialize LCV
System.out.println ("\nHere are the stars: )";
while (count < numOfStars) {          // LCV part of test to end loop
            System.out.print ( "*");
            count ++;                          // LCV changes in loop body
}
```

Memory has numOfStars < -- >3
count < -- >2

Screen has Here are the stars:
***

Example Analysis

```
int numOfStars,  count;

 System.out.print (“\nEnter number of stars to display:”);
 numOfStars = input.getInt();

count = 0;                                              // initialize LCV
System.out.println (“\nHere are the stars: )”;
while (count < numOfStars) {          // LCV part of test to end loop
            System.out.print ( “*”);
            count ++;                          // LCV changes in loop body
}
```

Memory has numOfStars < -- >3
count < -- >3

Screen has Here are the stars:
***

Example Analysis

```
int numOfStars,  count;

  System.out.print ("\nEnter number of stars to display:");
  numOfStars = input.getInt();

 count = 0;                                              // initialize LCV
 System.out.println ("\nHere are the stars: )";
 while (count < numOfStars) {          // LCV part of test to end loop
            System.out.print ( "*");
            count ++;                  // LCV changes in loop body
 }
```

Memory has numOfStars < -- >3
count < -- >3

Screen has Here are the stars:
***

LCV Notes

- We started LCV count at 0, incremented by one in each iteration of loop and tested for final value count < numOfStars(so loop stopped when count became equal to numOfStars)

LCV Notes

- We started LCV count at 0, incremented by one in each iteration of loop and tested for final value count < numOfStars(so loop stopped when count became equal to numOfStars)
- In effect, count contained the number of stars currently displayed to the monitor

LCV Notes

- We started LCV count at 0, incremented by one in each iteration of loop and tested for final value count < numOfStars(so loop stopped when count became equal to numOfStars)
- In effect, count contained the number of stars currently displayed to the monitor
- Works for case numOfStars = = 0

LCV Notes

- We started LCV count at 0, incremented by one in each iteration of loop and tested for final value count < numOfStars(so loop stopped when count became equal to numOfStars)
- In effect, count contained the number of stars currently displayed to the monitor
- Works for case numOfStars = = 0
- Takes practice to get the LCV values correct

LCV Notes

- We started LCV count at 0, incremented by one in each iteration of loop and tested for final value count < numOfStars (so loop stopped when count became equal to numOfStars)
- In effect, count contained the number of stars currently displayed to the monitor
- Works for case numOfStars = = 0
- Takes practice to get the LCV values correct
  - Could also have started count at 1 and stopped at count <= numOfStars count would go from 1, 2, 3, 4 - stop at 4

LCV Notes - cont

- If we had started count at 1 and stopped at count < numOfStars count would go from 1, 2, 3 - stop at 3 ---and only two stars would be displayed - one too few

LCV Notes - cont

- If we had started count at 1 and stopped at count < numOfStars count would go from 1, 2, 3 - stop at 3 ---and only two stars would be displayed - one too few

- If we had started count at 0 and stopped at count <= numOfStars count would go from 0, 1, 2, 3, 4 - stop at 4 ---and four stars would be displayed - one too many

LCV Notes - cont

- If we had started count at 1 and stopped at count < numOfStars count would go from 1, 2, 3 - stop at 3 ---and only two stars would be displayed - one too few
- If we had started count at 0 and stopped at count <= numOfStars count would go from 0, 1, 2, 3, 4 - stop at 4 ---and four stars would be displayed - one too many
- These are common errors for loops so you need to be very careful with LCV boundary values...there is not only one way to code the loop so it produces what you want....

Do while statement

- Syntax:
```
do {
    statement(s);
} while (boolean expression);
```
- Process:
  - Executes statement(s) in loop body in order
  - Check boolean expression
    - if true go back and execute statements in loop body again
    - if false continue at statement after the ;

  - **NOTE: THIS LOOP ALWAYS EXECUTES ONCE**

Example

- Want to write a loop that displays “numOfStars” asterisks to screen

```
int numOfStars,  count;

 System.out.print (“\nEnter number of stars to display:”);
 numOfStars = input.getInt();

count = 0;                                          // initialize LCV
  System.out.println (“\nHere are the stars: )”;
  do  {
            System.out.print ( “*”);
            count ++;                 // LCV changes in loop body
  } while (count < numOfStars);        // LCV part of test to end loop
```

Example Analysis

```
int numOfStars,  count;

System.out.print ("\nEnter number of stars to display:");
numOfStars = input.getInt();

count = 0;                                          // initialize LCV
 System.out.println ("\nHere are the stars: )";
 do  {
            System.out.print ( "*");
            count ++;                 // LCV changes in loop body
 } while (count < numOfStars);        // LCV part of test to end loop
```

Memory has numOfStars <---->2
            count <---->0

Example Analysis

```
int numOfStars,  count;

