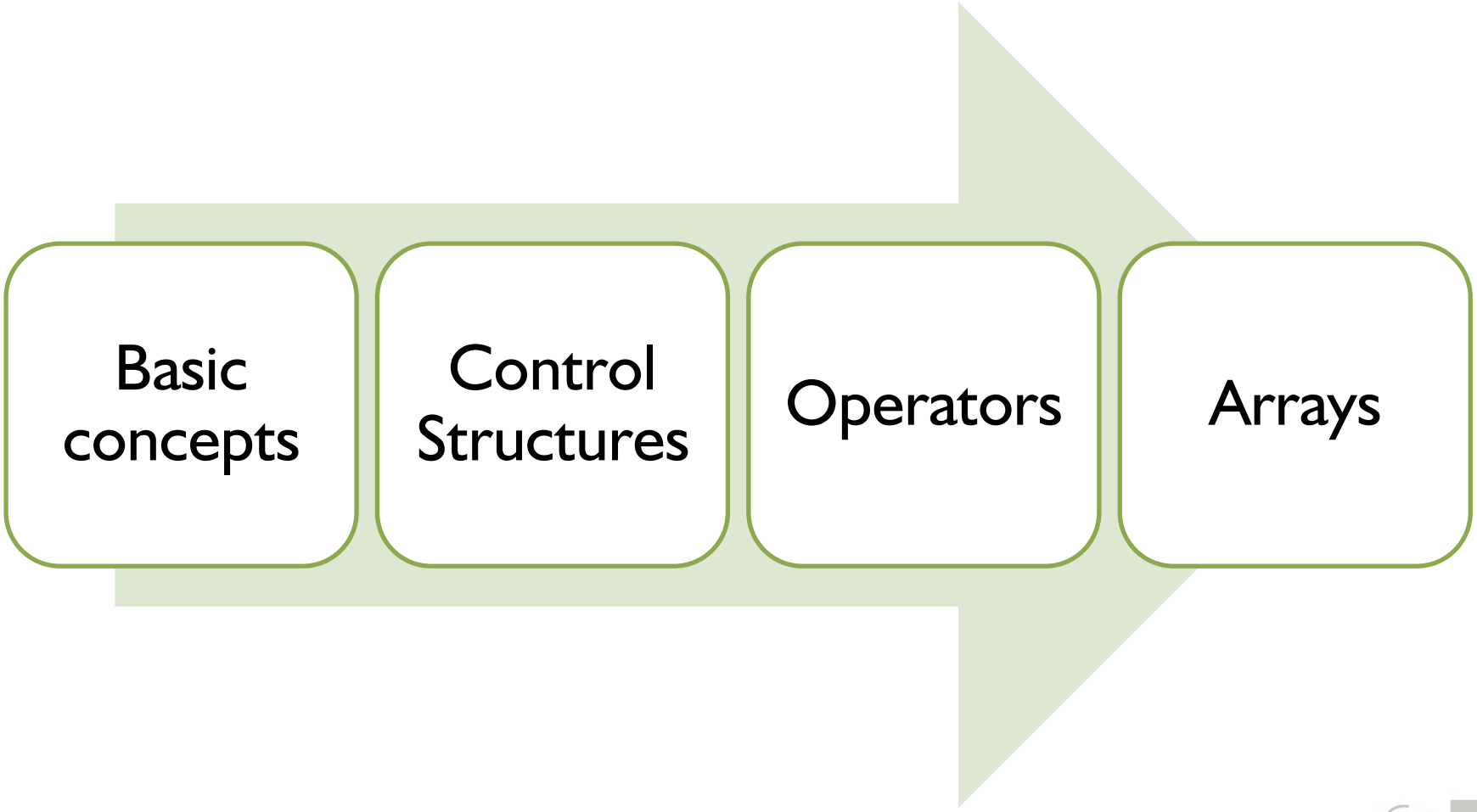
Basic programming review

Christian Rodríguez Bustos

Object Oriented Programming

Agenda



Basic concepts

Algorithm

Flow diagram

Speudo code

An **algorithm** is a step-by-step procedure

Any computing problem can be solved by executing a series of actions in a specific order.

Finite

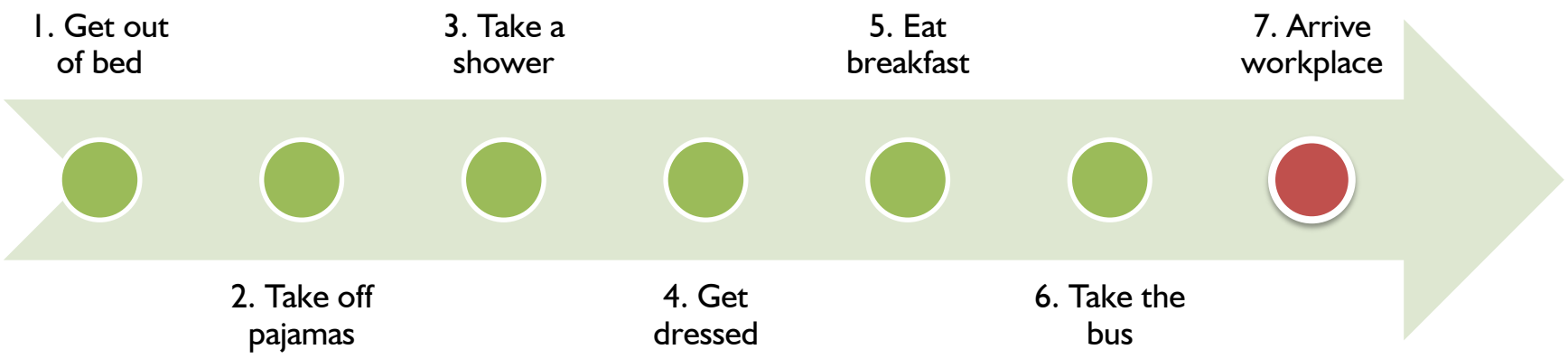
Deterministic

Precision

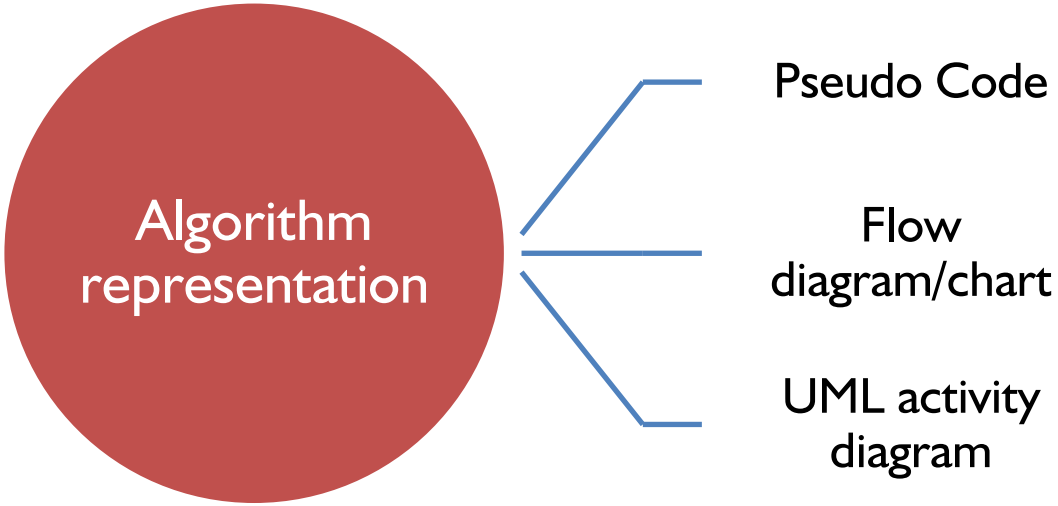
An **algorithm** is a step-by-step procedure

Problem that we want to solve

Going to work



Algorithms can be represented in several ways



Pseudo codes are informal descriptions of algorithms

Informal descriptions or languages help programmers to develop algorithms without having to worry about the strict details of a programming language syntax.

