

**ID: 14269**

**NAME :QAZIBILAL**

Q1. How many variables are being supported by java justify your answer with the help java coded example for each variable?

**Answer:**

There are three types of variables we use in java:

Local variable

Instance variable

Class/Static variable

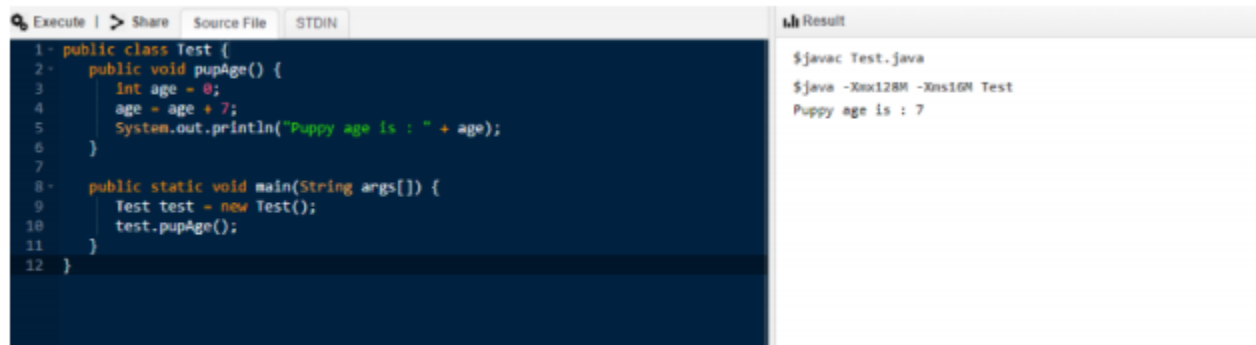
### **1) Local Variables**

- Local variables are declared in methods, constructors, or blocks.
- Local variables are created when the method, constructor or block is entered and the variable will be destroyed once it exits the method, constructor, or block.
- Access modifiers cannot be used for local variables.
- Local variables are visible only within the declared method, constructor, or block.
- Local variables are implemented at stack level internally.
- There is no default value for local variables, so local variables should be declared and an initial value should be assigned before the first use.

### **EXAMPLE:**

- Here, age is a local variable. This is defined inside pupAge() method and its scope is limited to only this method.

## CODE:



The image shows a screenshot of a code editor with two panes. The left pane displays the source code for a Java class named 'Test'. The code includes a 'pupAge()' method that increments an 'age' variable by 7 and prints it, and a 'main()' method that creates an instance of 'Test' and calls 'pupAge()'. The right pane shows the execution result, which includes the compilation command '\$javac Test.java', the execution command '\$java -Xmx128M -Xms16M Test', and the output 'Puppy age is : 7'.

```
1 public class Test {
2     public void pupAge() {
3         int age = 0;
4         age = age + 7;
5         System.out.println("Puppy age is : " + age);
6     }
7
8     public static void main(String args[]) {
9         Test test = new Test();
10        test.pupAge();
11    }
12 }
```

Result

```
$javac Test.java
$java -Xmx128M -Xms16M Test
Puppy age is : 7
```

## 2) Instance Variables

- Instance variables are declared in a class, but outside a method, constructor or any block.
- When a space is allocated for an object in the heap, a slot for each instance variable value is created.
- Instance variables are created when an object is created with the use of the keyword 'new' and destroyed when the object is destroyed.
- Instance variables hold values that must be referenced by more than one method, constructor or block, or essential parts of an object's state that must be present throughout the class.
- Instance variables can be declared in class level before or after use.
- Access modifiers can be given for instance variables.
- The instance variables are visible for all methods, constructors and block in the class. Normally, it is recommended to make these variables private (access level). However, visibility for subclasses can be given for these variables with the use of access modifiers.

- Instance variables have default values. For numbers, the default value is 0, for Booleans it is false, and for object references it is null. Values can be assigned during the declaration or within the constructor.
- Instance variables can be accessed directly by calling the variable name inside the class. However, within static methods (when instance variables are given accessibility), they should be called using the fully qualified name. `ObjectReference.VariableName`.

