while loops
for loops
compared
and a little on
nested loops
public void fifteenStarsUpVersion() {
    int countUp;
    countUp = 0;  // # stars THAT WERE PRINTED.
    while( countUp < 15 )
    {
        System.out.print("*");
        countUp = countUp + 1;
    }
}

public void fifteenStarsDownVersion() {
    int countDown;
    countDown = 15;  // # stars TO BE PRINTED
    while( countDown > 0 )
    {
        System.out.print("*");
        countUp = countDown - 1;
    }
}
Parameter Value is from the call-ER. Its purpose is usually to customize some detail of what a method call does.

(but people might write purposeless code, and Java will happily run it.)

Parameter Variable belongs to the call-EE (method body)'s code. Its initial value is copy of the param. value (FROM THE call-ER)

Its purpose is usually to customize some detail of what a method call does.

(but..Java will happily run purposeless code.)
public void paramStarsUpVersion(int ParVar)
{
    int countUp;
    countUp = 0;   // # stars THAT WERE PRINTED.
    while( countUp < ParVar )
    {
        System.out.print("*");
        countUp = countUp + 1;
    }
}

public void paramStarsDownVersion(int ParVar)
{
    int countDown;
    countDown = ParVar;   // # stars TO BE PRINTED
    while( countDown > 0 )
    {
        System.out.print("*");
        countDown = countDown - 1;
    }
}
public class PrintDemo
{
    public static void main(String[] a)
    {
        Printer pR = new Printer();
        pR.paramStarsUpVersion(13);
    }
}

We have lots of examples of methods like those above being defined in a class like the Printer class. One Printer object is like Benjamin Franklin, an entity that is capable of printing (he started his career as a printer in Philadelphia)

Here is an example of a caller of the paramStarsUpVersion method and how it specifies a parameter value of 13.

    public class PrintDemo
    {
        public static void main(String[] a)
        {
            Printer pR = new Printer();
            pR.paramStarsUpVersion(13);
        }
    }
Please FORGET ABOUT PARAMETERS FOR A WHILE

Software Engineering Principle: **Separation of Concerns.** You sometimes narrow and focus your attention to one new or difficult problem, and temporarily (but on purpose!) forget about others; avoid mental confusion.

- How parameter values get copied to parameter variables, usually for the purpose of varying a detail of what a method call does.
- How Java's (and other lang's) loop control statements work, so you will be able to use and debug them when you create software.
public void fifteenStarsUpVersion() {
    int countUp;
    countUp = 0;  // # stars THAT WERE PRINTED.
    while( countUp < 15 )
    {
        System.out.print("*");
        countUp = countUp + 1;
    }
}

public void fifteenStarsDownVersion() {
    int countDown;
    countDown = 15;  // # stars TO BE PRINTED
    while( countDown > 0 )
    {
        System.out.print("*");
        countDown = countDown - 1;
    }
}
How to convert while a loop into a for loop.

```
while(
    
) {
    INIT. ;
    TEST
    STATEMENTS;
    TO;
    REPEAT;
    UPDATE

    }
```

```
for(
    INIT. ;
    TEST
    STATEMENTS;
    TO;
    REPEAT;
    UPDATE

) {
    
    }
```
```java
public void fifteenStarsWhileUpVersion() {
    int countUp;
    countUp = 0;  // # stars THAT WERE PRINTED.
    while (countUp < 15 )
    {
        System.out.print("*”);
        countUp = countUp + 1;
    }
}

public void fifteenStarsForUpVersion() {
    //countDown # stars THAT WERE PRINTED
    for( int countDown = 0; countDown < 15; countDown = countDown + 1 )
    {
        System.out.print("*”);
    }
}
```
public void fifteenStarsWhileUpVersion()
{
    int countUp;
    countUp = 0; // # stars THAT WERE PRINTED.
    while(countUp < 15)
    {
        System.out.print("*");
        countUp = countUp + 1;
    }
}

public void fifteenStarsForUpVersion()
{
    //countDown # stars THAT WERE PRINTED
    for(int countDown = 0; countDown < 15; countDown = countDown + 1)
    {
        System.out.print("*");
    }
}
How to convert a for loop into a while loop.

```plaintext
while( TEST ) for( INIT.; TEST; UPDATE ) {
    statements; to; repeat;
    update;
} {
    statements; to; repeat;
} {
    statements; to; repeat;
}
```
A real computer science idea
(borrowed from other subjects of course)

