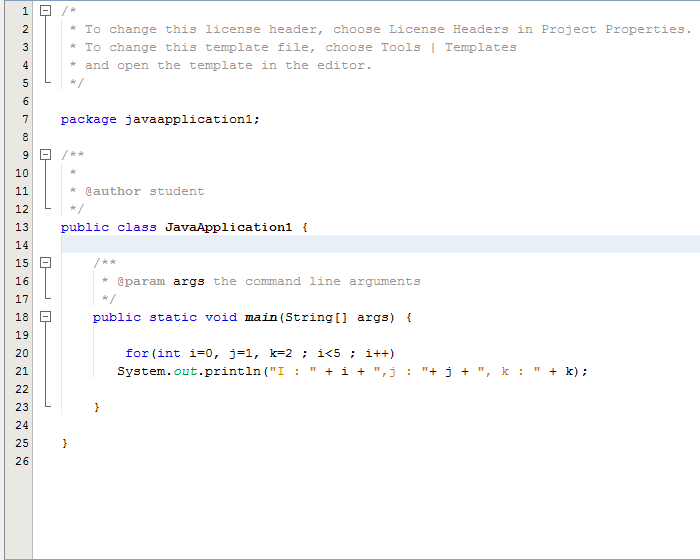
Pages 91-92 and Page 113 ***Java Programming A Comprehensive Introduction***

**Section 1: Define / Answer**

How does a **for** loop with multiple loop control variables operate?



Page 103, ***Java Programming A Comprehensive Introduction***

Explain the difference between a **for** loop and a **while loop**-

While loop is usually used when you need to repeat something until a given is true

For loop is usually used when you need to iterate a given number of times:

What is the basic difference between a **do-while** loop and [**or/while]** loops?

Do while would run the looping before the comparison while the while loop would run the comparison before looping

How do **break** statements work in relation to **for, while,** and **do-while** loops?

**Break would exit the loop**

Describe how an infinite **for** loop operates.

Once it operate, it would keep looping without stop

Data Structure:  **data structure** is a particular way of organizing **data** in a computer so that it can be used efficiently.

Dynamic Memory- **ynamic** random-access **memory** (DRAM) is a type of random-access**memory** that stores each bit of data in a separate capacitor within an integrated circuit.

1. Static Memory- **Static** random-access **memory** (SRAM or**static** RAM) is a type of semiconductor**memory** that uses bistable latching circuitry to store each bit. The term **static** differentiates it from dynamic RAM (DRAM) which must be periodically refreshed.
2. ArrayList- An **ArrayList** is a dynamic data structure,**meaning** items can be added and removed from the list. A normal array in java is a static data structure, because you stuck with the initial size of your array. To set up an **ArrayList**, you first have to import the package from the java.util library:

**Complete Array List Table**

|  |  |
| --- | --- |
| **Operation** | **Syntax** |
| Create or declare a list | ArrayList *list* **= new** ArrayList(); |
| Add string or object in to the list | list.add(“any object”); |
| Access to the indexlocation | list.get(indexlocation); |
| Remove an indexlocation in the list | list.remove(indexlocation); |
| Remove an object or string in the list | list.remove(object); |
| Clear the list | list.clear(); |
| Display the array size | list.size(); |
| assign that indexlocation to an object | list.set(index, “new object”); |
| Add that indexlocation to an objcet | list.add(index, “new object”); |

Pg. 577**,** Java Programming *A comprehensive Introduction*

[**http://www.oracle.com/technetwork/java/javase/documentation/index-137868.html#format**](http://www.oracle.com/technetwork/java/javase/documentation/index-137868.html#format)(Detailed explanation of Java documentation)

[**http://www.tutorialspoint.com/java/java\_documentation.htm**](http://www.tutorialspoint.com/java/java_documentation.htm)

[**http://www.liferay.com/community/wiki/-/wiki/Main/Javadoc+Guidelines#section-Javadoc+Guidelines-Class+Comments**](http://www.liferay.com/community/wiki/-/wiki/Main/Javadoc+Guidelines#section-Javadoc+Guidelines-Class+Comments)

Internal Documentation- the notes on how and why various parts of code operate is included within the [source code](https://en.wikipedia.org/wiki/Source_code) as comments. It is often combined with meaningful [variable](https://en.wikipedia.org/wiki/Variable_(programming)) names with the intention of providing potential future programmers a means of understanding the workings of the code.

Internal documentation would be comments and remarks made by the programmer in the form of line comments and boiler plates.

External Documentation- External documentation would be things like flow charts, UML diagrams, requirements documents, design documents etc.