## CODE:

```

1  import java.io.*;
2  public class Employee {
3
4      // this instance variable is visible for any child class.
5      public String name;
6
7      // salary variable is visible in Employee class only.
8      private double salary;
9
10     // The name variable is assigned in the constructor.
11     public Employee (String empName) {
12         name = empName;
13     }
14
15     // The salary variable is assigned a value.
16     public void setSalary(double empSal) {
17         salary = empSal;
18     }
19
20     // This method prints the employee details.
21     public void printEmp() {
22         System.out.println("name : " + name );
23         System.out.println("salary : " + salary);
24     }
25
26     public static void main(String args[]) {
27         Employee empOne = new Employee("Ransika");
28         empOne.setSalary(1000);
29         empOne.printEmp();
30     }
31 }

```

Result

```

$javac Employee.java
$java -Xmx128M -Xms16M Employee
name : Ransika
salary :1000.0

```

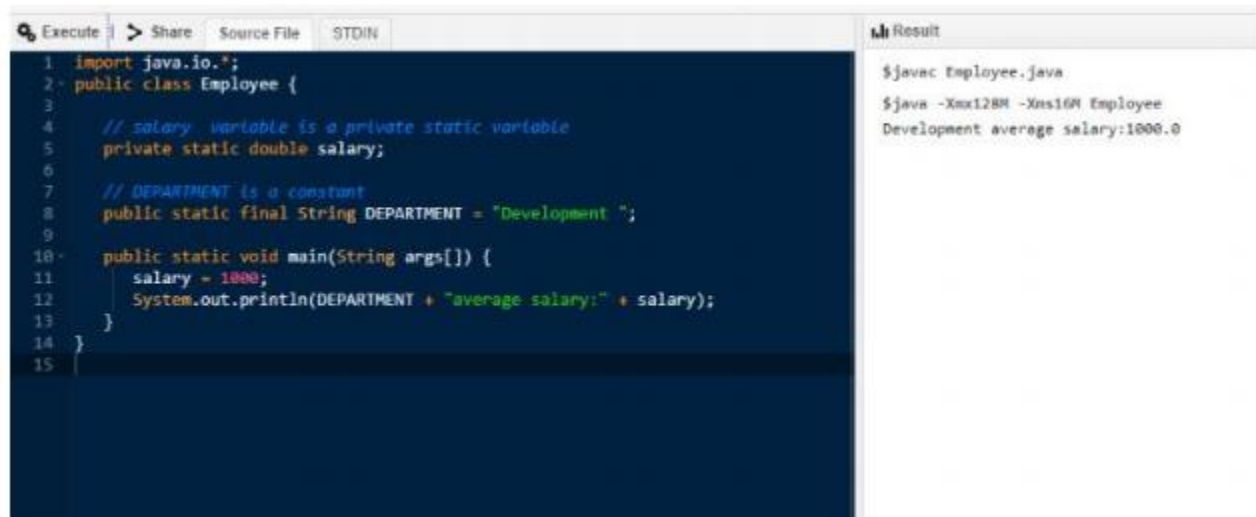
Activate  
Go to Settings

## 3) Class/Static Variables

- Class variables also known as static variables are declared with the static keyword in a class, but outside a method, constructor or a block.
- There would only be one copy of each class variable per class, regardless of how many objects are created from it.
- Static variables are rarely used other than being declared as constants. Constants are variables that are declared as public/private, final, and static. Constant variables never change from their initial value.

- Static variables are stored in the static memory. It is rare to use static variables other than declared final and used as either public or private constants.
- Static variables are created when the program starts and destroyed when the program stops.
- Visibility is similar to instance variables. However, most static variables are declared public since they must be available for users of the class.
- Default values are same as instance variables. For numbers, the default value is 0; for Booleans, it is false; and for object references, it is null. Values can be assigned during the declaration or within the constructor. Additionally, values can be assigned in special static initializer blocks.
- Static variables can be accessed by calling with the class name `ClassName.VariableName`.
- When declaring class variables as public static final, then variable names (constants) are all in upper case. If the static variables are not public and final, the naming syntax is the same as instance and local variables.

