COMPSCI 121: LOOPS

SPRING 20

GOALS FOR TODAY'S CLASS

More nuts & bolts of programming!

- **Looping**
  - While loop.
  - For loop
  - Do While loop.
- **Flow Charts.**
- **Writing pseudocode.**

THINK - PAIR - SHARE

```
public static void countdown(int n) {
  while (n > 0) {
    System.out.println(n);
    n = n - 1;
  }
  System.out.println("Blastoff!");
}
```

What does this code do?

Read the code. Think and then write (in your own words) what you think this code does.

When you finish, pair up and discuss your answers.

Finally, let's share.

WHILE LOOP - See `LoopDemo.java`

```
public static void countdown(int n) {
   while (n > 0) {
      System.out.println(n);
      n = n - 1;
   }
   System.out.println("Blastoff!");
}
```

condition is TRUE

update variable

Scope of while loop { }

This is pseudocode

```
While n is greater than zero {
     print the value of n and
     then reduce the value of n by 1.
}
When n gets to zero, print "Blastoff!"
```

Each time through a loop's statements is called an *iteration*.

CLICKER QUESTION #1

```
int x = 10;
while( x < 15 ) {
        System.out.println(x +" " );
}
```

A. prints 10
B. prints 10 11 12 13 14
C. prints 10 11 12 13 14 15
D. Prints 10 endlessly
E. Does not execute

What does this loop do when executed?

READY FOR THE ANSWER?

CLICKER QUESTION #1 ANSWER

```
int x = 10;
while( x < 15 ) {
        System.out.print(x + "  " );
    }
```

A. prints `10` wrong, as loop runs more than once
B. prints `10 11 12 13 14` wrong output
C. prints `10 11 12 13 14 15` wrong output
D. `Prints 10 endlessly` variable not updated
   infinite loop
E. `Does not execute` executes (but *infinite loop*)

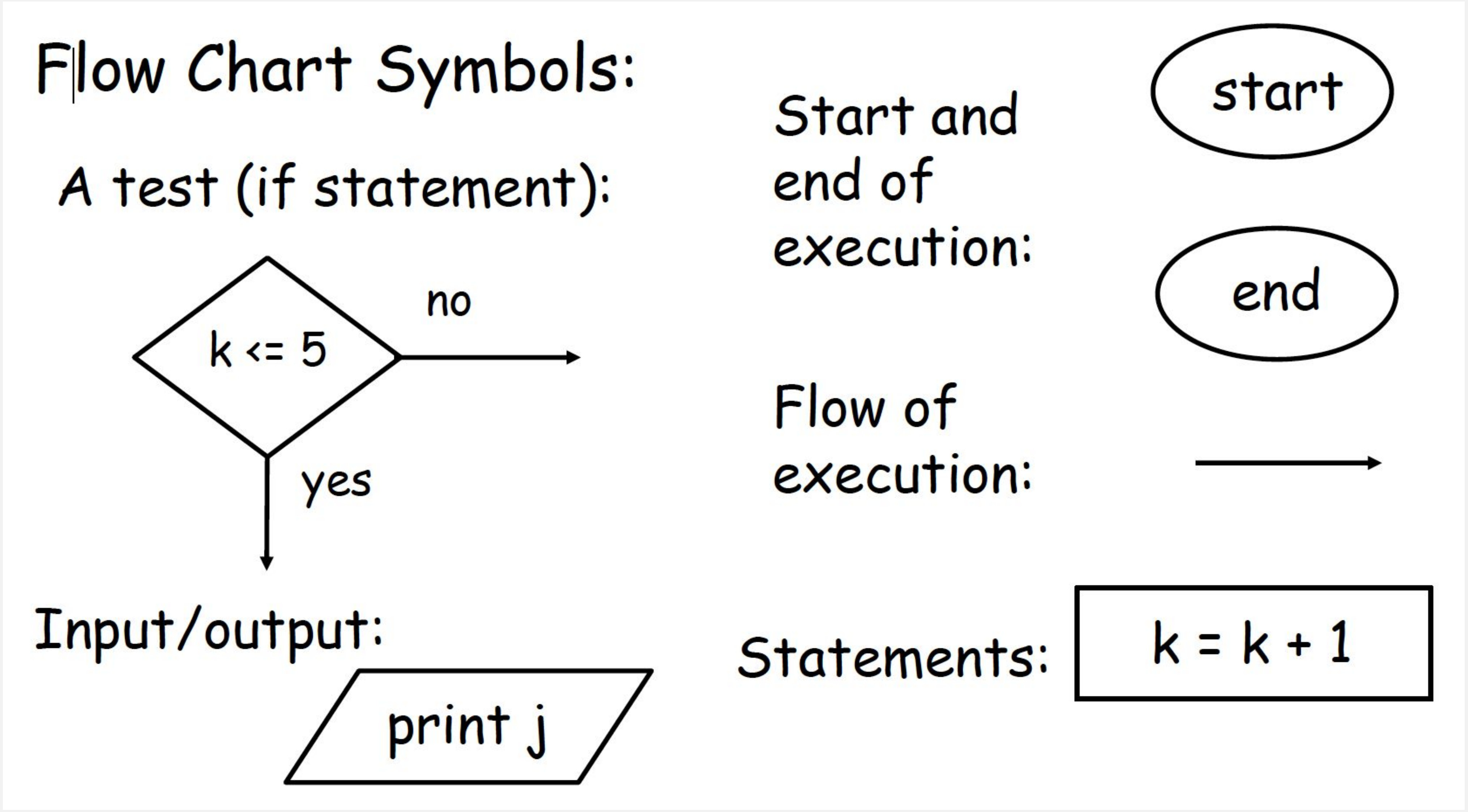
FLOW CHARTS

- Shows the Flow of Control through a program.

- Can be used to describe any algorithm - not just code.

- A useful visual design technique.