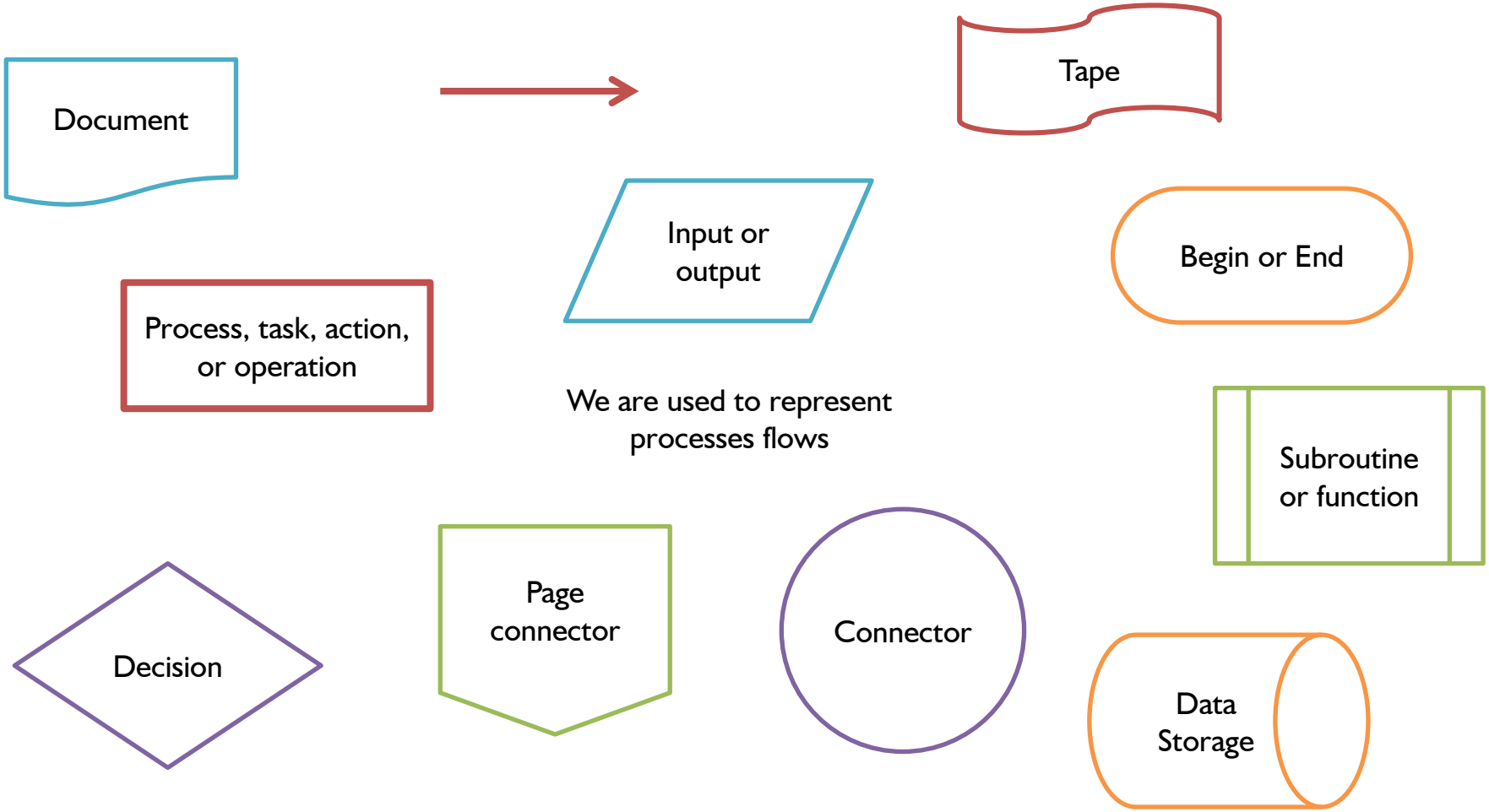
```
Set grade counter to one

While grade counter is less than or equal to ten

      Input the next grade
      Add the grade into the total

Set the class average to the total divided by ten

Print the class average.
```

I am a pseudo code

All pseudo codes should be:

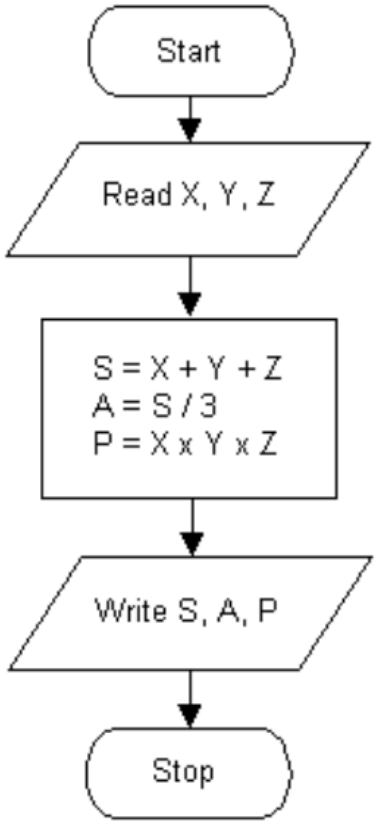
- Human readable
- Can easily be converted to any programming language

Flow diagrams are used to represent algorithms



Resource: [What do the different flowchart shapes mean?]

These Flow diagram and Pseudo code are equivalent

Flowchart	Pseudocode
	Read X, Y, Z Compute Sum (S) as X + Y + Z Compute Average (A) as S / 3 Compute Product (P) as X x Y x Z Write (Display) the Sum, Average and Product

Activity diagrams describe the workflow of a system

Activities

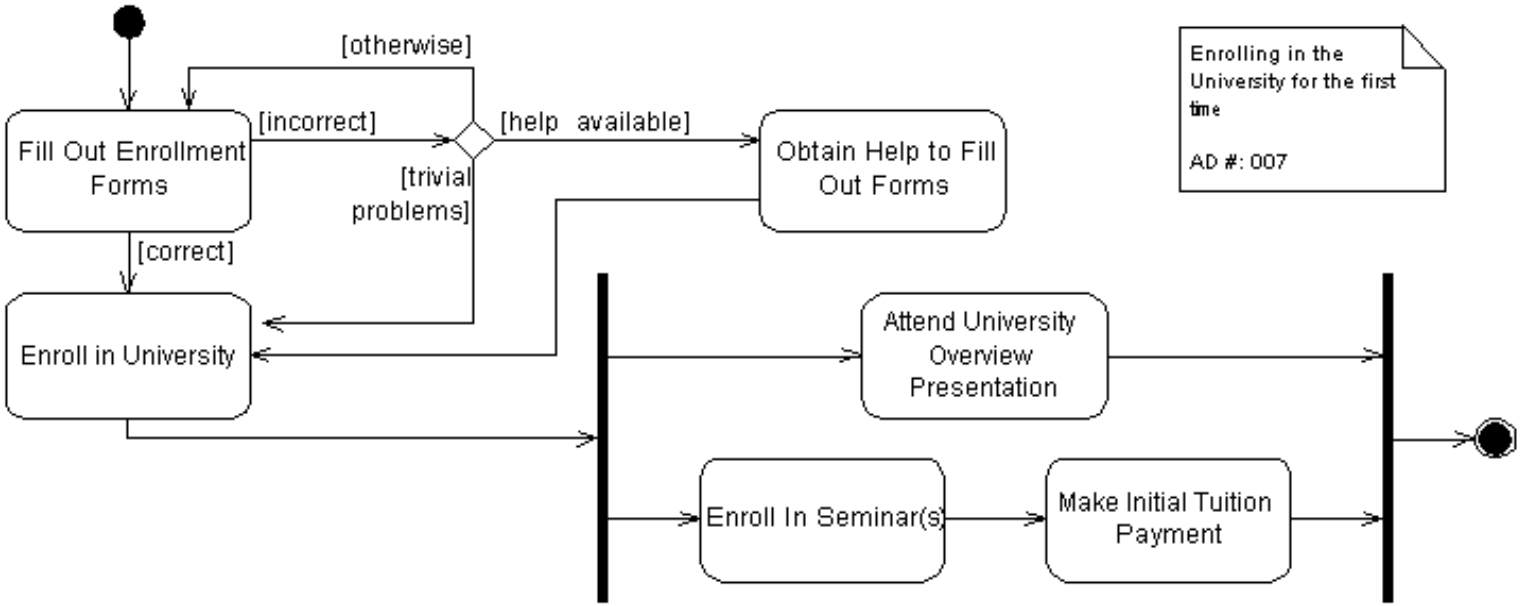
Decisions

Start (*split*) or end (*join*) of concurrent activities

The start (*initial state*) of the workflow

The end (*final state*).