Reducibility

A for loop can be REDUCED TO a while loop.
You can get away with while loops ONLY.
Every task a for loop might do can just as well be done by a while loop.
while loops have AT LEAST AS MUCH (and maybe more) “COMPUTATIONAL POWER” as for loops.
Inter-reducibility is a kind of equivalence

Everything a for loop can do SO CAN a while loop do.

So whiles have equal or greater power than forss.

Everything a while loop can do SO CAN a for loop do.

So forss have equal or greater power than whiles.

This all implies: while loops and for loops can do EXACTLY THE SAME computational tasks.
Which is better? Which do you prefer?

- while (clicker A) ??
- for (clicker B) ??

Opinion: The best choice is the one that expresses most clearly to OTHER PEOPLE what you have programmed the computer to do.

Sometimes the for loop is more clear, other times the while loop is more clear.

The computer will do the same things either way.
1. (+) The parts are done in the order they are written.
2. (+) Easier to get the logic right.
3. (+) The INIT. stuff can be several statements.
4. (-) The loop is not expressed as a self-contained thing.

1. (-) It's hard for beginners to analyze the sequence exactly right.
2. (+) The 3 parts (usually) needed for loop control are in the same line.
3. (+) The 3 part for( ; ; ) reminds us to code: (1) INITIALIZATION (2) TEST (3) UPDATE (increment or decrement.)
4. (+) Declarations in INIT. are local to the loop.
public void fifteenStarsWhileDownVersion() {
    int countDown;
    countDown = 15;  // #stars TO BE PRINTED
    while( countDown > 0 )
    {
        System.out.print("*");
        countUp = countDown - 1;
    }
}

public void fifteenStarsForDownVersion() {
    int countDown;
    countDown = 15;  // #stars TO BE PRINTED
    for( int countDown = 15; countDown > 0; countDown = countDown - 1 )
    {
        System.out.print("*");
    }
}
public void fifteenStarsWhileDownVersion( )
{
    int countDown;
    countDown = 15;  // #stars TO BE PRINTED
    while( countDown > 0 )
    {
        System.out.print("*");
        countUp = countDown - 1;
    }
}

public void fifteenStarsForDownVersion( )
{
    for( int countDown = 15; countDown > 0;
        countDown = countDown - 1 )
    {
        System.out.print("*");
    }
}
A nested loop is a loop, called the outer loop, whose statements to repeat comprise a loop, called the inner loop.

The group of statements to repeat IS a loop...the group is a loop INSIDE a loop!
while( TEST ) {
    INIT. ;
    while( TEST ) {
        STATEMENTS; TO; REPEAT;
        UPDATE ;
    }
    UPDATE ;
}
When the inner loop is a for loop, you can easily see the inner loop as a SELF-CONTAINED thing.
Thinking about nested loops.

• FIRST, think of what the whole inner loop does, in a single thought. For example, “paint a row of Pixels.”

• SECOND, think that the outer loop repeats the single thing the inner loop does: Paint many rows of pixels to fill in a rectangle.

• FIRST, think of how many repetitions the outer loop makes, and what they are for. For example: “repeat once for each row”

• SECOND, think of what the whole inner loop does in a single thought.
public void copyInSafely( int xOrig, int yOrig, Picture source )
{
    for(int xS = 0; xS < source.getWidth(); xS=xS+1 )
    {
        for(int yS = 0; yS < source.getHeight(); yS=yS+1 )
        {
            if( (xS+xOrig < this.getWidth())
                &&
                (yS+yOrig < this.getHeight())
            )
            {
                this.getPixel(xS+xOrig,yS+yOrig)
                    .setColor(
                        source.getPixel(xS,yS).getColor()
                    );
            }
        }
    }
}