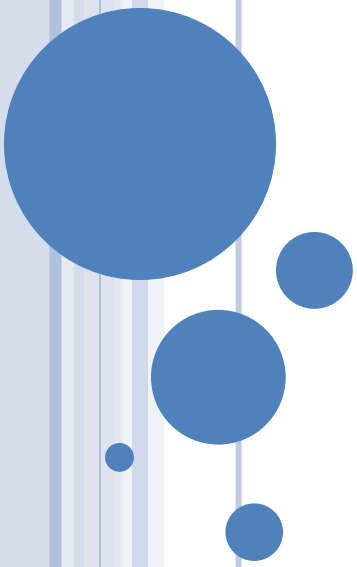


Comp 110
Loops Advanced

Instructor: Prasun Dewan

Prerequisite

- Loops

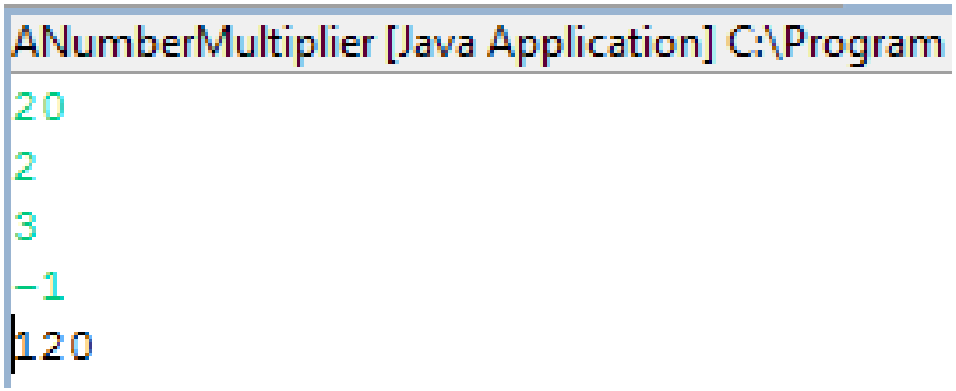
.

Loops

- More loops
- Off-by-one errors
- Infinite loops
- Nested Loops
- Animations
- Concurrency
- Synchronized methods
- Property changes

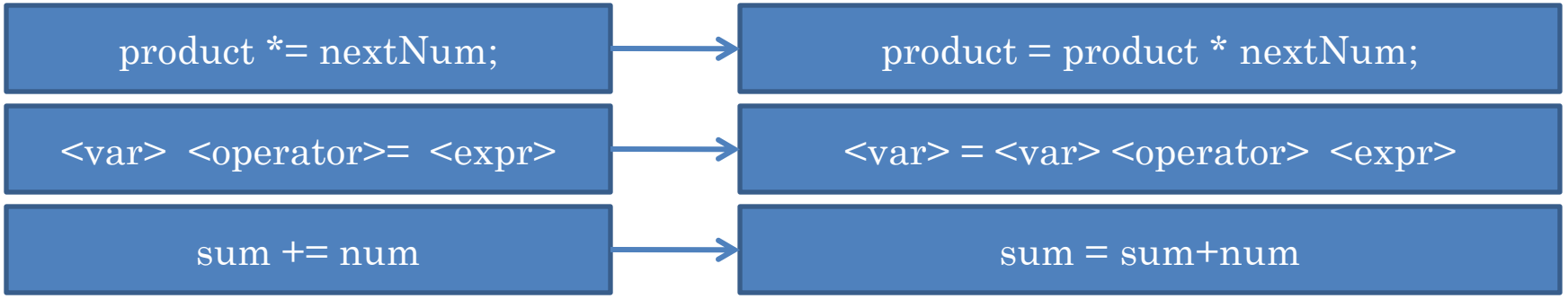
Multiplying Numbers

```
int product = 1;
int nextNum = Console.readInt();
while (nextNum >= 0) {
        product = product* nextNum;
        nextNum = Console.readInt();
}
print (product);
```