Activity diagrams describe the workflow of a system



I am a UML activity diagram

Control Structures

Sequence Structure

Selection Statements

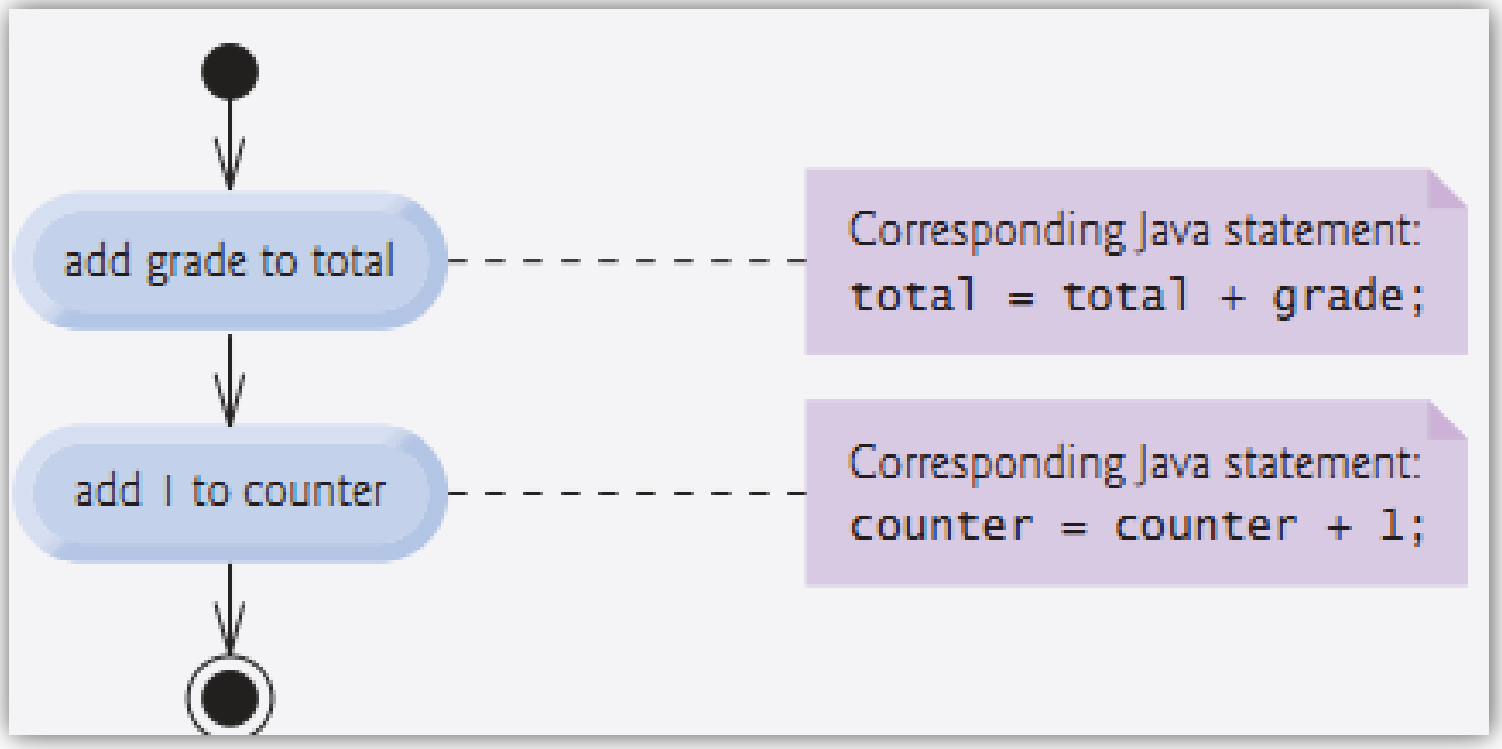
Repetition Statements

Control Structures

Programs are formed by combining as many **sequence**, **selection** and **repetition** statements.

selection	repetition
if	while
if…else	do…while
switch	for

Sequence structure



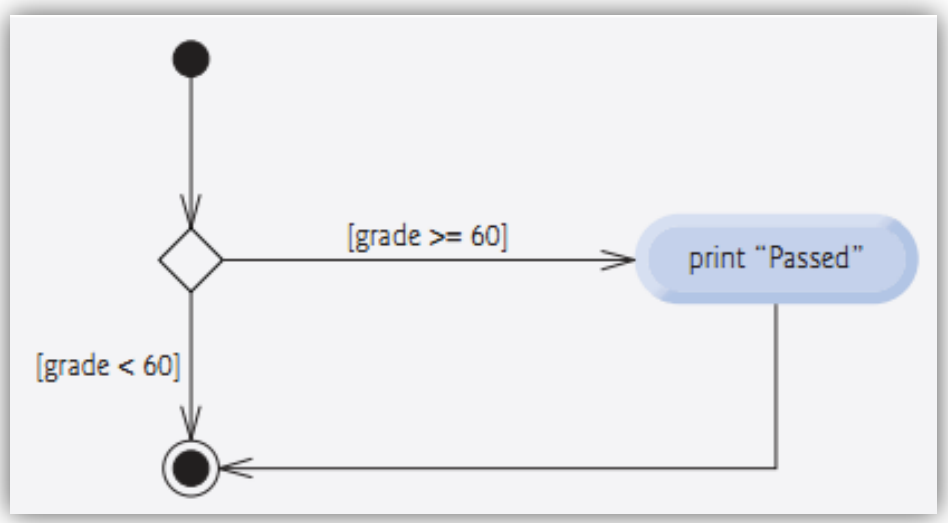
An ordered execution of two or more statements are called **sequence structure**

IF Selection Statement

If student's grade is greater than or equal to 60

Print "**Passed**"

```
if ( studentGrade >= 60 )
   System.out.println( "Passed" );
```



IF..ELSE Selection Statement

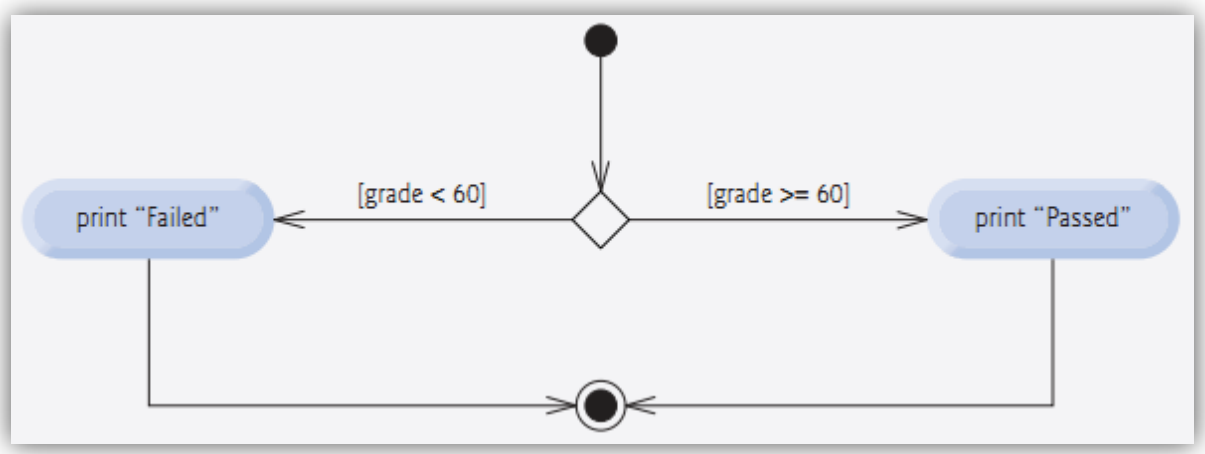
If student's grade is greater than or equal to 60

Print "**Passed**"

Else

Print "**Failed**"

```
if ( grade >= 60 )
   System.out.println( "Passed" );
else
   System.out.println( "Failed" );
```



IF..ELSE Selection Statement abbreviated form

```
if ( grade >= 60 )
   System.out.println( "Passed" );
else
   System.out.println( "Failed" );
```

=

```
System.out.println( studentGrade >= 60 ? "Passed" : "Failed" );
```

Nested IF..ELSE Selection Statement

```
if ( studentGrade >= 90 )
   System.out.println( "A" );
else
   if ( studentGrade >= 80 )
      System.out.println( "B" );
   else
      if ( studentGrade >= 70 )
         System.out.println( "C" );
      else
         if ( studentGrade >= 60 )
            System.out.println( "D" );
         else
            System.out.println( "F" );
```

If student's grade is greater than or equal to 90
    Print "**A**"
else
    If student's grade is greater than or equal to 80
        Print "**B**"
    else
        If student's grade is greater than or equal to 70
            Print "**C**"
        else
            If student's grade is greater than or equal to 60
                Print "**D**"
            else
                Print "**F**"

Do not forget…

Indent both body statements of an if…else statement.

```
if ( grade >= 60 ) {
    System.out.println( "Passed" );
} else {
    System.out.println( "Failed" );
}
```

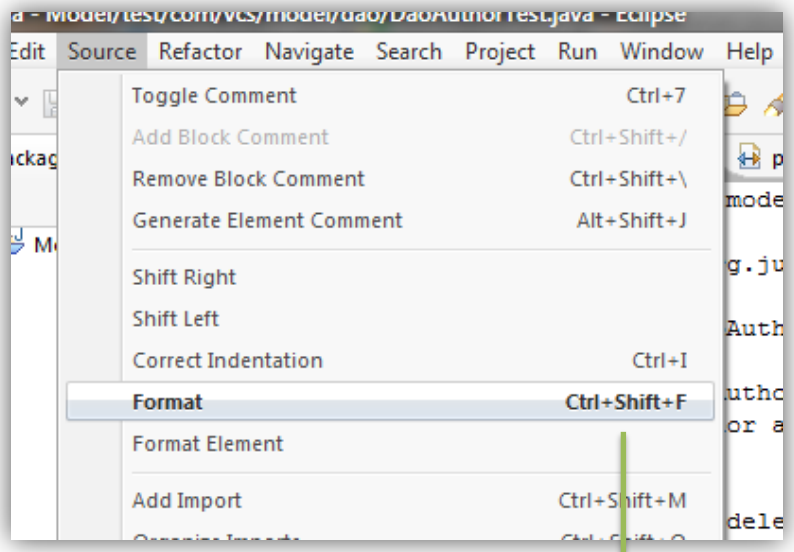
Always using braces in an if…else (or other) statement helps prevent their accidental omission