System.out.print ("\nEnter number of stars to display:");
numOfStars = input.getInt();

count = 0;                                          // initialize LCV
 System.out.println ("\nHere are the stars: )";
 do  {
            System.out.print ( "*");
            count ++;                 // LCV changes in loop body
 } while (count < numOfStars);        // LCV part of test to end loop
```

Memory has numOfStars <---->2
            count <---->0

Example Analysis

```
int numOfStars,  count;

 System.out.print ("\nEnter number of stars to display:");
 numOfStars = input.getInt();

count = 0;                                          // initialize LCV
  System.out.println ("\nHere are the stars: )";
  do  {
            System.out.print ( "*");
            count ++;                 // LCV changes in loop body
  } while (count < numOfStars);        // LCV part of test to end loop
```

Memory has numOfStars <---->2
            count <---->0

Screen has Here are the stars:

Example Analysis

```
int numOfStars,  count;

 System.out.print (“\nEnter number of stars to display:”);
 numOfStars = input.getInt();

count = 0;                                          // initialize LCV
  System.out.println (“\nHere are the stars: )”;
  do  {
            System.out.print ( “*”);
            count ++;                 // LCV changes in loop body
  } while (count < numOfStars);        // LCV part of test to end loop
```

Memory has numOfStars <---->2
            count <---->0

Screen has Here are the stars:
            *

Example Analysis

```
int numOfStars,  count;

System.out.print ("\nEnter number of stars to display:");
numOfStars = input.getInt();

count = 0;                                  // initialize LCV
System.out.println ("\nHere are the stars: )";
do  {
        System.out.print ( "*");
        count ++;                           // LCV changes in loop body

} while (count < numOfStars);        // LCV part of test to end loop
```

Memory has numOfStars <---->2
count <---->1

Screen has Here are the stars:
*

Example Analysis

```
int numOfStars,  count;

 System.out.print (“\nEnter number of stars to display:”);
 numOfStars = input.getInt();

count = 0;                                          // initialize LCV
  System.out.println (“\nHere are the stars: )”;
  do  {
            System.out.print ( “*”);
            count ++;                 // LCV changes in loop body
  } while (count < numOfStars);        // LCV part of test to end loop
```

Memory has numOfStars <---->2
            count   <---->1

Screen has Here are the stars:
            *

Example Analysis

```
int numOfStars,  count;

 System.out.print ("\nEnter number of stars to display:");
 numOfStars = input.getInt();

count = 0;                                          // initialize LCV
  System.out.println ("\nHere are the stars: )";
  do  {
            System.out.print ( "*");
            count ++;                 // LCV changes in loop body
  } while (count < numOfStars);        // LCV part of test to end loop
```

Memory has numOfStars <---->2
            count   <---->1

Screen has Here are the stars:
            **

Example Analysis

```
int numOfStars,  count;

 System.out.print (“\nEnter number of stars to display:”);
 numOfStars = input.getInt();

count = 0;                                          // initialize LCV
  System.out.println (“\nHere are the stars: )”;
  do  {
              System.out.print ( “*”);
              count ++;                 // LCV changes in loop body
  } while (count < numOfStars);        // LCV part of test to end loop
```

Memory has numOfStars <---->2
            count <---->2

Screen has Here are the stars:
            **

Example Analysis

```
int numOfStars,  count;

 System.out.print ("\nEnter number of stars to display:");
 numOfStars = input.getInt();

count = 0;                                          // initialize LCV
  System.out.println ("\nHere are the stars: )";
  do  {
            System.out.print ( "*");
            count ++;                 // LCV changes in loop body
  } while (count < numOfStars);        // LCV part of test to end loop
```

Memory has numOfStars <---->2
            count   <---->2

Screen has Here are the stars:
            **

LCV Notes

- We started LCV count at 0, incremented by one in each iteration of loop and tested for final value count < numOfStars (so loop stopped when count became equal to numOfStars)
- In effect, count contained the number of stars currently displayed to the monitor
- DOES NOT work for case numOfStars = = 0 - we always get one iteration
- Takes practice to get the LCV values correct - just like while statement
  - Same errors possible as with while loop

for statement

- Syntax:

```
for (initialization; boolean expression; increment)  {
        statement(s);
}
```

- Process:
  - Executes initialization
  - Check boolean expression
    - if true execute statements in loop body in order; then execute increment; then check boolean expression again
    - if false continue at statement after the }

Example

- Want to write a loop that displays “numOfStars” asterisks to screen

```
int numOfStars,  count;

System.out.print (“\nEnter number of stars to display:”);
numOfStars = input.getInt();

System.out.println (“\nHere are the stars: )”;
for (count = 0;  count < numOfStars;  count++ )  {

        System.out.print ( “*”);
}
```

Example Analysis

```
int numOfStars,  count;

System.out.print (“\nEnter number of stars to display:”);
numOfStars = input.getInt();