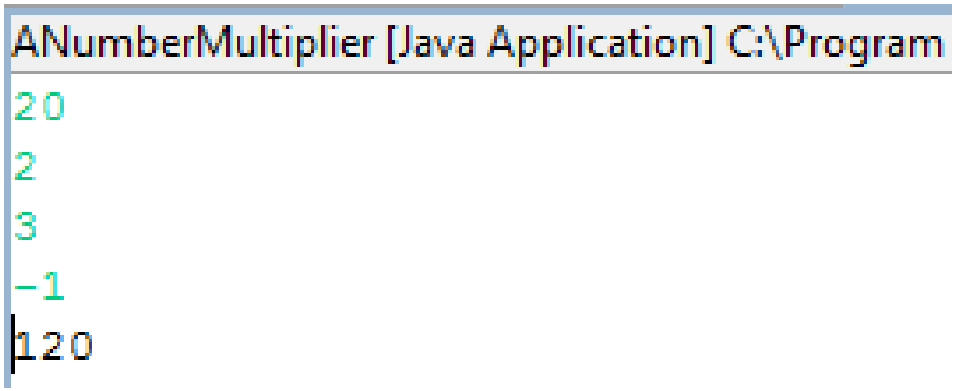
Cumulative Assignment

```
int product = 1;
int nextNum = Console.readInt();
while (nextNum >= 0) {
        product *= nextNum;
        nextNum = Console.readInt();
}
print (product);
```



Multiplying Positive Numbers

```
int product = 1;
int nextNum = Console.readInt();
while (nextNum >= 0) {
        product *= nextNum;
        nextNum = Console.readInt();
}
print(product);
```



Multiplying N Numbers (Edit)

```
<terminated> AnNNumberMultiplier [Java Ap
List Length?
3
Next Number?
20
Next Number?
-3
Next Number?
2
List Product:
-120
```

```
int product = 1;
int nextNum = Console.readInt();
while (nextNum >= 0) {
        product *= nextNum;
        nextNum = Console.readInt();
}
print(product);
```

Multiplying N Numbers

```
<terminated> AnNNumberMultiplier [Java Ap
List Length?
3
Next Number?
20
Next Number?
-3
Next Number?
2
List Product:
-120
```

```
int product = 1;
int nextNum = Console.readInt();
while (nextNum >= 0) {
        product *= nextNum;
        nextNum = Console.readInt();
}
print (product);
```

```
int listLength = readListLength();
int counter = 0;
int nextNum;
int product = 1;
while (counter <  listLength) {
        counter += 1;
        nextNum = readNextNumber();
        product *= nextNum;
}
print (product);
```

Multiplying First N Numbers: N! (Edit)

1*2*3*4*…*n

Factorial(n) = n!

Multiplying First N Numbers: N!

```
int product = 1;
int n = 2;
int counter = 0;
while (counter < n) {
        product *= counter;
        counter += 1;
}
System.out.println (product);
```

1*0*1

1*2*3*4*…*n

Multiplying First N Numbers: N!

```
int product = 1;
int n = ???;
int counter = 0;
while (counter < n) {
        product *= counter;
        counter += 1;
}
System.out.println (product);
```

1*0*1*2*3*4*…*n-1

1*2*3*4*…*n

Off by one

Multiplying First N Numbers: N!

```
int product = 1;
int n = ???;
int counter = 1;
while (counter < n) {
        product *= counter;
        counter += 1;
}
System.out.println (product);
```

1*1*2*3*4*…*n-1

1*2*3*4*…*n

Off by one

Multiplying First N Numbers: N!

```
int product = 1;
int n = ???;
int counter = 1;
while (counter <= n) {
        product *= counter;
        counter += 1;
}
System.out.println (product);
```

1*1*2*3*4*…*n-1*n

1*2*3*4*…*n

Better Name

```
int product = 1;
int n = ???;
int nextMultiplier = 1;
while (nextMultiplier <= n) {
        product *= nextMultiplier;
        nextMultiplier += 1;
}
System.out.println (product);
```