## CODE:



```
1 import java.io.*;
2 public class Employee {
3
4     // salary variable is a private static variable
5     private static double salary;
6
7     // DEPARTMENT is a constant
8     public static final String DEPARTMENT = "Development ";
9
10    public static void main(String args[]) {
11        salary = 1000;
12        System.out.println(DEPARTMENT + "average salary:" + salary);
13    }
14 }
15
```

Result

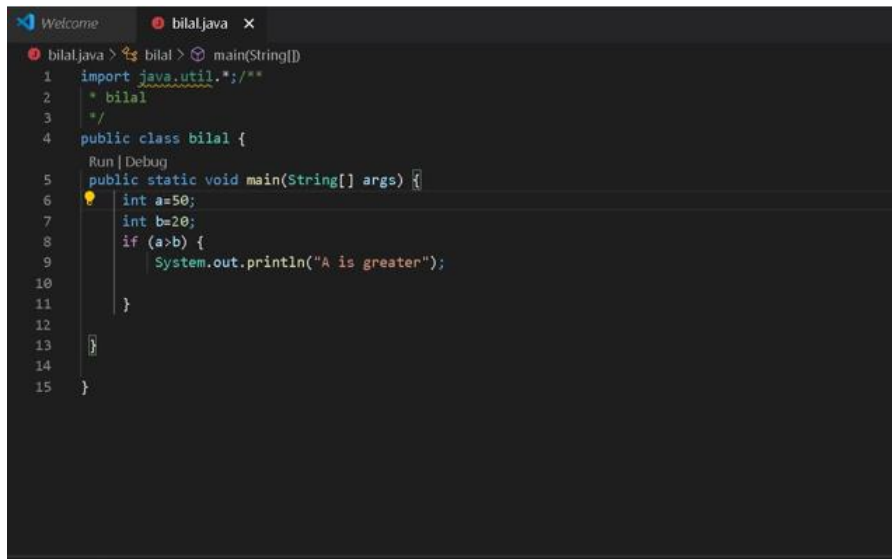
```
$javac Employee.java
$java -Xmx128M -Xms16M Employee
Development average salary:1000.0
```

Q2. Why “If” is used in java justify your answer with the help java coded example and explain in detail?

Answer:

We Use if statement to specify a block of Java code to be executed if a condition is true.

Code:

A screenshot of an IDE window titled 'bilaljava'. The code is as follows:

```
bilaljava > bilal > main(String[])
1 import java.util.*;/**
2  * bilal
3  */
4 public class bilal {
5     Run | Debug
6     public static void main(String[] args) {
7         int a=50;
8         int b=20;
9         if (a>b) {
10            System.out.println("A is greater");
11        }
12    }
13 }
14 }
15 }
```