REVIEW: FLOW CHART SYMBOLS



Flow Chart Symbols:

A test (if statement):

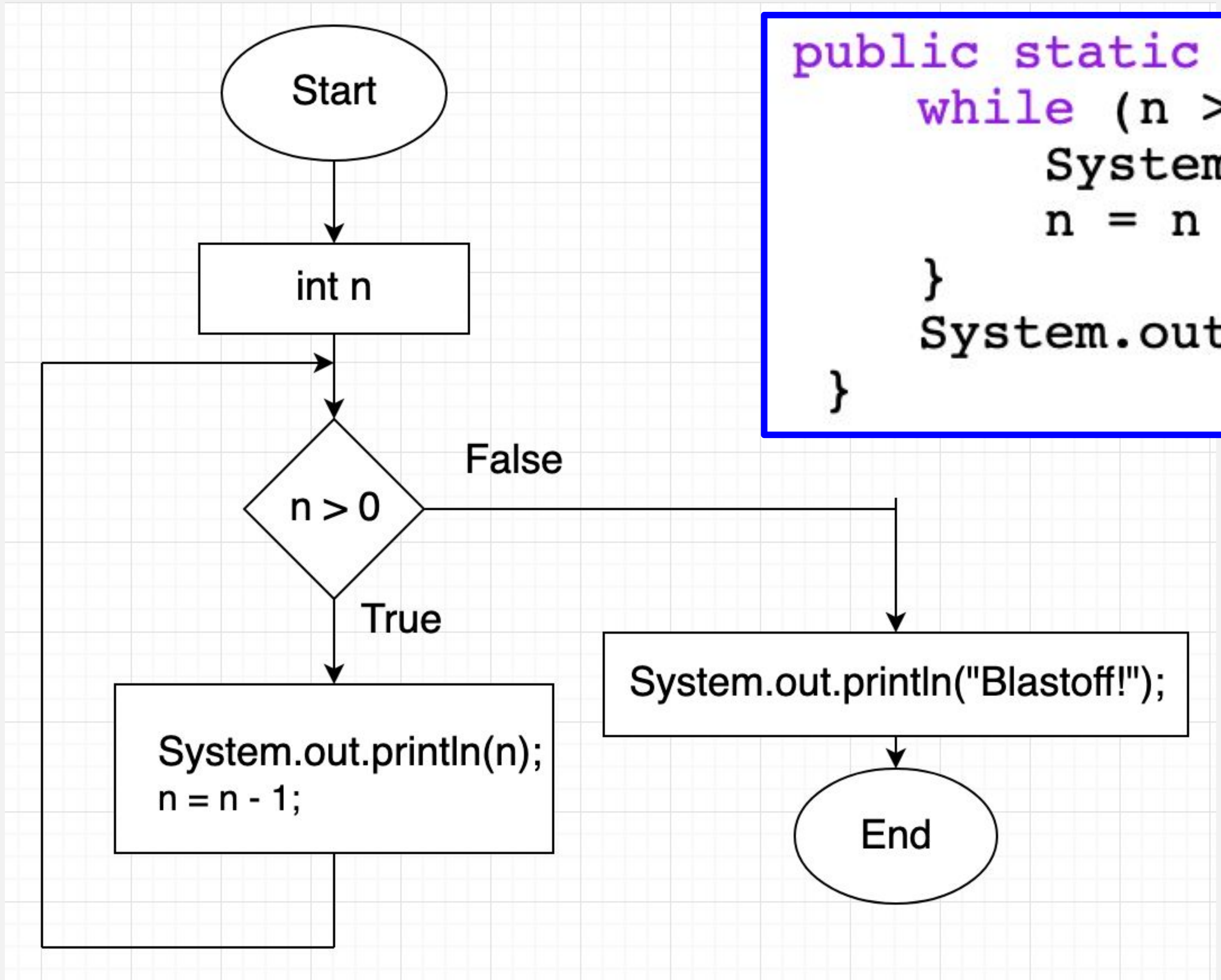
Start and end of execution:

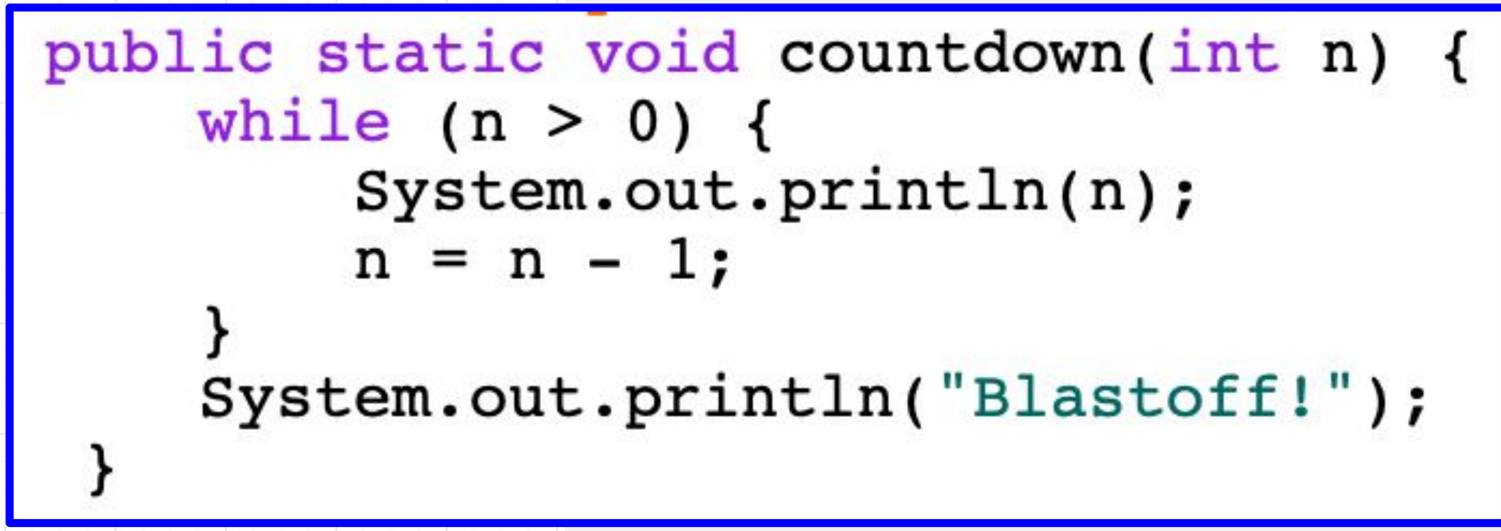
Flow of execution:

Input/output:

Statements:

FLOW CHART EXAMPLE 1





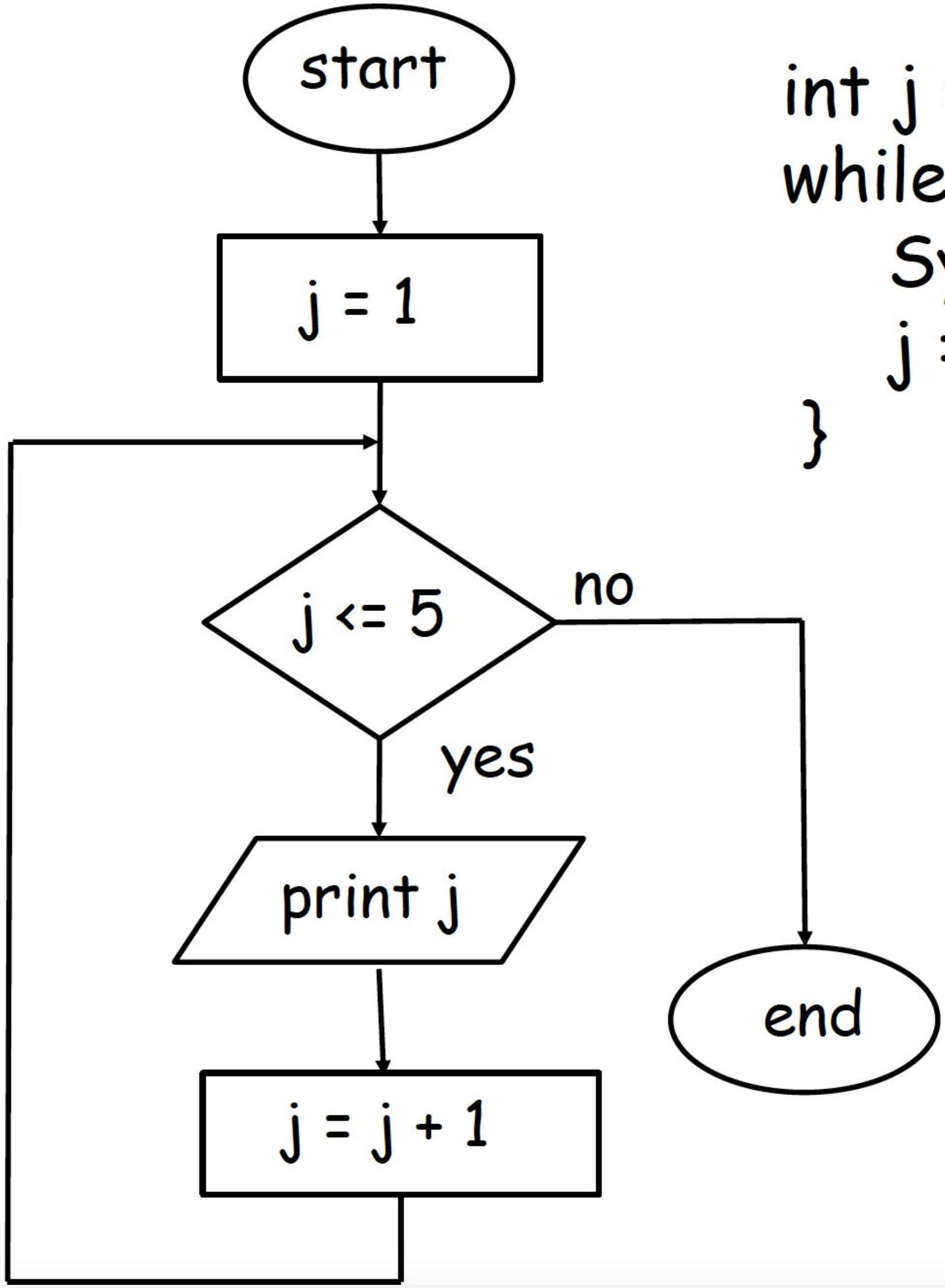
```
public static void countdown(int n) {
    while (n > 0) {
        System.out.println(n);
        n = n - 1;
    }
    System.out.println("Blastoff!");
}
```