Do not be the beast

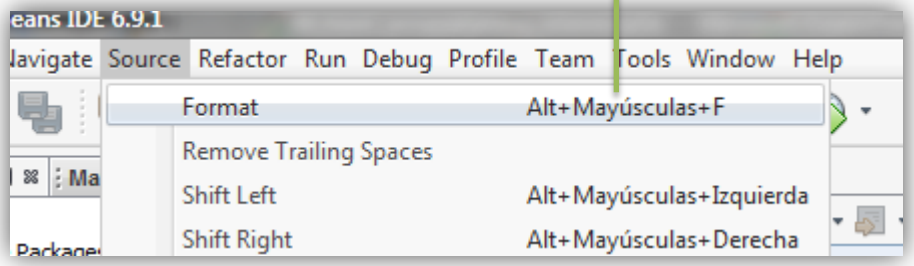
Ugly code is written by ugly people.


```
if(grade>=60)System.out.
println("Passed"          );else System.
out.println("Failed");
```

Do not worry, we can format the code automatically



Eclipse



NetBeans

Alt+Mayúsculas+F

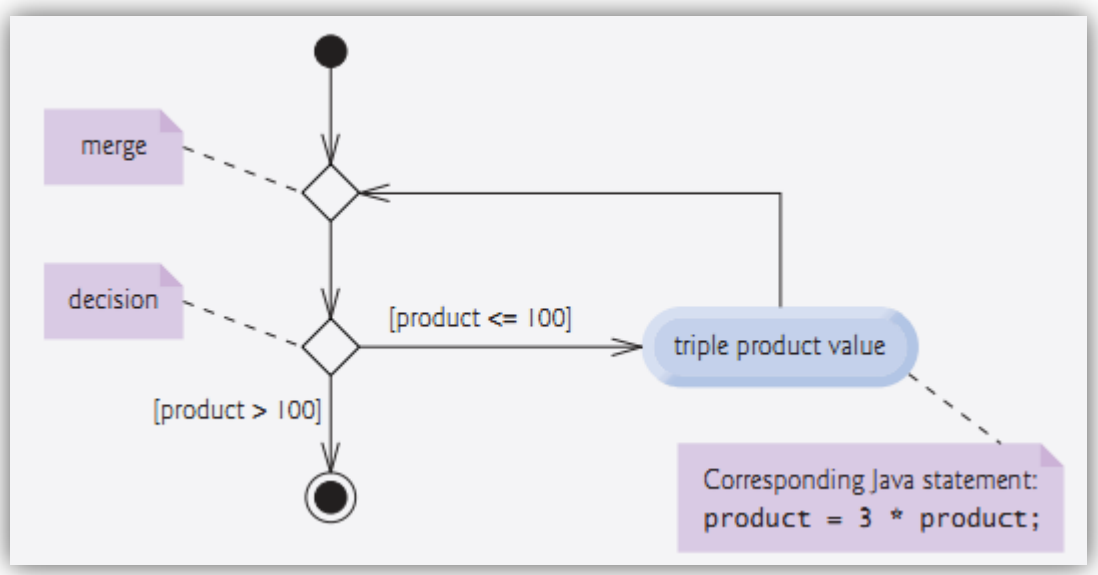
Ctrl+Shift+F

WHILE Repetition Statement

While product is less or equal than 100 products

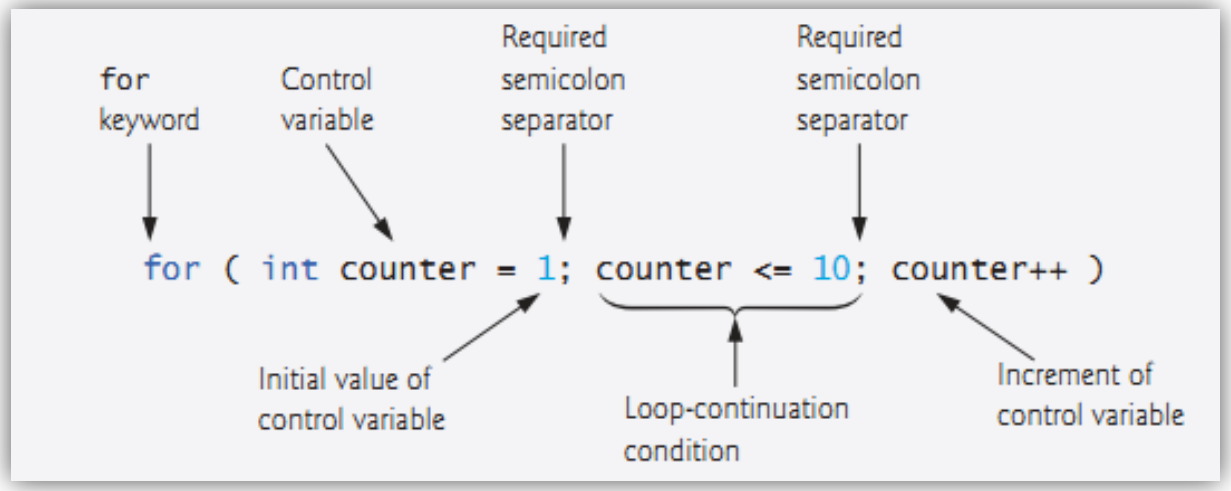
Multiply by 3 the number of products

```
int product = 3;

while ( product <= 100 )
    product = 3 * product;
```



Be careful with infinite loops!!

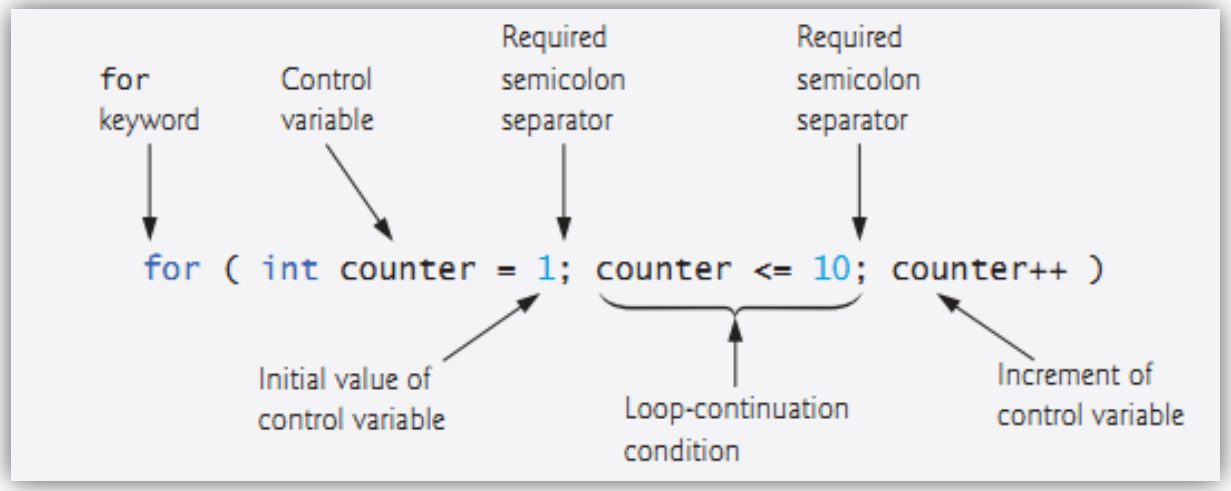
FOR Repetition Statement




```
// for statement header includes initialization,
// loop-continuation condition and increment
for ( int counter = 1; counter <= 10; counter++ )
   System.out.printf( "%d  ", counter );
```

?????