**Explanation :**

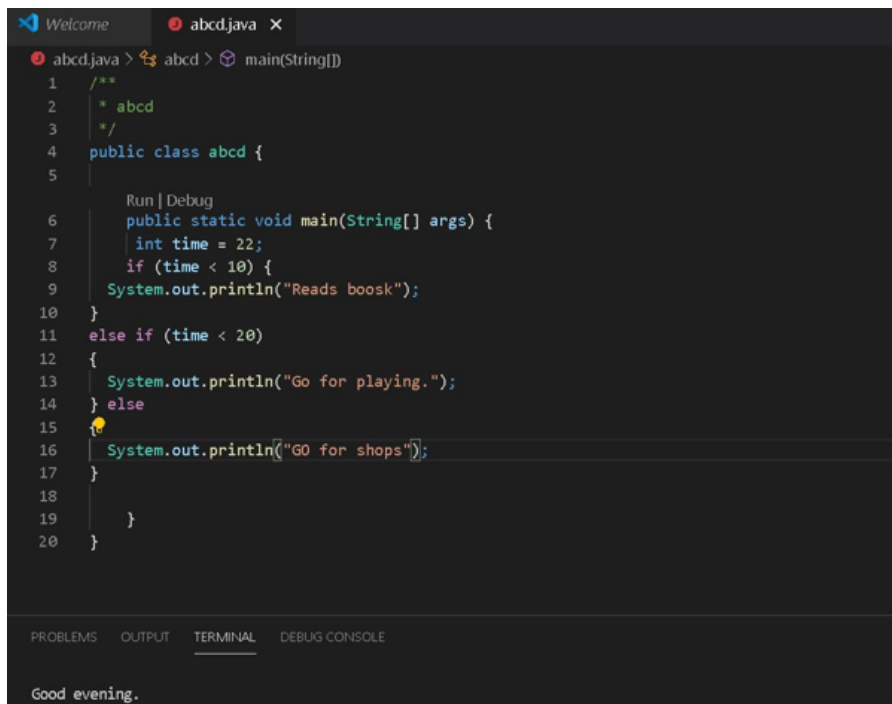
In this program I am using a simple if statement, I have declare two int values ‘a,b’ and then I am using if statement which is if(a>b) which means if a is greater then b then print “A is greater”.

Q3. Why “if else if” is used in java justify your answer with the help java coded example and explain in detail?

Answer

Use the else if statement to specify a new condition if the first condition is false

Code:



```
1  /**
2   * abcd
3   */
4  public class abcd {
5
6      Run | Debug
7      public static void main(String[] args) {
8          int time = 22;
9          if (time < 10) {
10             System.out.println("Reads book");
11         }
12         else if (time < 20)
13         {
14             System.out.println("Go for playing.");
15         }
16         else
17         {
18             System.out.println("GO for shops");
19         }
20     }
}
```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

Good evening.

## Explanation :

in this program I am using “if else if”. I have declare an integer name “time” and I have use if else if statement for it. If the time is less then “10 ” it will display “read book”,if this the first condition goes wrong the compiler will come to the second condition which is “else if (time <20)” which means if time is less then 20 it will display “go for playing”,if these both condition goes wrong it will display “go for shops”.

---

Q4. What are loops, why they are used in java and how many types of loops are being supported by java explain in detail?

**Answer**

### LOOP:

In computer science, a loop is a programming structure that repeats a sequence of instructions until a specific condition is met. Programmers use loops to cycle through values, add sums of numbers, repeat functions, and many other things.

## WHY THEY ARE USED :

Looping in programming languages is a feature which facilitates the execution of a set of instructions/functions repeatedly while some condition evaluates to true.

Java provides three ways for executing the loops. While all the ways provide similar basic functionality, they differ in their syntax and condition checking time.

## TYPES OF LOOP:

There are 3 types of loop ,following are;

1. While loop
2. For loop
3. Do while loop

## WHILE LOOP:

A while loop is a control flow statement that allows code to be executed repeatedly based on a given Boolean condition. The while loop can Be thought of as a repeating if statement.

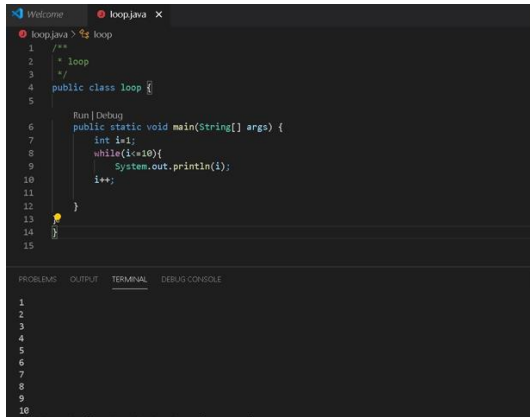
Syntax :

```
while (boolean condition)
{
    loop statements...
}
```

While loop starts with the checking of condition. If it evaluated to true, then the loop body statements are executed otherwise first statement following the loop is executed. For this reason it is also called Entry control loop .Once the condition is evaluated to true, the statements in the loop body are executed. Normally the statements contain an update value for the variable being processed for the next iteration.