Java Doc Tags-  is a [documentation generator](https://en.wikipedia.org/wiki/Documentation_generator) from [Oracle Corporation](https://en.wikipedia.org/wiki/Oracle_Corporation) for generating [API](https://en.wikipedia.org/wiki/Application_programming_interface" \o "Application programming interface)documentation in [HTML](https://en.wikipedia.org/wiki/HTML) format from [Java](https://en.wikipedia.org/wiki/Java_(programming_language)) source code. The HTML format is used to add the convenience of being able to [hyperlink](https://en.wikipedia.org/wiki/Hyperlink) related documents together.[[2]](https://en.wikipedia.org/wiki/Javadoc#cite_note-2)

**Javadoc tags (Examples)**

|  |  |  |
| --- | --- | --- |
| **Tag** | **Description** | **Syntax** |
| @author | Adds the author of a class. | @author name-text |
| {@code} | Displays text in code font without interpreting the text as HTML markup or nested javadoc tags. | {@code text} |
| {@docRoot} | Represents the relative path to the generated document's root directory from any generated page | {@docRoot} |
| @deprecated | Adds a comment indicating that this API should no longer be used. | @deprecated deprecated-text |
| @exception | Adds a **Throws** subheading to the generated documentation, with the class-name and description text. | @exception class-name description |
| {@inheritDoc} | Inherits a comment from the **nearest** inheritable class or implementable interface | Inherits a comment from the immediate surperclass. |
| {@link} | Inserts an in-line link with visible text label that points to the documentation for the specified package, class or member name of a referenced class. T | {@link package.class#member label} |
| {@linkplain} | Identical to {@link}, except the link's label is displayed in plain text than code font. | {@linkplain package.class#member label} |
| @param | Adds a parameter with the specified parameter-name followed by the specified description to the "Parameters" section. | @param parameter-name description |
| @return | Adds a "Returns" section with the description text. | @return description |
| @see | Adds a "See Also" heading with a link or text entry that points to reference. | @see reference |
| @serial | Used in the doc comment for a default serializable field. | @serial field-description | include | exclude |
| @serialData | Documents the data written by the writeObject( ) or writeExternal( ) methods | @serialData data-description |
| @serialField | Documents an ObjectStreamField component. | @serialField field-name field-type field-description |
| @since | Adds a "Since" heading with the specified since-text to the generated documentation. | @since release |
| @throws | The @throws and @exception tags are synonyms. | @throws class-name description |
| {@value} | When {@value} is used in the doc comment of a static field, it displays the value of that constant: | {@value package.class#field} |
| @version | Adds a "Version" subheading with the specified version-text to the generated docs when the -version option is used. | @version version-text |

**Programming Assignment**

Task 1- Create a multiplication table for numbers 1 – 9,and all the multiples up to 9.

Use a Nested **for** Loop to print the table.