You can draw on paper or use https://www.draw.io/

EXAMPLE 2

Output

```
1
2
3
4
5
```

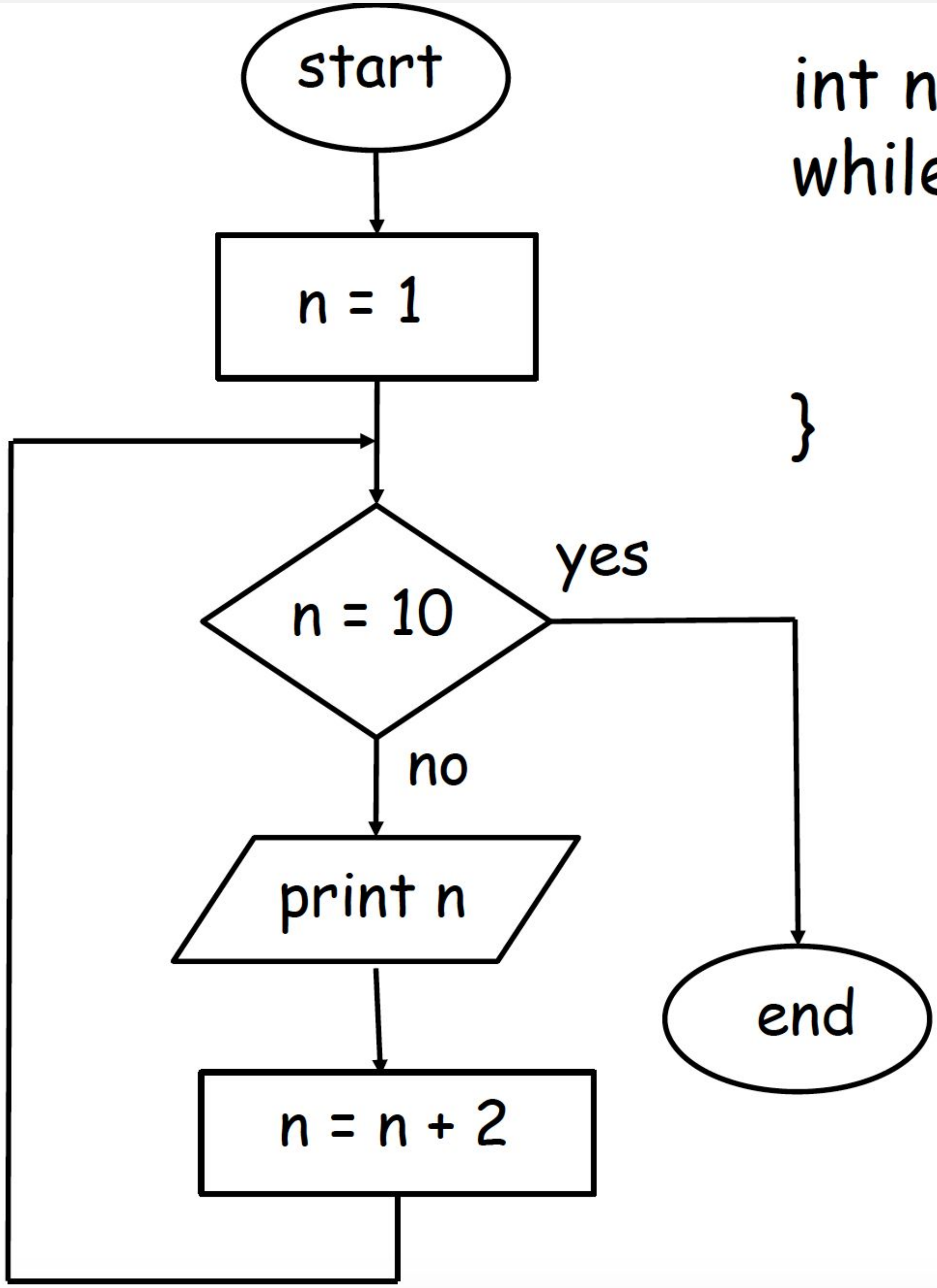


```
int j = 1;
while (j <= 5)  {
        System.out.println(j);
        j = j + 1;
 }
```

Flow chart for this while loop Code.

EXAMPLE 3

T-P-S
What's happening here?



```
int n = 1;
while (n != 10){
        System.out.println(n);
        n = n + 2;
}
```

Does the logic of the diagram agree with the code?

's

CLICKER QUESTION #2

```
while (n != 1) {
   System.out.print(n + " ");
   if (n % 2 == 0) {
      n = n / 2;
   } else {
       n = n * 3 + 1;
   }
}
```

What is the the output when `n = 5`?

A. 2
B. 5 16 8 4 2 -1
C. 5 16 8 4 2 1
D. 5 16 8 4 2
E. 0

READY FOR THE ANSWER?

CLICKER QUESTION #2 ANSWER

```
while (n != 1) {
   System.out.print(n + " ");
    if (n % 2 == 0) {
        n = n / 2;
    } else {
         n = n * 3 + 1;
 }}
```

What is the the output when `n = 5`?