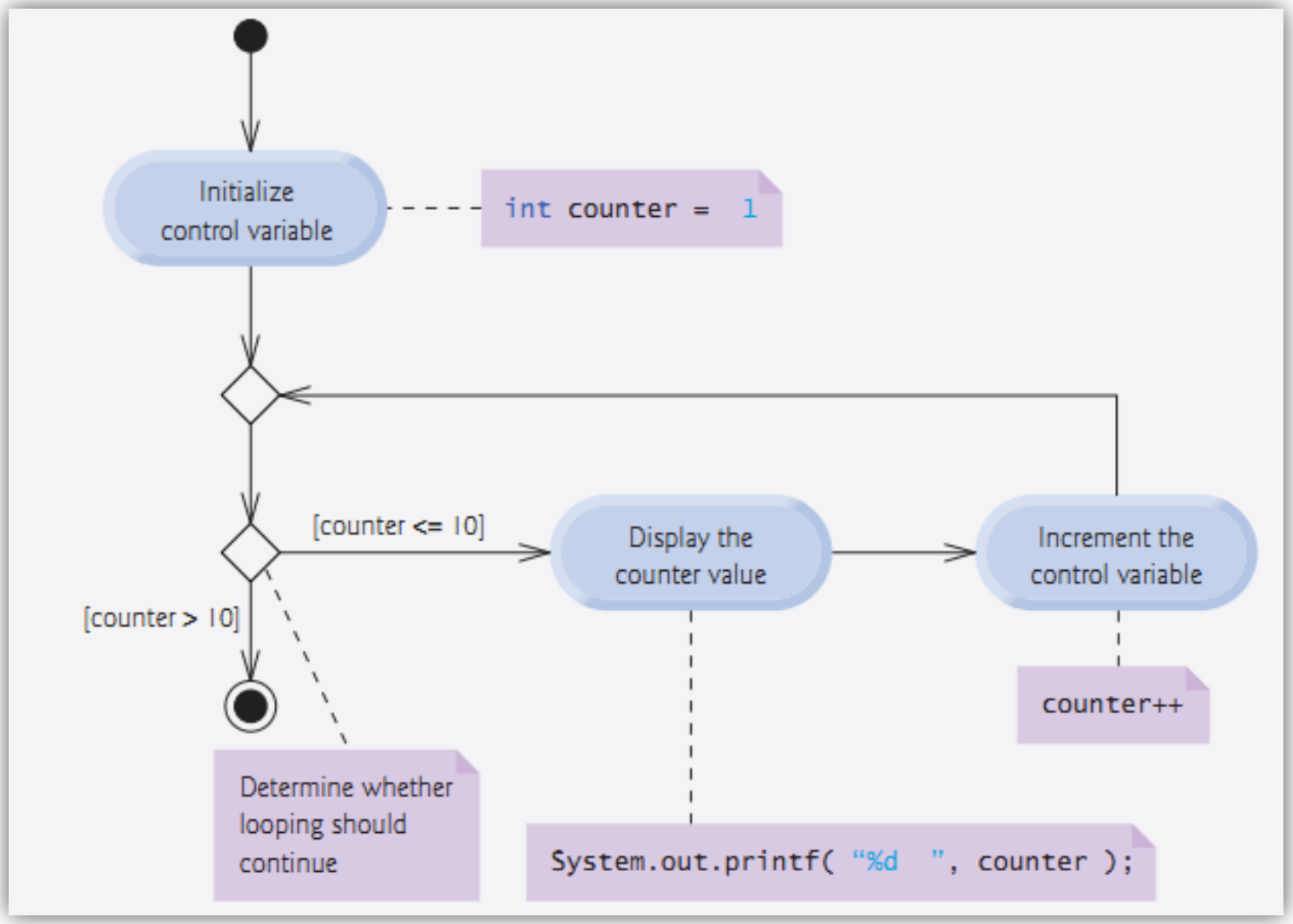
FOR Repetition Statement




```
// for statement header includes initialization,
// loop-continuation condition and increment
for ( int counter = 1; counter <= 10; counter++ )
   System.out.printf( "%d  ", counter );
```

```
1  2  3  4  5  6  7  8  9  10
```

FOR Repetition Statement



FOR Statements header examples

Vary the control variable from **1 to 100 in increments of 1**

```
for ( int i = 1; i <= 100; i++ )
```

Vary the control variable from **100 to 1 in decrements of 1**

```
for ( int i = 100; i >= 1; i-- )
```

Vary the control variable from **7 to 77 in increments of 7**

```
for ( int i = 7; i <= 77; i += 7 )
```

Vary the control variable from **20 to 2 in decrements of 2**

```
for ( int i = 20; i >= 2; i -= 2 )
```

Vary the control variable over the following sequence of values: ?, ?, ?, ?, ?, ?, ?

```
for ( int i = 2; i <= 20; i += 3 )
```

Vary the control variable over the following sequence of values: ?, ?, ?, ?, ?,?, ?, ?, ?, ?

```
for ( int i = 99; i >= 0; i -= 11 )
```

FOR Statements header examples

Vary the control variable from **1 to 100 in increments of 1**

```
for ( int i = 1; i <= 100; i++ )
```

Vary the control variable from **100 to 1 in decrements of 1**

```
for ( int i = 100; i >= 1; i-- )
```

Vary the control variable from **7 to 77 in increments of 7**

```
for ( int i = 7; i <= 77; i += 7 )
```

Vary the control variable from **20 to 2 in decrements of 2**

```
for ( int i = 20; i >= 2; i -= 2 )
```

Vary the control variable over the following sequence of values: **2, 5, 8, 11, 14, 17, 20**

```
for ( int i = 2; i <= 20; i += 3 )
```

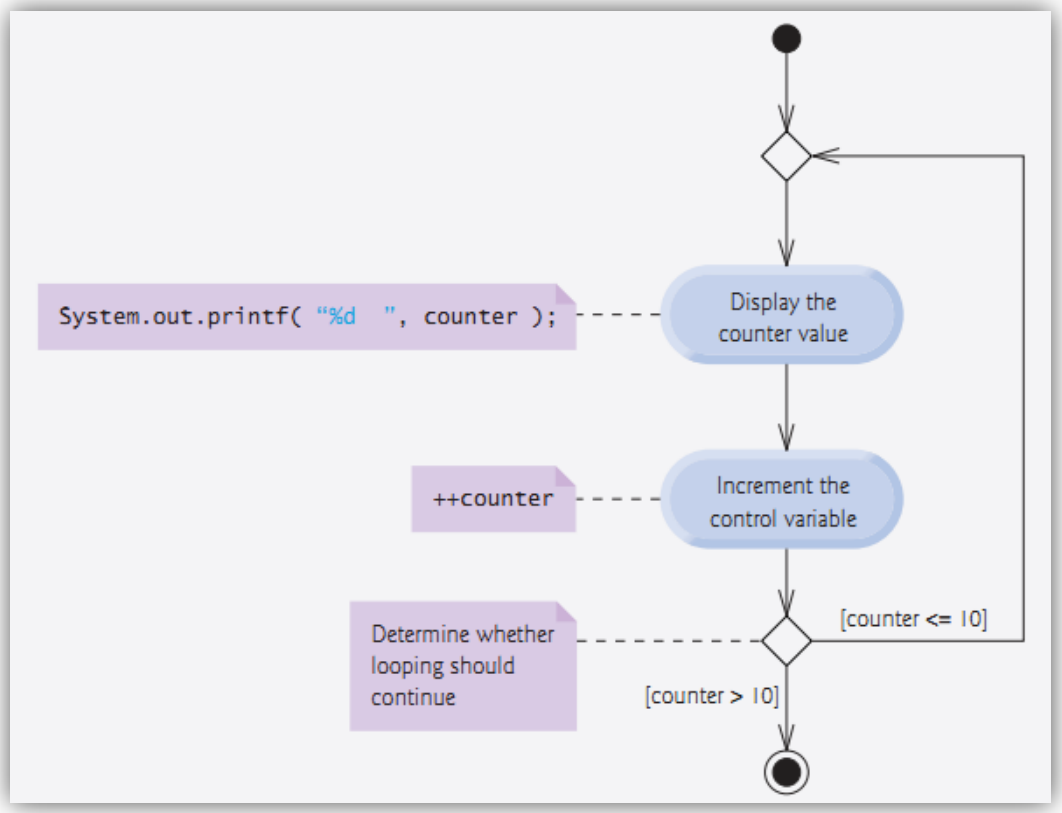
Vary the control variable over the following sequence of values: **99, 88, 77, 66, 55,44, 33, 22, 11, 0**

```
for ( int i = 99; i >= 0; i -= 11 )
```

FOR Statement example

```java
// Fig. 5.5: Sum.java
// Summing integers with the for statement.

public class Sum
{
   public static void main( String args[] )
   {
      int total = 0; // initialize total

      // total even integers from 2 through 20
      for ( int number = 2; number <= 20; number += 2 )
         total += number;

      System.out.printf( "Sum is %d\n", total ); // display results
   } // end main
} // end class Sum
```

