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

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

product *= nextNum;
<var> <operator> = <expr>
sum += num

product = product * nextNum;
<var> = <var> <operator> <expr>
sum = sum+num
MULTIPLYING POSITIVE NUMBERS

```java
int product = 1;
int nextNum = Console.readInt();
while (nextNum >= 0) {
    product *= nextNum;
    nextNum = Console.readInt();
}
print(product);
```
MULTIPLYING N NUMBERS (EDIT)

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

```java
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!

```java
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!

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

1*2*3*4*...*n

Off by one

1*0*1*2*3*4*...*n-1
MULTIPLYING FIRST N NUMBERS: N!

```java
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!**

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

$1 \times 2 \times 3 \times 4 \times \ldots \times n-1 \times n$
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
```java
int product = 1;
int n = ???;
int nextMultiplier = 0;
while (nextMultiplier <= n) {
    product *= nextMultiplier;
    nextMultiplier += 1;
}
System.out.println(product);
```

Better Name

Easier to spot off-by-one errors

1*1*2*3*4*...*n

1*2*3*4*...*n

1*1*2*3*4*...*n-1*n
Incrementing Counter Before Operation

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

1*2*3*4*...*n-1*n*n+1

Off by one
INCREMENTING COUNTER BEFORE OPERATION

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

1*2*3*4*...*n-1*n
```java
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

```java
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
```java
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

-5*-4*-3*-2*-1*0*1*2...

Infinite loop
int product = 1;
int n = ???;
int prevMultiplier = 0;
while (prevMultiplier < n) {
    product *= prevMultiplier;
}
System.out.println (product);
**Counter Changed in the Wrong Direction**

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

1*0*-1*-2*…

1*2*3*4*…*n

Infinite loop
GUARDING AGAINST INFINITE LOOPS

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

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

Backwards multiplication

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

1\*2\*3\*4\*...\*n
Decrementing Solution

```java
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

**Event-controlled**

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

**Counter-controlled**

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

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

<table>
<thead>
<tr>
<th>&lt;terminated&gt;</th>
<th>AFactorialL</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>factorial = 6</td>
</tr>
<tr>
<td>2</td>
<td>factorial = 2</td>
</tr>
<tr>
<td>-1</td>
<td></td>
</tr>
</tbody>
</table>
**Factorial List**

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

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

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

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

```java
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

```java
int i = 0;

while (i < s.length()) {
    System.out.println(s.charAt(i));
    i++;
}
```

Prints each character on a separate line
int i = 0;

while (i < s.length()) {
    System.out.println(s.charAt(i));
    i++;
}

Loop Condition

Loop Body

DISSECTING A LOOP
// String s declared and initialized earlier

```java
int i = 0;
while (i < s.length()) {
    System.out.println(s.charAt(i));
    i++;
}
```

---

```
for (int i=0; i<s.length(); i++)
    System.out.println(s.charAt(i));
```
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

for (;;)
   S3

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