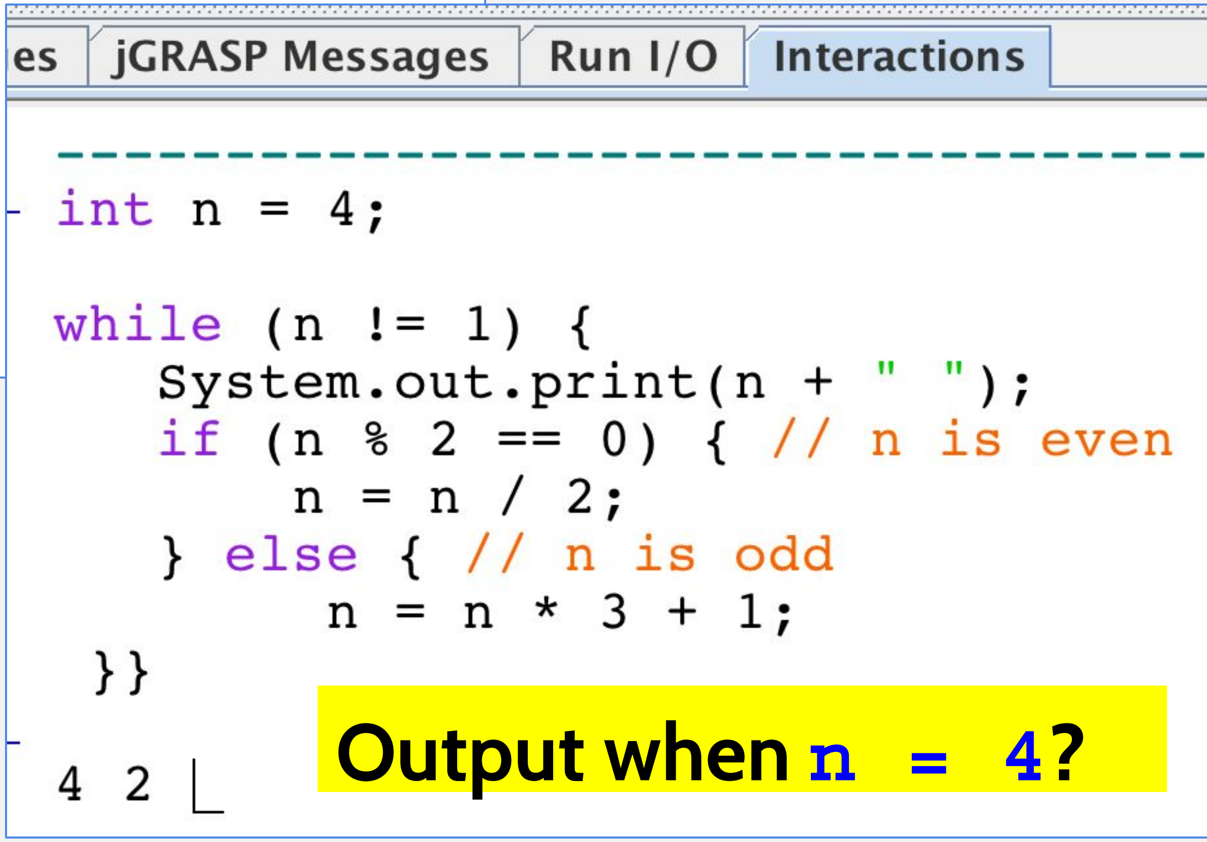
A. 2 wrong starting value

B. 5 16 8 4 2 -1 loop stops at 2

C. 5 16 8 4 2 1 loop stops at 2

D. 5 16 8 4 2

E. 0 wrong starting value



```
int n = 4;

while (n != 1) {
    System.out.print(n + " ");
    if (n % 2 == 0) { // n is even
        n = n / 2;
    } else { // n is odd
         n = n * 3 + 1;
 }}

4 2
```

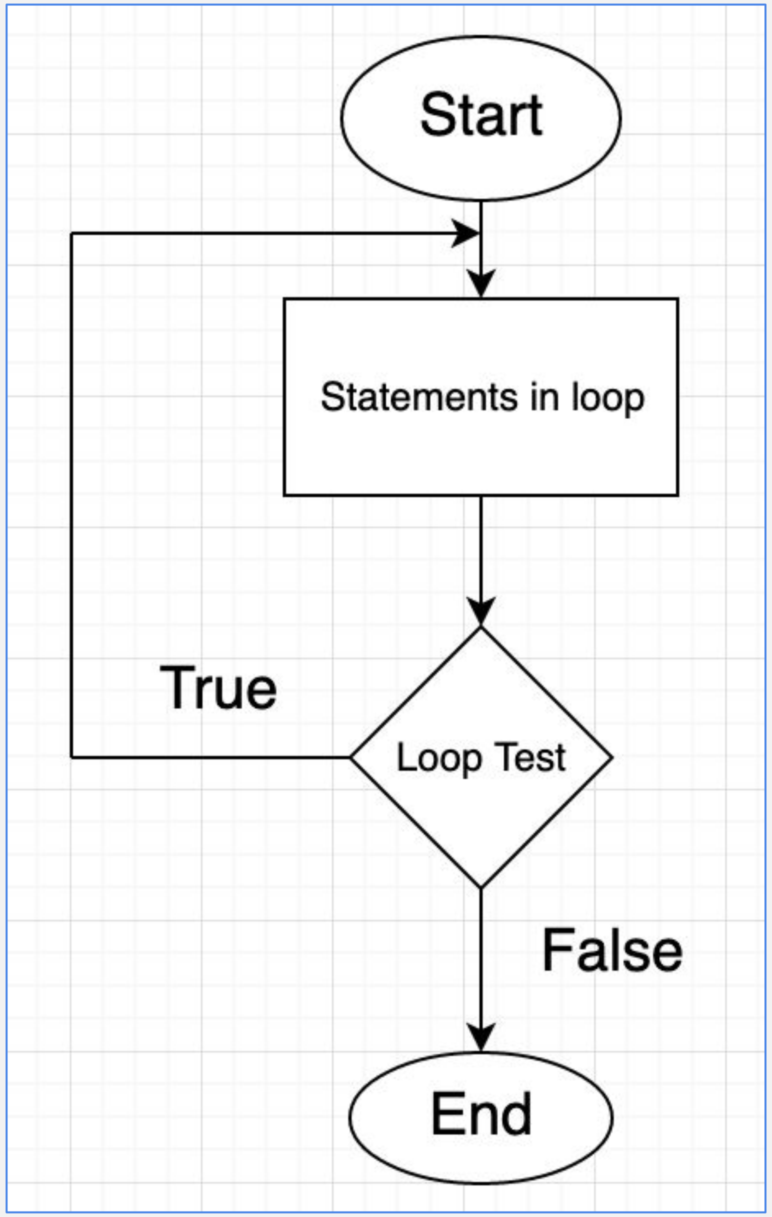
Output when `n = 4`?

DO-WHILE LOOP

- `While` loops execute the loop test first, then body.
- `Do While` loop executes the body first then the test.