DO…WHILE Repetition Statement



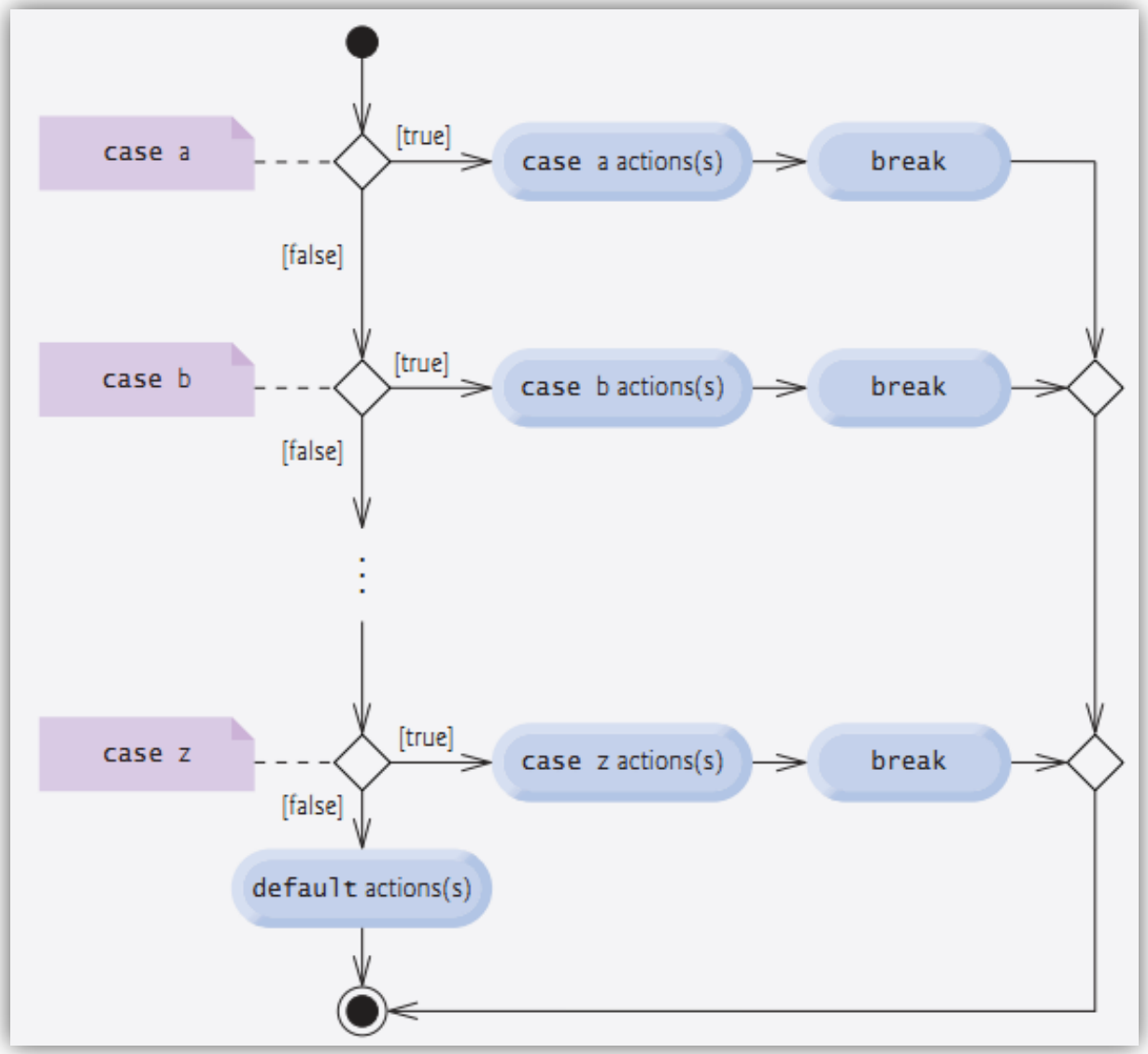

```
do
{
    statement
} while ( condition );
```

```
int counter = 1; // initialize counter

do
{
   System.out.printf( "%d  ", counter );
   ++counter;
} while ( counter <= 10 ); // end do...while
```

```
1  2  3  4  5  6  7  8  9  10
```

Remember always include braces !!!

SWITCH Multiple-Selection Statement



SWITCH Multiple-Selection Statement

```
// determine which grade was entered
switch ( grade / 10 )
{
   case 9:  // grade was between 90
   case 10: // and 100
      ++aCount; // increment aCount
      break; // necessary to exit switch

   case 8: // grade was between 80 and 89
      ++bCount; // increment bCount
      break; // exit switch

   case 7: // grade was between 70 and 79
      ++cCount; // increment cCount
      break; // exit switch

   case 6: // grade was between 60 and 69
      ++dCount; // increment dCount
      break; // exit switch

   default: // grade was less than 60
      ++fCount; // increment fCount
      break; // optional; will exit switch anyway
} // end switch
```

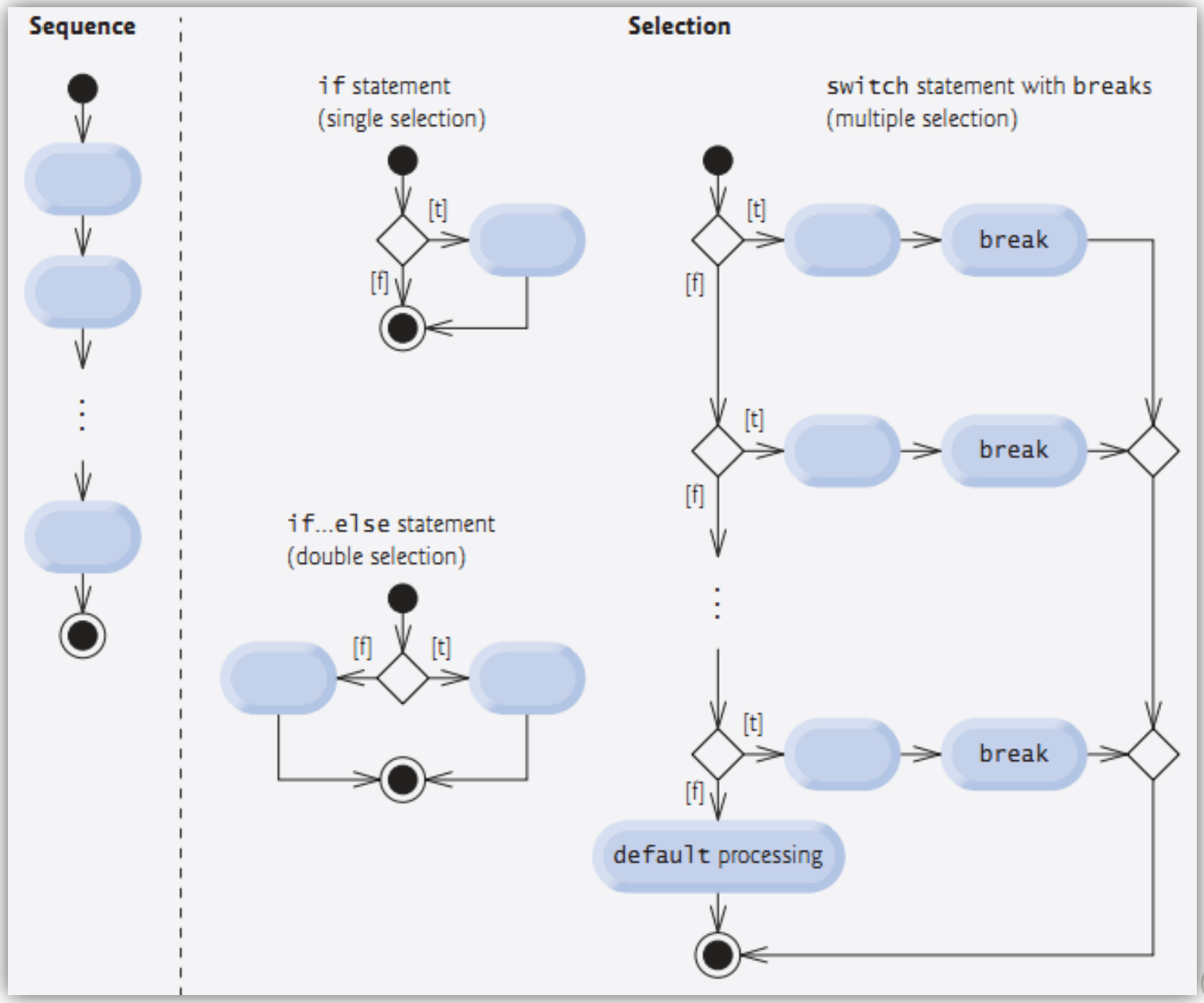
BREAK and CONTINUE Statements

```
int count; // control variable also used after loop terminates

for ( count = 1; count <= 10; count++ ) // loop 10 times
{
   if ( count == 5 ) // if count is 5,
      break;         // terminate loop

   System.out.printf( "%d ", count );
} // end for
```