1*1*2*3*4*…*n-1*n

1*2*3*4*…*n

Better Name

```
int product = 1;
int n = ???;
int nextMultiplier = 0;
while (nextMultiplier <= n) {
        product *= nextMultiplier;
        nextMultiplier += 1;
}
System.out.println (product);
```

1*1*2*3*4*…*n-1*n

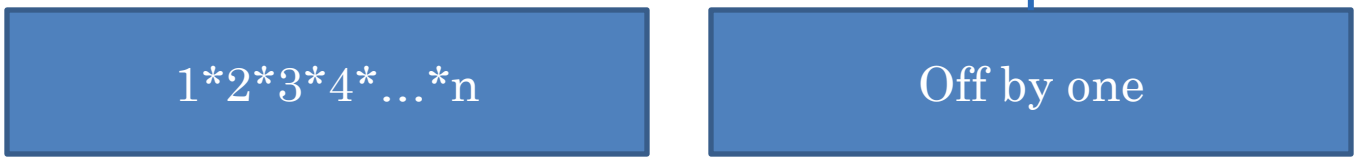
1*2*3*4*…*n

Easier to spot off-by-one errors

Incrementing Counter Before Operation

```
int product = 1;
int n = ???;
int prevMultiplier = 0;
while (prevMultiplier <= n) {
        prevMultiplier += 1;
        product *= prevMultiplier;
}
System.out.println (product);
```

1*1*2*3*4*…*n-1*n*n+1



Incrementing Counter Before Operation

```
int product = 1;
int n = ???;
int prevMultiplier = 0;
while (prevMultiplier < n) {
        prevMultiplier += 1;
        product *= prevMultiplier;
}
System.out.println (product);
```

1*1*2*3*4*…*n-1*n

1*2*3*4*…*n

Checking of Non Equality

```
int product = 1;
int n = ???;
int prevMultiplier = 0;
while (prevMultiplier != n) {
        prevMultiplier += 1;
        product *= prevMultiplier;
}
System.out.println (product);
```

1*2*3*4*…*n

Checking of Non Equality

```
int product = 1;
int n = -5;
int prevMultiplier = 0;
while (prevMultiplier != n) {
        prevMultiplier += 1;
        product *= prevMultiplier;
}
System.out.println (product);
```

1*2*3*4*…*n

Checking of Non Equality

```
int product = 1;
int n = -5;
int prevMultiplier = 0;
while (prevMultiplier != n) {
        prevMultiplier += 1;
        product *= prevMultiplier;
}
System.out.println (product);
```

-5*-4*-3*-2*-1*0*1*2…

1*2*3*4*…*n

Infinite loop

Counter Not Changed

```
int product = 1;
int n = ???;
int prevMultiplier = 0;
while (prevMultiplier < n) {

        product *= prevMultiplier;
}
System.out.println (product);
```

1*0*0*0*…

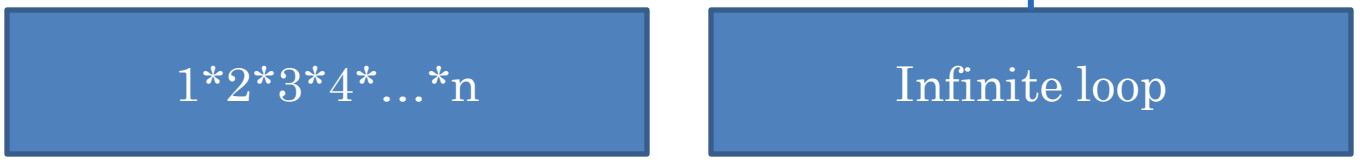
1*2*3*4*…*n

Infinite loop

Counter Changed in the Wrong Direction

```
int product = 1;
int n = ???;
int prevMultiplier = 0;
while (prevMultiplier < n) {
        prevMultiplier -= 1;
        product *= prevMultiplier;
}
System.out.println (product);
```