General form:

```
do {
    // statements in body
} while (loop test);
```

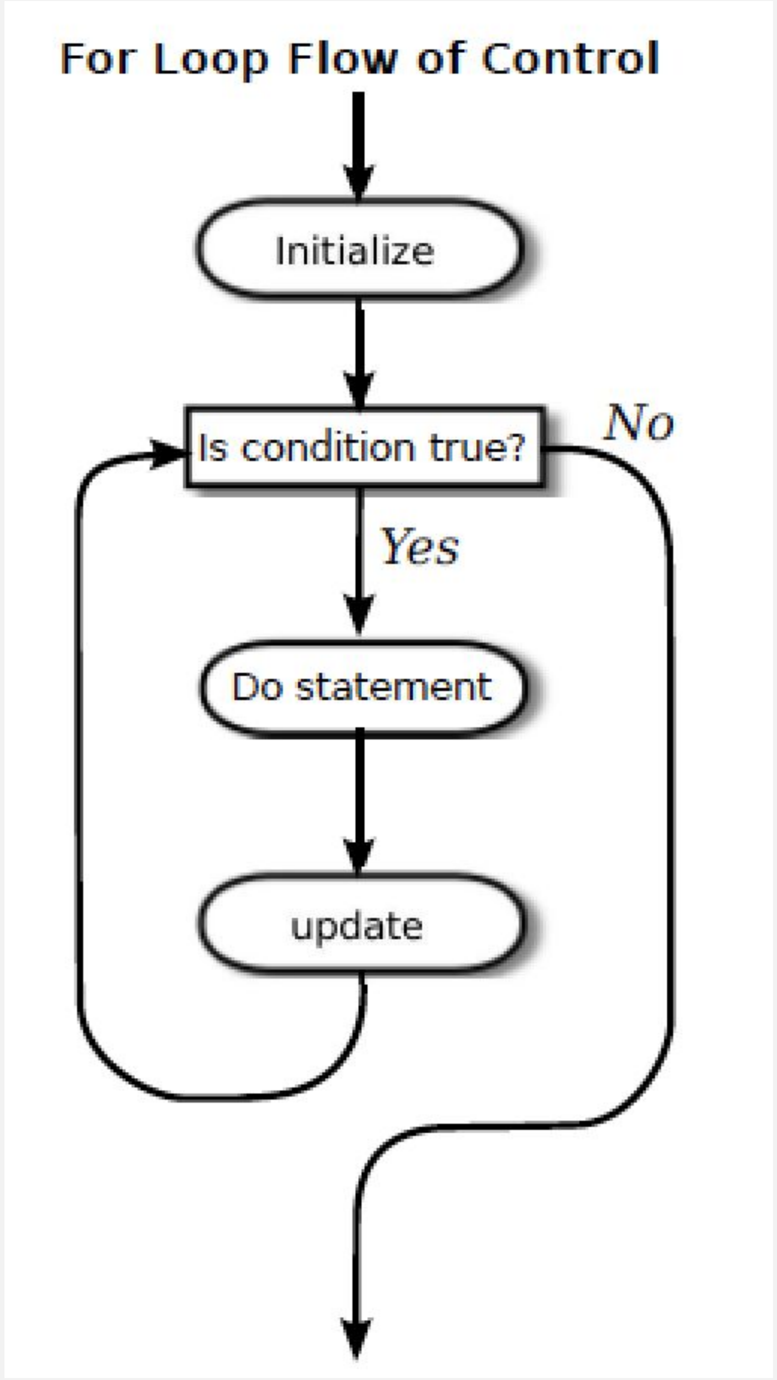


DO-WHILE LOOP See `LoopDemo.java`

```
Scanner in = new Scanner(System.in);
boolean okay; double inputNum;
do{
   System.out.print("Enter a number: ");
   if (in.hasNextDouble()) {
      okay = true;
   }
   else {
      okay = false;
      String word = in.next();
      System.err.println(word + " is not a number");
   }
} while (!okay);
inputNum = in.nextDouble();
```

The "do" part, executes at least once.

Loop exit check

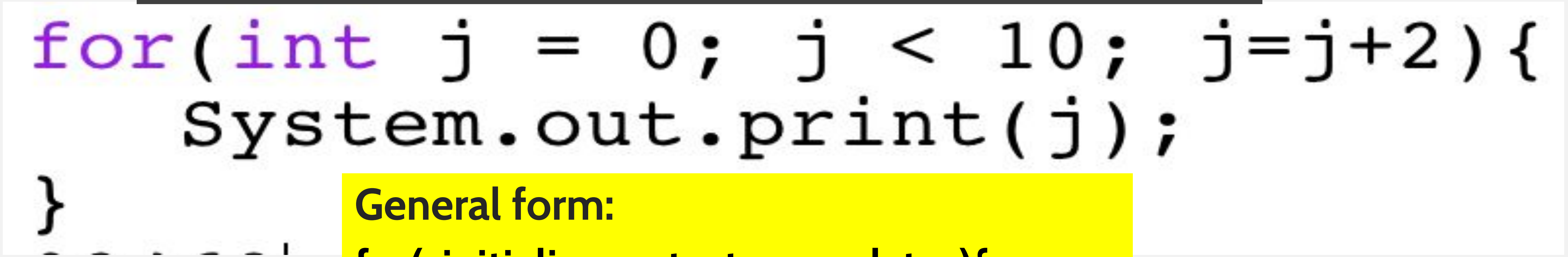


FOR LOOP

General form:

```
for(<initialize> ; <test> ; <update>){
    do a bunch of stuff (in loop body);
}
```

FOR LOOP: THINK-PAIR-SHARE



```
for(int j = 0; j < 10; j=j+2){
    System.out.print(j);
}
```

General form:

```
for(<initialize> ; <test> ; <update>){
    do a bunch of stuff (in loop body);
}
```

What does this code do?

Read the code. Think and then write the pseudocode.

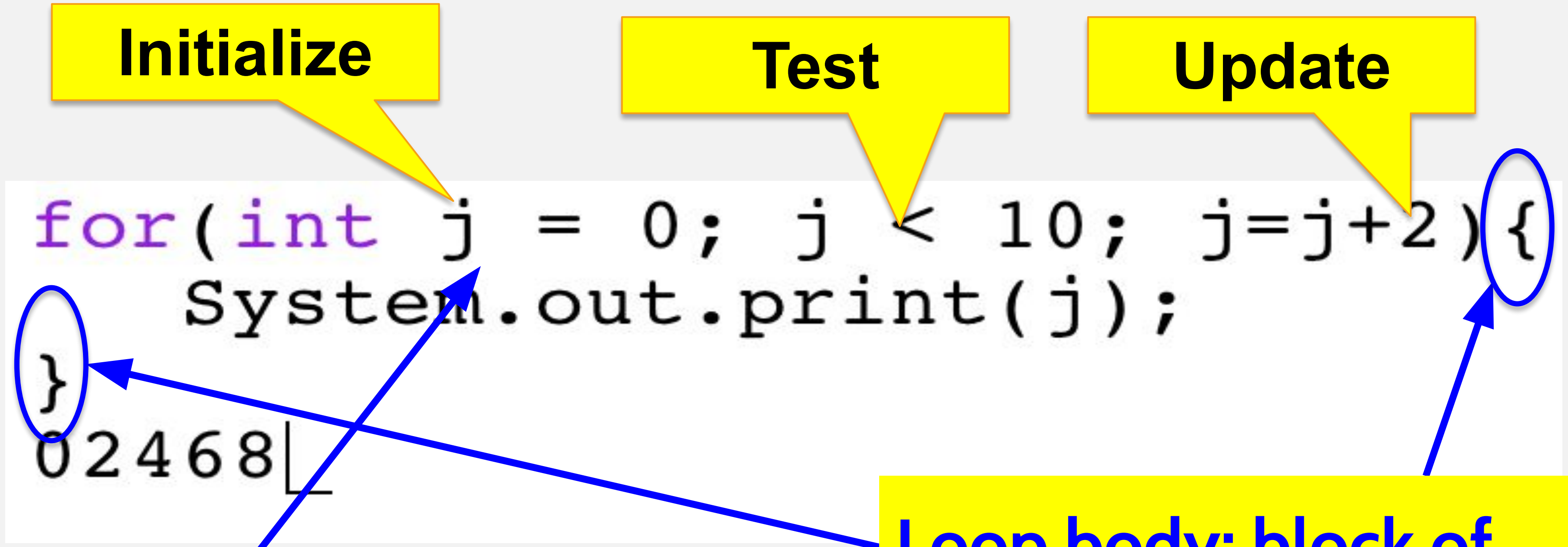
When you finish, pair up and discuss your answers.