**Formatting is key for this assignment**, your output should exactly match the output below.

Expected Output

Multiplication Table

1 2 3 4 5 6 7 8 9

---------------------------------------------------------

1| 1 2 3 4 5 6 7 8 9

2| 2 4 6 8 10 12 14 16 18

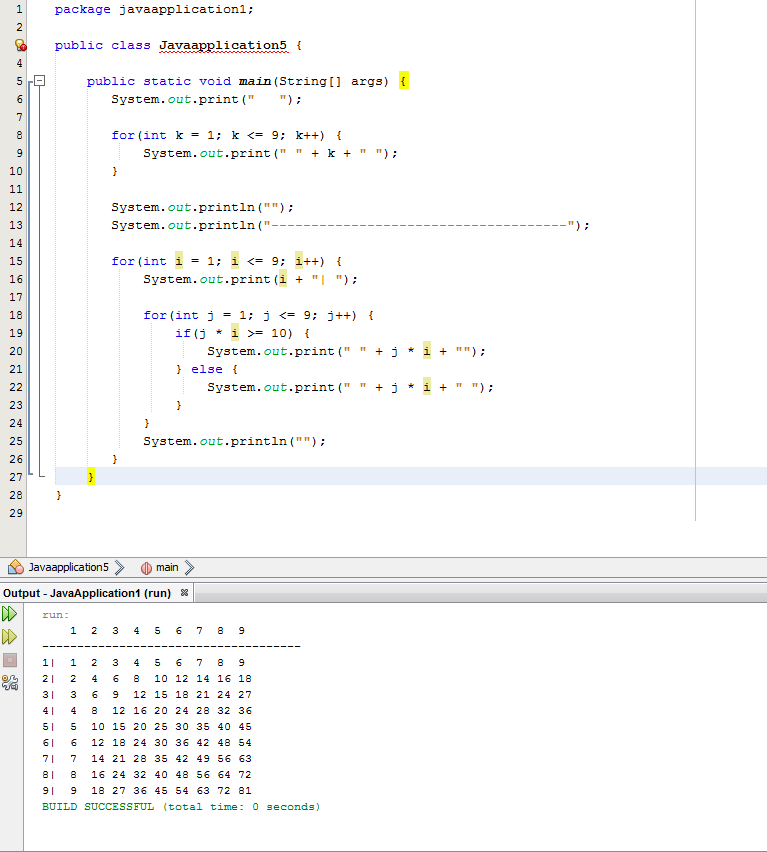
3| 3 6 9 12 15 18 21 24 27

4| 4 8 12 16 20 24 28 32 36

5| 5 10 15 20 25 30 35 40 45

1| 6 12 18 24 30 36 42 48 54

Etc...



Task 2-

**Pg. 115 #15** Java Programming *A Comprehensive Guide*

There are 6 different programs + output required for this assignment.

Please Have 6 Different snipping photos with programs and outputs.

Hint\*\* The programs should not be very long for each answer.

Use differentloops to **print the odd / negative numbers 1 to 101.** All programs will print the same output in the same order.

1. Using a **for** loop that increments the loop control variable by 2 each iteration
2. Using a **for** loop whose loop control variable goes from 0 to 50.
3. Using a **for** loop whose loop control variable goes from 100 down to 0.
4. Using an infinite **for** loop with no conditional expression and exiting the loop with a **break** statement.
5. Using a **while** loop.
6. Using a **do-while** loop.

There should be 6 different Snipping photos. One photo for each program A – F.