1*0*-1*-2*…



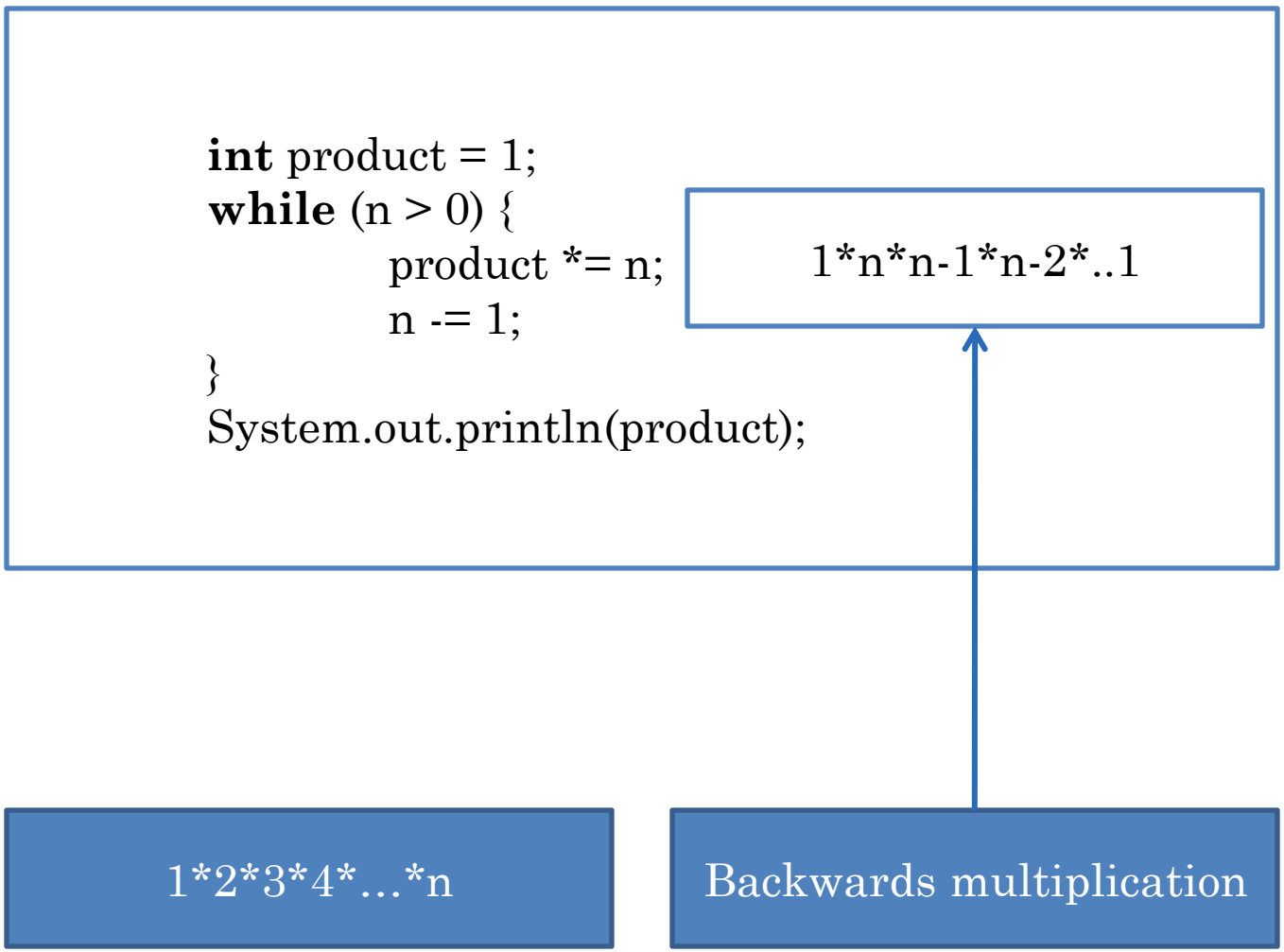
Guarding Against Infinite Loops

- Update variable(s) in loop expression
- Expression must converge to false

Decrementing Solution

```
int product = 1;
while (n > 0) {
        product *= n;
        n -= 1;
}
System.out.println(product);
```