Finally, let's share.

FOR LOOP SYNTAX

```
Declare and initialize j
Test j condition
    Print j
Update j
```

Repeat

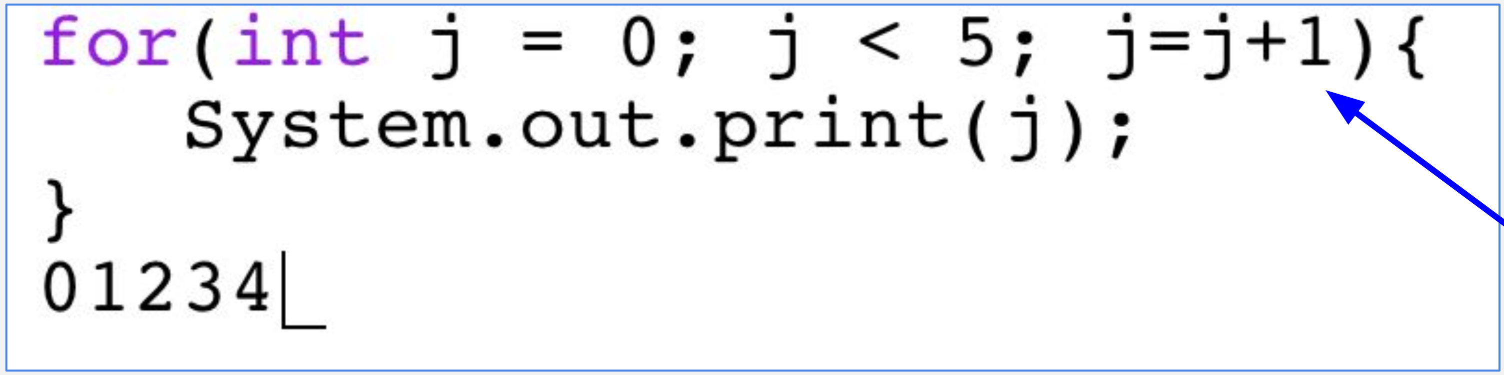


```
for(int j = 0; j < 10; j=j+2){
    System.out.print(j);
}
02468
```

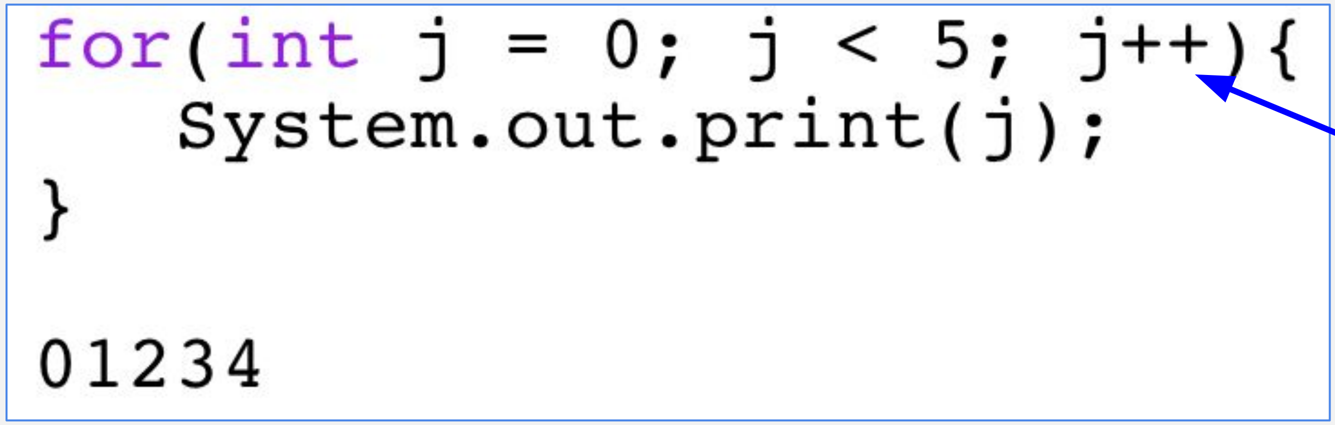
The *for* loop features the use of a loop counter variable.