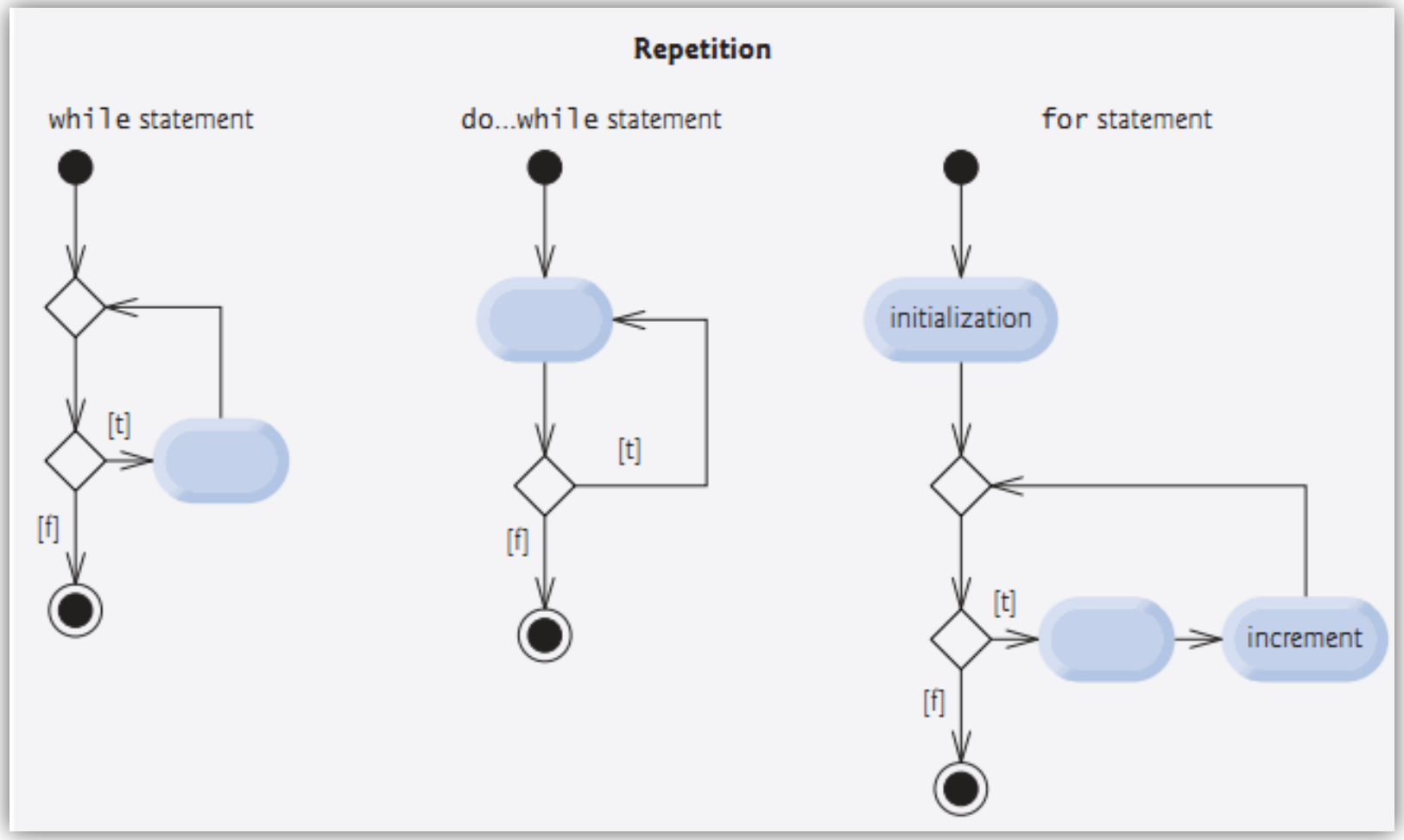
Break

Continue

```
for ( int count = 1; count <= 10; count++ ) // loop 10 times
{
   if ( count == 5 ) // if count is 5,
      continue; // skip remaining code in loop

   System.out.printf( "%d ", count );
} // end for
```


Sequence
Selection
if statement
(single selection)
[t]
[f]
switch statement with breaks
(multiple selection)
[t]
[f]
break
[t]
[f]
break
if...else statement
(double selection)
[f]
[t]
[t]
[f]
break
default processing

Summary


Repetition
while statement
[t]
[f]
do...while statement
[t]
[f]
for statement
initialization
[t]
increment
[f]

Operators

Logical Operators

Assignment Operators

Increment and Decrement Operators

Logical Operators

Conditional AND (&&)

```
if ( gender == FEMALE && age >= 65 )
   ++seniorFemales;
```

Conditional OR (||)

```
if ( ( semesterAverage >= 90 ) || ( finalExam >= 90 ) )
   System.out.println ( "Student grade is A" );
```

Logical Negation (!)

```
if ( ! ( grade == sentinelValue ) )
   System.out.printf( "The next grade is %d\n", grade );
```

Logical Operators - Truth tables

```
Conditional AND (&&)
false && false: false
false && true: false
true && false: false
true && true: true

Conditional OR (||)
false || false: false
false || true: true
true || false: true
true || true: true

Boolean logical AND (&)
false & false: false
false & true: false
true & false: false
true & true: true
```

```
Boolean logical inclusive OR (|)
false | false: false
false | true: true
true | false: true
true | true: true

Boolean logical exclusive OR (^)
false ^ false: false
false ^ true: true
true ^ false: true
true ^ true: false

Logical NOT (!)
!false: true
!true: false
```

Assignment Operators

```
variable = variable operator expression;
```

```
c = c + 3;
```

```
c += 3;
```

Compound Assignment Operators

Assignment operator	Sample expression	Explanation	Assigns
Assume: `int c = 3, d = 5, e = 4, f = 6, g = 12;`			
`+=`	`c += 7`	`c = c + 7`	10 to c
`-=`	`d -= 4`	`d = d - 4`	1 to d
`*=`	`e *= 5`	`e = e * 5`	20 to e
`/=`	`f /= 3`	`f = f / 3`	2 to f
`%=`	`g %= 9`	`g = g % 9`	3 to g

Increment and Decrement Operators

Operator	Operator name	Sample expression	Explanation
++	prefix increment	++a	Increment a by 1, then use the new value of a in the expression in which a resides.
++	postfix increment	a++	Use the current value of a in the expression in which a resides, then increment a by 1.
--	prefix decrement	--b	Decrement b by 1, then use the new value of b in the expression in which b resides.
--	postfix decrement	b--	Use the current value of b in the expression in which b resides, then decrement b by 1.

Prefix and Postfix Example

```java
int c;

// demonstrate postfix increment operator
c = 5; // assign 5 to c
System.out.println( c );   // prints 5
System.out.println( c++ ); // prints 5 then postincrements
System.out.println( c );   // prints 6

System.out.println(); // skip a line

// demonstrate prefix increment operator
c = 5; // assign 5 to c
System.out.println( c );   // prints 5
System.out.println( ++c ); // preincrements then prints 6
System.out.println( c );   // prints 6
```

```
5
5
6

5
6
6
```

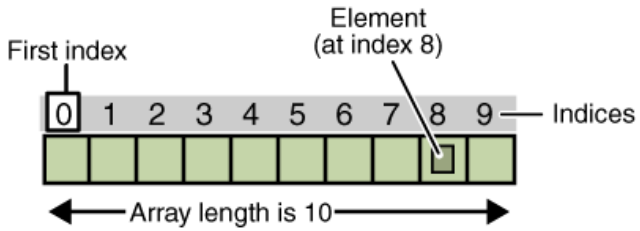
Java Primitive Types

Type	Size in bits	Values	Standard
`boolean`		`true` or `false`	
[*Note:* A `boolean`'s representation is specific to the Java Virtual Machine on each platform.]			
`char`	16	`'\u0000'` to `'\uFFFF'` (0 to 65535)	(ISO Unicode character set)
`byte`	8	–128 to +127 (-2^7 to $2^7 - 1$)	
`short`	16	–32,768 to +32,767 (-2^{15} to $2^{15} - 1$)	
`int`	32	–2,147,483,648 to +2,147,483,647 (-2^{31} to $2^{31} - 1$)	
`long`	64	–9,223,372,036,854,775,808 to +9,223,372,036,854,775,807 (-2^{63} to $2^{63} - 1$)	
`float`	32	*Negative range:* –3.4028234663852886E+38 to –1.40129846432481707e–45 *Positive range:* 1.40129846432481707e–45 to 3.4028234663852886E+38	(IEEE 754 floating point)
`double`	64	*Negative range:* –1.7976931348623157E+308 to –4.94065645841246544e–324 *Positive range:* 4.94065645841246544e–324 to 1.7976931348623157E+308	(IEEE 754 floating point)