1*n*n-1*n-2*..1



1*2*3*4*…*n

Backwards multiplication

Decrementing Solution

```
int product = 1;
while (n > 0) {
        product *= n;
        n--;
}
System.out.println(product);
```

n-- → n = n - 1

n++ → n = n + 1

Counter-Controlled vs. Event-Controlled

```
int product = 1;
int nextNum = Console.readInt();
while (nextNum >= 0) {
        product *= nextNum;
        nextNum = Console.readInt();
}
print (product);
```

Event-controlled

```
int product = 1;
int n = readNumElements();
int counter = 0;
while (counter < n) {
        int nextNum = readNum();
        product *= nextNum;
        counter += 1;
}
```

Counter-controlled

Counter-Controlled vs. Event-Controlled

- Number of loop iterations (executions of loop body) known before loop executed
  - initialize counter to some value
  - increment/decrement counter by fixed step beginning/end of body
  - exit when counter reaches limit
- Limit not known before loop starts
  - Test one more events (e.g. reading of input) occurring while loop executes
  - Terminate when events make loop condition false

Counter-Controlled vs. Event-Controlled

Counting until 10 in Hide & Seek	Counter-controlled
Searching for others in Hide & Seek	Event-controlled
Grading Comp14 exams	Counter-controlled
Fishing for the right answer from students	Event-controlled & counter-controlled
Counting the days until it is vacation	Counter-controlled
Counting # of candies in a jar	Event-controlled

Factorial List (Edit)

```
<terminated> AFactorialL
3
factorial = 6
2
factorial = 2
-1
```

```
int n = ???;
int product = 1;
while (n > 0) {
        product *= n;
        n -= 1;
}
return product;
```

Factorial List

```
<terminated> AFactorialL
3
factorial = 6
2
factorial = 2
-1
```

```
public static int factorial (int n) {
        int product = 1;
        while (n > 0) {
                product *= n;
                n -= 1;
        }
        return product;
}
```

```
public static void main (String[] args) {
        int newVal = Console.readInt();
        while (newVal >= 0) {
                System.out.println("factorial =" + factorial(newVal));
                newVal = Console.readInt();
        }
}
```

Removing Code Duplication (Edit)

```
<terminated> AFactorialL
3
factorial = 6
2
factorial = 2
-1
```

```
public static int factorial (int n) {
        int product = 1;
        while (n > 0) {
                product *= n;
                n -= 1;
        }
        return product;
}
```

```
public static void main (String[] args) {
        int newVal = Console.readInt();
        while (newVal >= 0) {
                System.out.println("factorial =" + factorial(newVal));
                newVal = Console.readInt();
        }
}
```

Break Statement

```
<terminated> AFactorialL
3
factorial = 6
2
factorial = 2
-1
```

```
public static int factorial (int n) {
        int product = 1;
        while (n > 0) {
                product *= n;
                n -= 1;
        }
        return product;
}
```