Loop body: block of statements enclosed by { }

FOR STATEMENT INCREMENT VARIATIONS

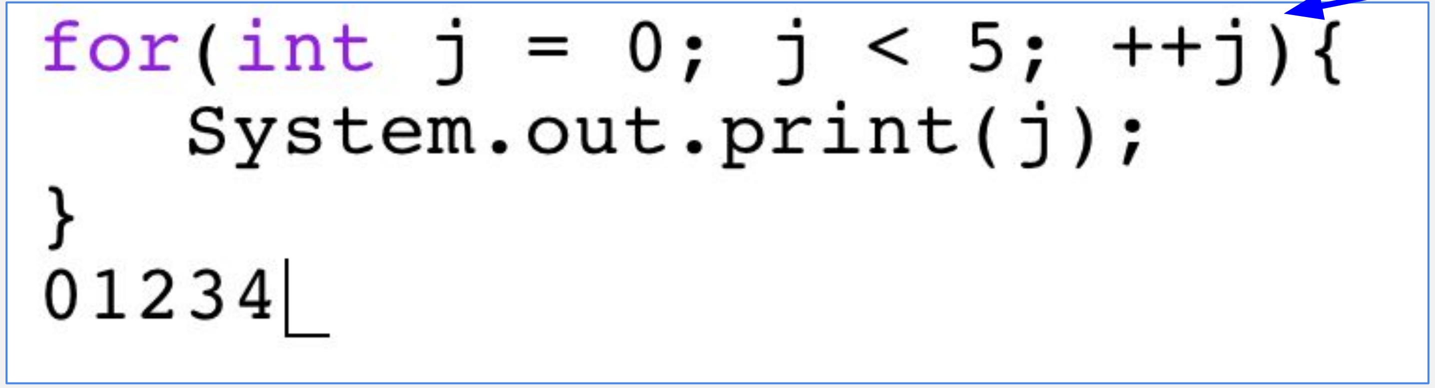


```
for(int j = 0; j < 5; j=j+1){
    System.out.print(j);
}
01234
```



```
for(int j = 0; j < 5; j++){
    System.out.print(j);
}

01234
```



```
for(int j = 0; j < 5; ++j){
    System.out.print(j);
}
01234
```

All increments are equivalent in these loops.

FOR LOOP INCREMENT NOTES

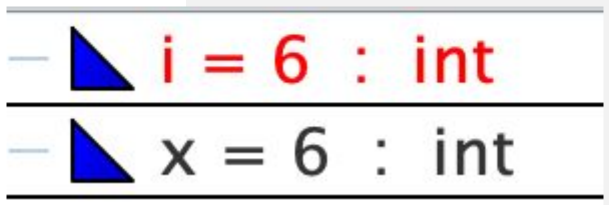
1. Shortcut operators: `++` and `--`
2. The operator can appear as ++i (*prefix form*) or as i++ (*postfix form*).

   `++i` increments `i` first; then evaluates result

   `i++` evaluates result first; then increments `i`.

```
int i = 5;
int x = ++i;
```

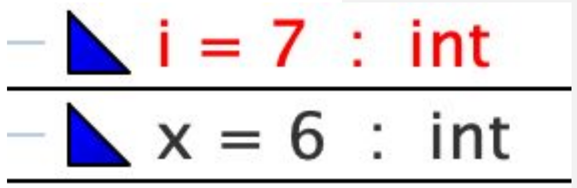
Answer: `x` is 6; `i` is 6



```
i = 6 : int
x = 6 : int
```

```
x = i++;
```

Answer: `x` is 6; `i` is 7



```
i = 7 : int
x = 6 : int
```

WHILE vs FOR SYNTAX

Initialize

Test

Update

```java
for (int i=0; i<10; i++){
   System.out.println("Random number "+ Math.random());
}
```

Initialize

Test

```java
int i = 0;
while (i<10){
   System.out.println("Random number "+ Math.random());
   i++;
}
```

Update

They both do the same operations.
The difference is mainly for convenience.

WHILE LOOPS VS. *FOR* LOOPS

- They are equivalent in operation.
- Sometimes it's more natural to use one or the other.
- *for* loops tend to be used when *we know ahead of time when we will end the loop*.
  - "from a to z"
  - "from 1 to 10 by twos"
- *while* loops tend to be used *when termination condition is more complicated*:
  - "loop until a certain input is seen".

CLICKER QUESTION #3

```
for(int j = 2; j < 100; j = j*j){
   System.out.print(j + " ");
}
```

A. 2 4 6 8 10
B. 2 4 16
C. 2 4 16 256
D. 2 4 16 32 64
E. Error

What does this loop print when executed?

READY FOR THE ANSWER?

CLICKER QUESTION #3 ANSWER

```
for(int j = 2; j < 100; j = j*j){
    System.out.print(j + " ");
}
```

What does this loop print when executed?

A. 2 4 6 8 10 not product of j
B. 2 4 16 correct
C. 2 4 16 256 should be <100
D. 2 4 16 32 64 incorrect (32 and 64 incorrect)
E. Error

CLICKER QUESTION #4

```
for(int j = 2; j < 100; j = j*j){
    System.out.print(j + " ");
}
System.out.print(j);
```

What does this loop print when executed?

A. 2 4 6 8 10
B. 2 4 16 256
C. 2 4 16 16
D. 2 4 16 32 128
E. Error

READY FOR THE ANSWER?

CLICKER QUESTION #4 ANSWER

```java
for(int j = 2; j < 100; j = j*j){
    System.out.print(j + "  ");
}
System.out.print(j);
```

Error: variable out of scope

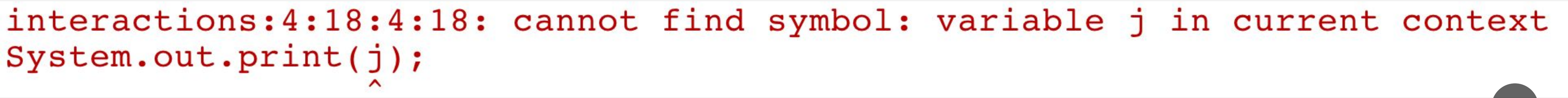
B. 2 4 16 256

C. 2 4 16 16

D. 2 4 16 32 128

E. **Error**

Tip: Try the code in the jGrasp Interactions pane.



```
interactions:4:18:4:18: cannot find symbol: variable j in current context
System.out.print(j);
                 ^
```

TO-DO LIST:

- Check your iClicker grades in Moodle.
- Complete zyBook chapter 6 exercises *before* the exam.
- Communicate with us using the Moodle private forum or Piazza.
- Remember to start early on projects :-)