Arrays

Arrays are containers

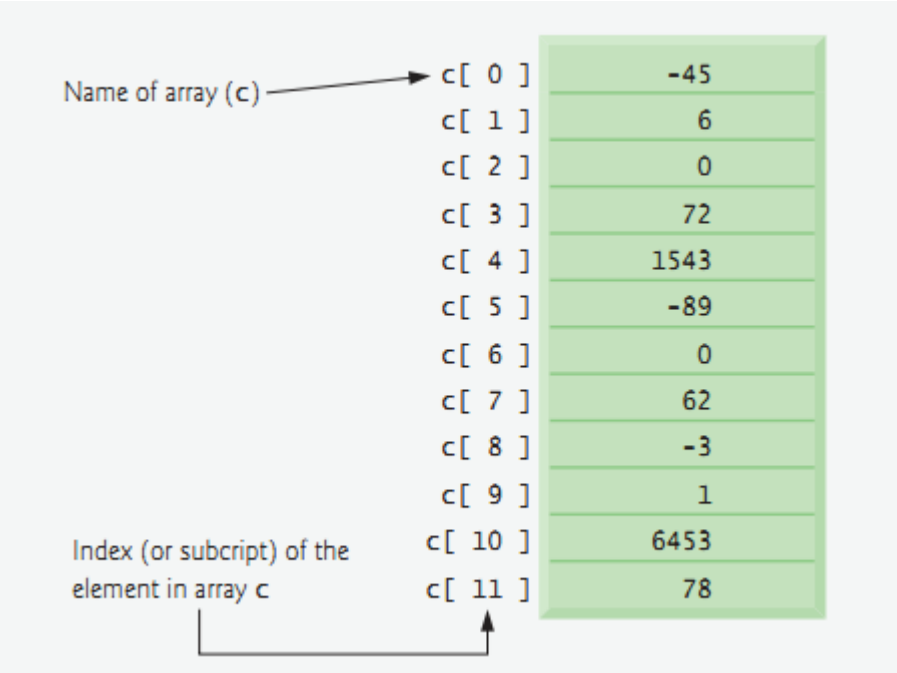
Container object that holds a fixed number of values of a **single type**




```
int c[] = new int[ 12 ];
```

or

```
int c[];
c = new int[ 12 ];
```



Arrays use example I

```java
// Fig. 7.2: InitArray.java
// Creating an array.

public class InitArray
{
   public static void main( String args[] )
   {
      int array[]; // declare array named array

      array = new int[ 10 ]; // create the space for array

      System.out.printf( "%s%8s\n", "Index", "Value" ); // column headings

      // output each array element's value
      for ( int counter = 0; counter < array.length; counter++ )
         System.out.printf( "%5d%8d\n", counter, array[ counter ] );
   } // end main
} // end class InitArray
```

```
Index   Value
    0       0
    1       0
    2       0
    3       0
    4       0
    5       0
    6       0
    7       0
    8       0
    9       0
```

Arrays use example 2

```java
// Fig. 7.4: InitArray.java
// Calculating values to be placed into elements of an array.

public class InitArray
{
   public static void main( String args[] )
   {
      final int ARRAY_LENGTH = 10; // declare constant
      int array[] = new int[ ARRAY_LENGTH ]; // create array

      // calculate value for each array element
      for ( int counter = 0; counter < array.length; counter++ )
         array[ counter ] = 2 + 2 * counter;

      System.out.printf( "%s%8s\n", "Index", "Value" ); // column headings

      // output each array element's value
      for ( int counter = 0; counter < array.length; counter++ )
         System.out.printf( "%5d%8d\n", counter, array[ counter ] );
   } // end main
} // end class InitArray
```

??????

Arrays use example 2

```java
// Fig. 7.4: InitArray.java
// Calculating values to be placed into elements of an array.

public class InitArray
{
   public static void main( String args[] )
   {
      final int ARRAY_LENGTH = 10; // declare constant
      int array[] = new int[ ARRAY_LENGTH ]; // create array

      // calculate value for each array element
      for ( int counter = 0; counter < array.length; counter++ )
         array[ counter ] = 2 + 2 * counter;

      System.out.printf( "%s%8s\n", "Index", "Value" ); // column he

      // output each array element's value
      for ( int counter = 0; counter < array.length; counter++ )
         System.out.printf( "%5d%8d\n", counter, array[ counter ] );
   } // end main
} // end class InitArray
```

```
Index   Value
    0       2
    1       4
    2       6
    3       8
    4      10
    5      12
    6      14
    7      16
    8      18
    9      20
```

Arrays use example 3

```java
// Fig. 7.3: InitArray.java
// Initializing the elements of an array with an array initializer.

public class InitArray
{
   public static void main( String args[] )
   {
      // initializer list specifies the value for each element
      int array[] = { 32, 27, 64, 18, 95, 14, 90, 70, 60, 37 };

      System.out.printf( "%s%8s\n", "Index", "Value" ); // column headings

      // output each array element's value
      for ( int counter = 0; counter < array.length; counter++ )
         System.out.printf( "%5d%8d\n", counter, array[ counter ] );
   } // end main
} // end class InitArray
```

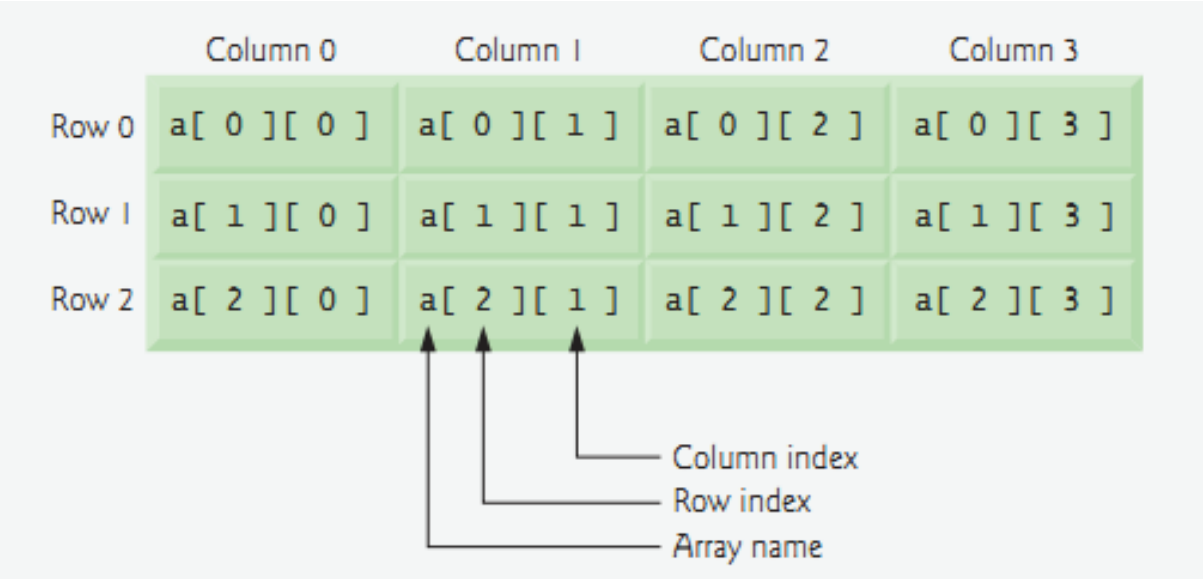
??????

Arrays use example 3

```java
// Fig. 7.3: InitArray.java
// Initializing the elements of an array with an array initializer.

public class InitArray
{
   public static void main( String args[] )
   {
      // initializer list specifies the value for each element
      int array[] = { 32, 27, 64, 18, 95, 14, 90, 70, 60, 37 };

      System.out.printf( "%s%8s\n", "Index", "Value" ); // column headings

      // output each array element's value
      for ( int counter = 0; counter < array.length; counter++ )
         System.out.printf( "%5d%8d\n", counter, array[ counter ] );
   } // end main
} // end class InitArray
```

```
Index   Value
    0      32
    1      27
    2      64
    3      18
    4      95
    5      14
    6      90
    7      70
    8      60
    9      37
```

Working with multidimensional arrays




```
int a[][] = new int [3][4];
```

Multidimensional array use example

```java
// Fig. 7.17: InitArray.java
// Initializing two-dimensional arrays.

public class InitArray
{
   // create and output two-dimensional arrays
   public static void main( String args[] )
   {
      int array1[][] = { { 1, 2, 3 }, { 4, 5, 6 } };
      int array2[][] = { { 1, 2 }, { 3 }, { 4, 5, 6 } };

      System.out.println( "Values in array1 by row are" );
      outputArray( array1 ); // displays array1 by row

      System.out.println( "\nValues in array2 by row are" );
      outputArray( array2 ); // displays array2 by row
   } // end main

   // output rows and columns of a two-dimensional array
   public static void outputArray( int array[][] )
   {
      // loop through array's rows
      for ( int row = 0; row < array.length; row++ )
      {
         // loop through columns of current row
         for ( int column = 0; column < array[ row ].length; column++ )
            System.out.printf( "%d  ", array[ row ][ column ] );

         System.out.println(); // start new line of output
      } // end outer for
   } // end method outputArray
} // end class InitArray
```

Method 1 (function)

Method 2 (function)

Multidimensional array use example

```
Values in array1 by row are
1  2  3
4  5  6

Values in array2 by row are
1  2
3
4  5  6
```

Class Exercise

1. Do the Eclipse HelloWord!! or NetBeans HelloWord!!
2. Modify the "Multidimensional array use example "code in order to:
   - print the main diagonal of the next two multidimensional arrays

1	2	3
4	5	6
7	8	9

Numbers array

a	b	c	d
e	f	g	h
i	j	k	l
m	n	o	p

Letters array

References

- [Deitel] H.M. Deitel and P.J. Deitel, *Java How to Program: Early Objects Version*, Prentice Hall, 2009.

- **Oracle – Java Lesson: Language Basics**
  - http://download.oracle.com/javase/tutorial/java/nutsandbolts/index.html