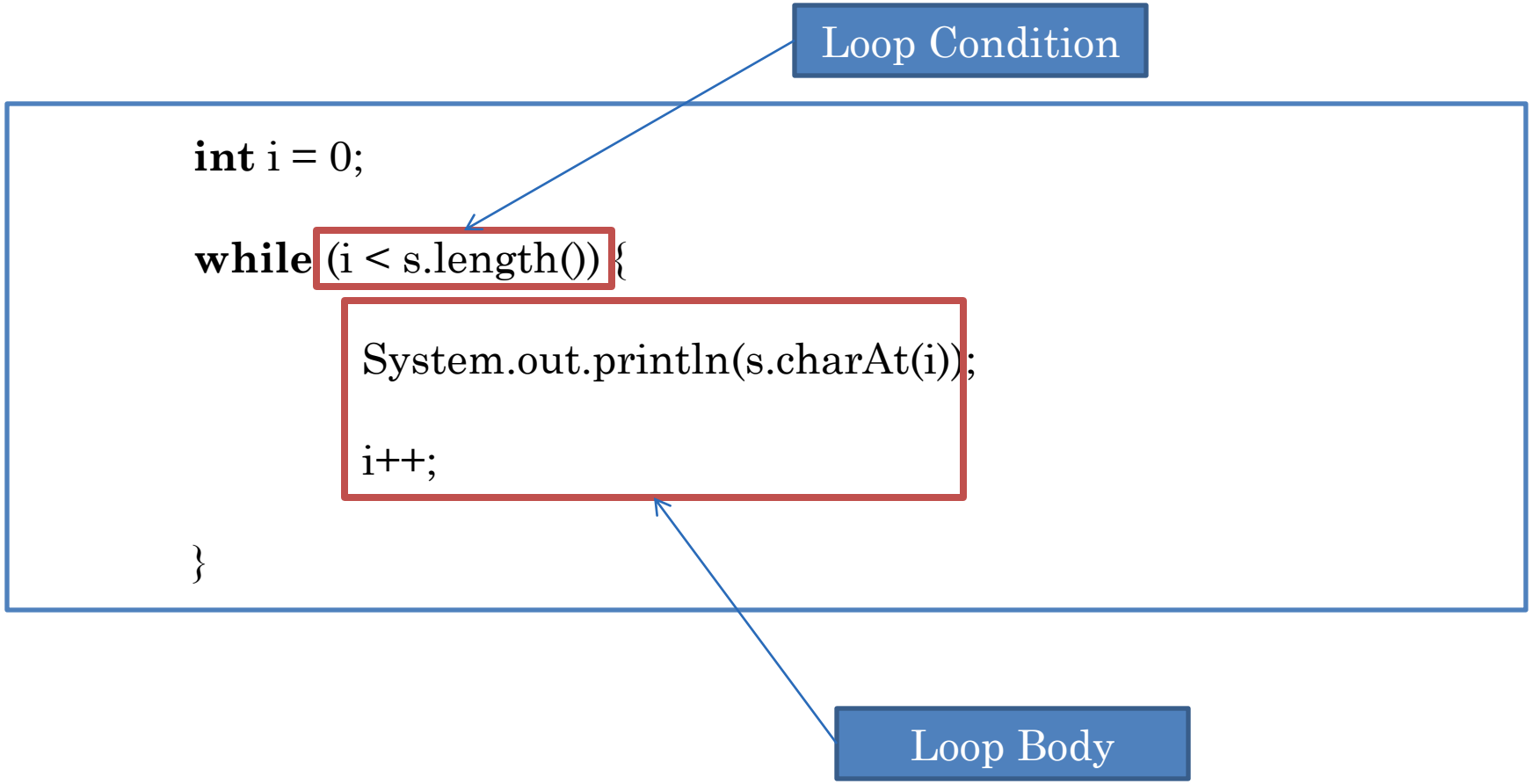
```
public static void main (String[] args) {
        while (true) { // loop condition never false
                int newVal = Console.readInt();
                if (newVal < 0) {
                        break;
                }
                System.out.println("factorial =" + factorial(newVal);
        }
}
```