-1

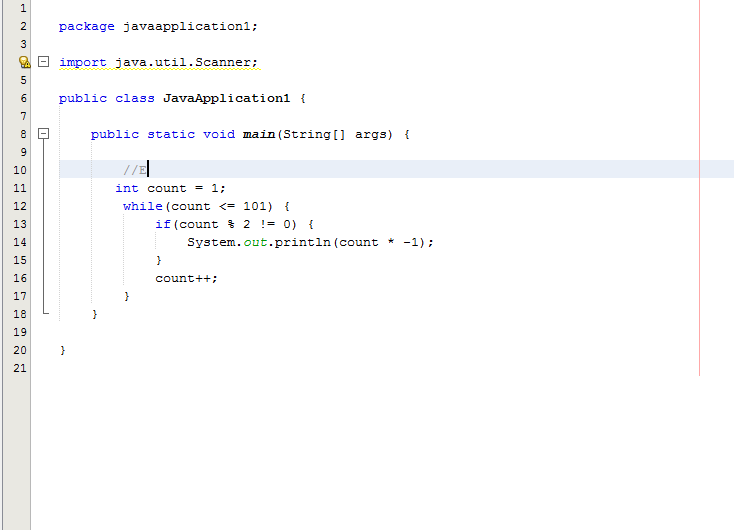
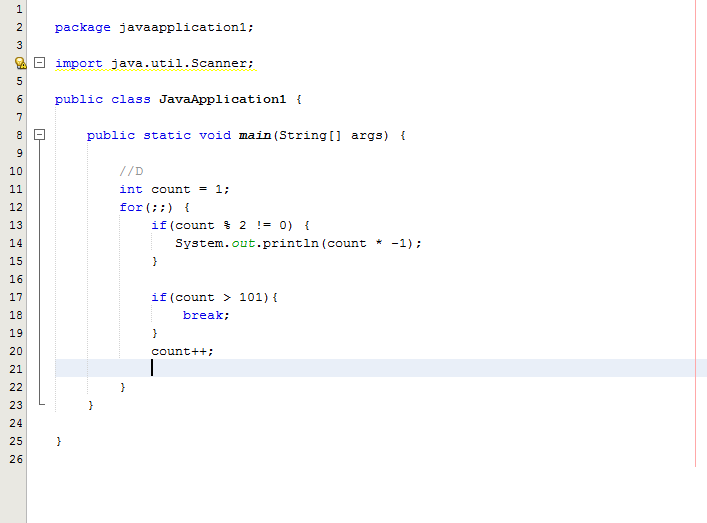
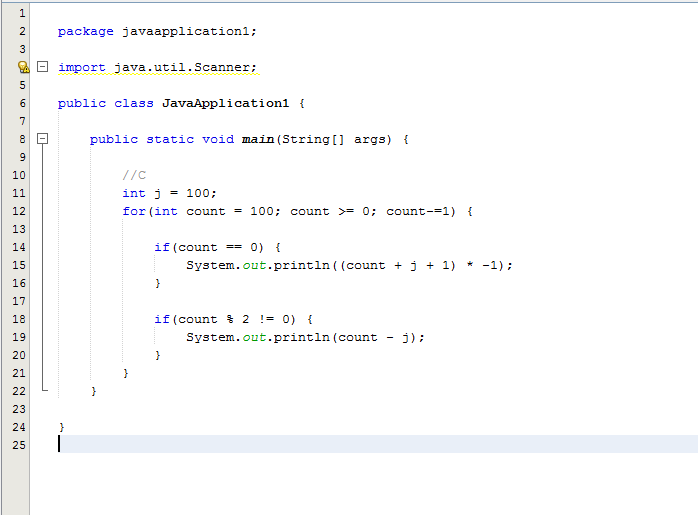
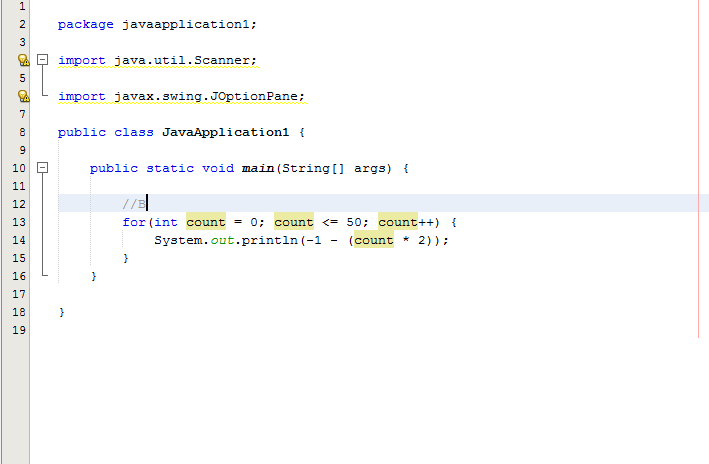
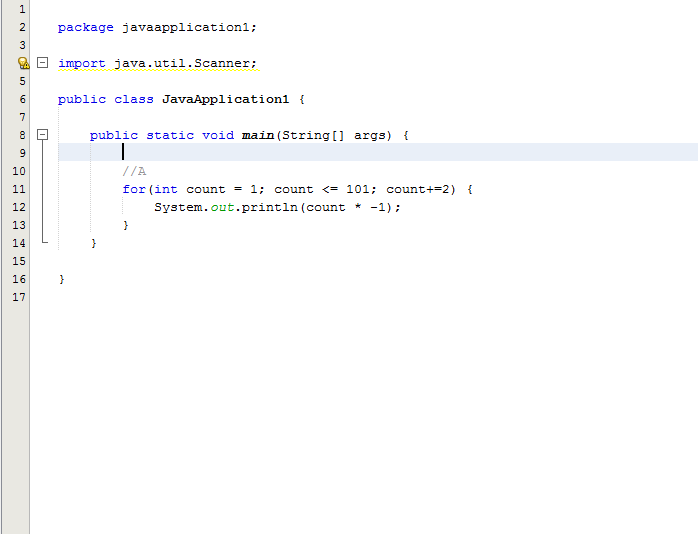
-3

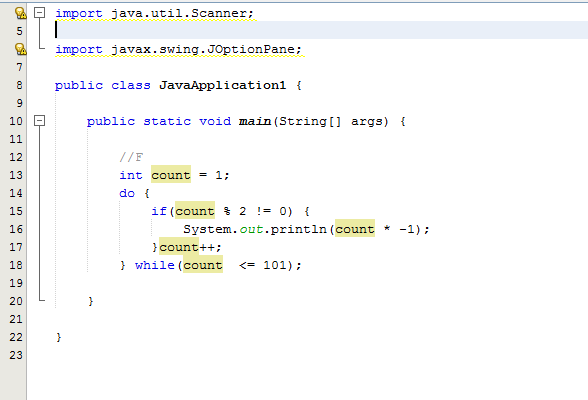
-5

-7

…

-101





Task 3-

Write a program that creates an integer Arraylist called **data** and then uses a **for** loop to a new **String** that displays the contents of the **data** array surrounded by braces and separated by commas. For example, if the **data** array is of length 4 and contains values 3,4,15, then the **String** should be “{3,4,1,5}” should be created and printed.