System.out.println (“\nHere are the stars: )”;
for (count = 0;  count < numOfStars;  count++  ) {
            System.out.print ( “*”);
}
```

Memory has numOfStars <----> 2
count <----> 0

Screen has Here are the stars:

Example Analysis

```
int numOfStars,  count;

System.out.print (“\nEnter number of stars to display:”);
numOfStars = input.getInt();

System.out.println (“\nHere are the stars: )”;
for (count = 0;  count < numOfStars;  count++  ){
            System.out.print ( “*”);
}
```

Memory has numOfStars <---->2
count <---->0

Screen has Here are the stars:

Example Analysis

```
int numOfStars,  count;

System.out.print (“\nEnter number of stars to display:”);
numOfStars = input.getInt();

System.out.println (“\nHere are the stars: )”;
for (count = 0;  count < numOfStars;  count++ ) {
            System.out.print ( “*”);
}
```

Memory has numOfStars <---->2
count <---->0

Screen has Here are the stars:

Example Analysis

```
int numOfStars,  count;

System.out.print (“\nEnter number of stars to display:”);
numOfStars = input.getInt();

System.out.println (“\nHere are the stars: )”;
for (count = 0;  count < numOfStars;  count++ ) {
        System.out.print ( “*”);
}
```

Memory has numOfStars <---->2
count <---->0

Screen has Here are the stars:
*

Example Analysis

```
int numOfStars,  count;

System.out.print (“\nEnter number of stars to display:”);
numOfStars = input.getInt();

System.out.println (“\nHere are the stars: )”;
for (count = 0;  count < numOfStars;  count++ ) {
            System.out.print ( “*”);
}
```

Memory has numOfStars <---->2
count <---->1

Screen has Here are the stars:
*

Example Analysis

```
int numOfStars,  count;

System.out.print (“\nEnter number of stars to display:”);
numOfStars = input.getInt();

System.out.println (“\nHere are the stars: )”;
for (count = 0;  count < numOfStars;  count++ ) {
            System.out.print ( “*”);
}
```

Memory has numOfStars <---->2
count <---->1

Screen has Here are the stars:
*

Example Analysis

```
int numOfStars,  count;

System.out.print (“\nEnter number of stars to display:”);
numOfStars = input.getInt();

System.out.println (“\nHere are the stars: )”;
for (count = 0;  count < numOfStars;  count++ ) {
        System.out.print ( “*”);
}
```

Memory has numOfStars <---->2
count <---->1

Screen has Here are the stars:
**

Example Analysis

```
int numOfStars,  count;

System.out.print (“\nEnter number of stars to display:”);
numOfStars = input.getInt();

System.out.println (“\nHere are the stars: )”;
for (count = 0;  count < numOfStars;  count++ ) {
            System.out.print ( “*”);
}
```

Memory has numOfStars <---->2
count <---->2

Screen has Here are the stars:
**

Example Analysis

```
int numOfStars,  count;

System.out.print (“\nEnter number of stars to display:”);
numOfStars = input.getInt();

System.out.println (“\nHere are the stars: )”;
for (count = 0;  count < numOfStars;  count++ ) {
            System.out.print ( “*”);
}
```

Memory has numOfStars <---->2
count <---->2

Screen has Here are the stars:
**

LCV Notes

- We started LCV count at 0, incremented by one in each iteration of loop and tested for final value count < numOfStars (so loop stopped when count became equal to numOfStars)
- In effect, count contained the number of stars currently displayed to the monitor
- DOES work for case numOfStars = = 0
- Takes practice to get the LCV values correct - just like while statement
- Can be useful for this type of example, when you know at the beginning of the loop how many times you want to execute

break statement

- Can be used in body of an if or switch statement or in the body of a loop (do; do/while; for)

- When executed, it causes control to immediately pass out of body of current statement to the next statement in the program

continue statement

- Can be used in body of a loop (do; do/while; for)

- When executed, it causes control to immediately pass out of body of loop and continue back at the top of the loop

Loop comparison

Description	While	Do-while	For

Simple **while**	while(*condition*) {
	…
	}
		if(*condition*)
		do {
		…
		} while(*condition*);
			for(;*condition*;) {
			…
			}

Simple **do-while**	while(true) {
	….
	If(!*condition*) break;
	}
		do {
		…
		} while(*condition*);
			for(;true;) {
			….
			If(!*condition*) break;
			}

Simple **for**	*initialization*
	while(*condition*) {
	…
	increment
	}
		initialization
		if(*condition*)
		do {
		…
		increment
		} while(*condition*);
			for (*initialization*;
			condition;
			increment) {
			…
			}

SUMMARY

- Three different statements to do repetition in Java
  - while statement
  - do while statement
  - for statement
- In all three cases...if there is only one statement in loop body, don't need the { } braces....conversely if braces are missing - Java assumes only one statement in loop body
- Practice, Practice, Practice