String Processing

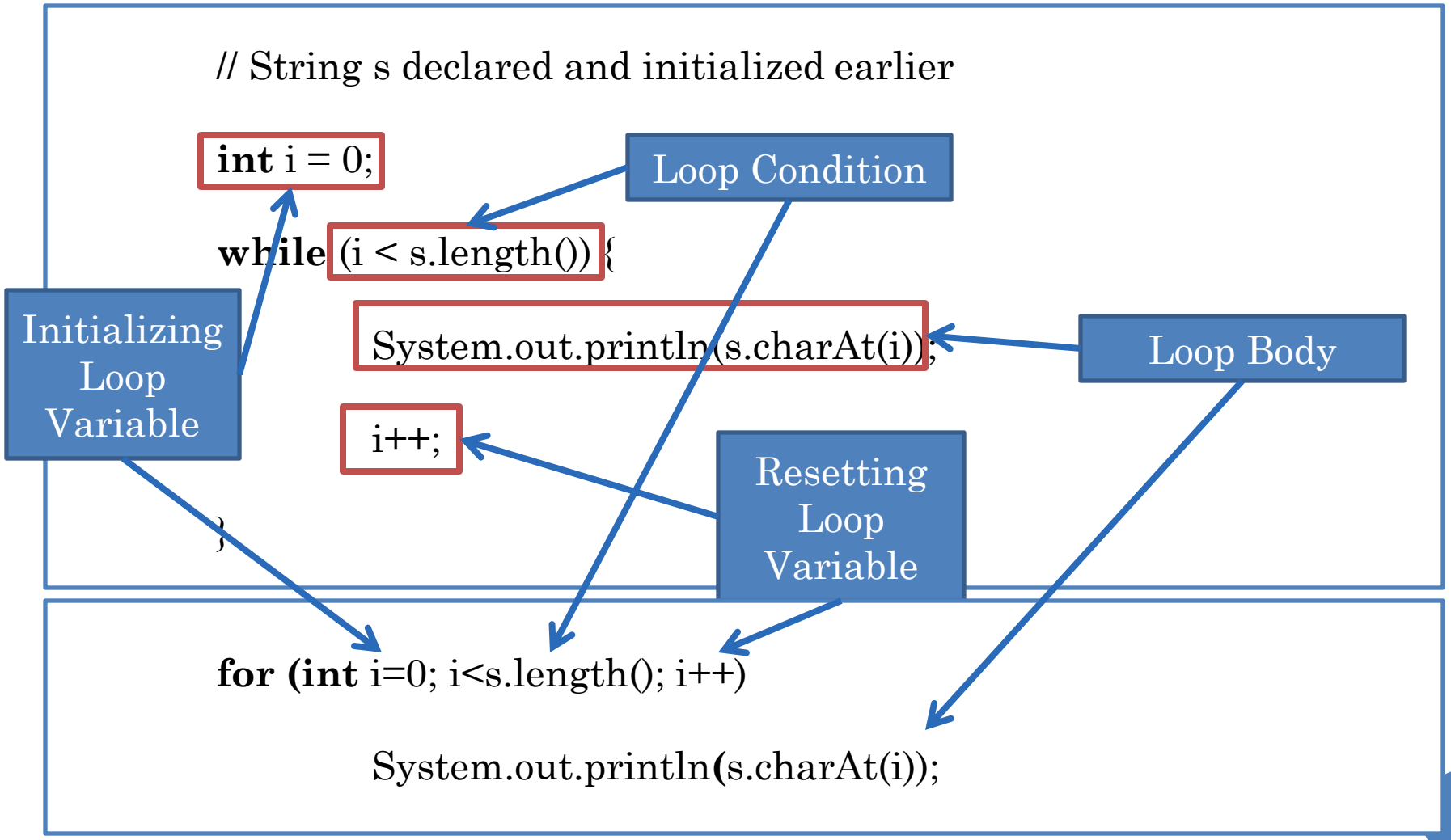
```
int i = 0;

while (i < s.length()) {

        System.out.println(s.charAt(i));

        i++;

}
```

Prints each character on a separate line

Dissecting a Loop



Finger-grained Dissection



Meaning of For Loop

```
for (S1; E; S2)
        S3
```

```
S1;
while (E) {
        S3;
        S2;
}
```

```
for (; E; S2)
        S3
```

```
while (E) {
        S3;
        S2;
}
```

```
for (; E;)
        S3
```

```
while (E) {
        S3;
}
```

```
for (;;)
        S3
```

```
S1;
while (true) {
        S3;
        S2;
}
```