When the condition becomes false, the loop terminates which marks the end of its life cycle.

**Code:**



```
1 /**
2  * loop
3  */
4  public class loop {
5
6      Run [Debug]
7      public static void main(String[] args) {
8          int i=1;
9          while(i<=10){
10             System.out.println(i);
11             i++;
12         }
13     }
14 }
15
```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

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

## Explanation :

In this program I am using while loop .i have declare an integer name i=1 and then run a while loop which is “while (i<=10)” .it will print counting from 1 to 10.

---

## FOR LOOP:

for loop provides a concise way of writing the loop structure. Unlike a while loop, a for statement consumes the initialization, condition and increment/decrement in one line thereby providing a shorter, easy to debug structure of looping.

## Syntax:

for (initialization condition; testing condition;

increment/decrement)

{

statement(s)

}

Initialization condition: Here, we initialize the variable in use. It marks the start of a for loop. An already declared variable can be used or a variable can be declared, local to loop only.



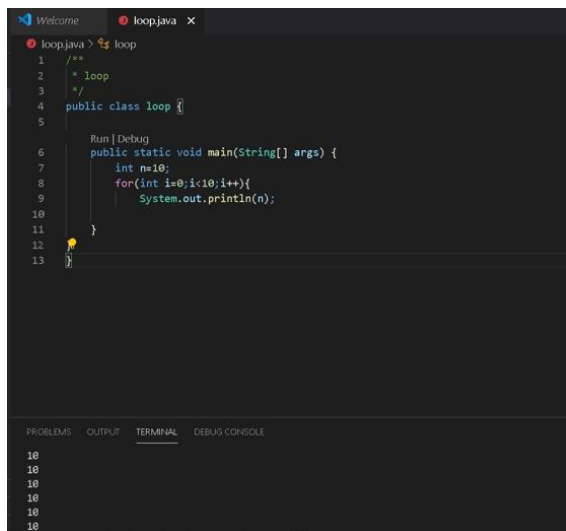
**Testing Condition:** It is used for testing the exit condition for a loop. It must return a boolean value. It is also an Entry Control Loop as the condition is checked prior to the execution of the loop statements.

**Statement execution:** Once the condition is evaluated to true, the statements in the loop body are executed.

**Increment/ Decrement:** It is used for updating the variable for next iteration.

**Loop termination:**When the condition becomes false, the loop terminates marking the end of its life cycle.

### Code:



```
1  /**
2   * loop
3   */
4  public class loop {
5
6      Run [Debug]
7      public static void main(String[] args) {
8          int n=10;
9          for(int i=0;i<10;i++){
10             System.out.println(n);
11         }
12     }
13 }
```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

10  
10  
10  
10  
10  
10

### Explanation :

In this program I am using for loop .i have declare an integer name n=10 and then run a for loop which is “for (int i=0;i<10;i++)” .it will print 10 ten times .

---

### DO WHILE LOOP:

do while loop is similar to while loop with only difference that it checks for condition after executing the statements, and therefore is an example of Exit Control Loop.

Syntax:

do

```
{  
    statements..  
}
```

while (condition);

do while loop starts with the execution of the statement(s). There is no checking of any condition for the first time .After the execution of the statements, and update of the variable value, the condition is checked for true or false value. If it is evaluated to true, next iteration of loop starts.

When the condition becomes false, the loop terminates which marks the end of its life cycle. It is important to note that the do-while loop will execute its statements atleast once before any condition is checked, and therefore is an example of exit control loop.

---

Q5. Write 3's table in decremented form in java which takes input from user write java coded program and explain in detail?

```
Welcome | table.java X
table.java > table > main(String[])
1 import java.util.Scanner;
2 /**
3  * table
4  */
5 public class table {
6
7     Run | Debug
8     public static void main(String[] args) {
9         Scanner s = new Scanner(System.in);
10        System.out.print("Enter number:");
11        int n=s.nextInt();
12        for(int i=10; i >= 1; i--)
13        {
14            System.out.println(n+" * "+i+" = "+n*i);
15        }
16    }

```

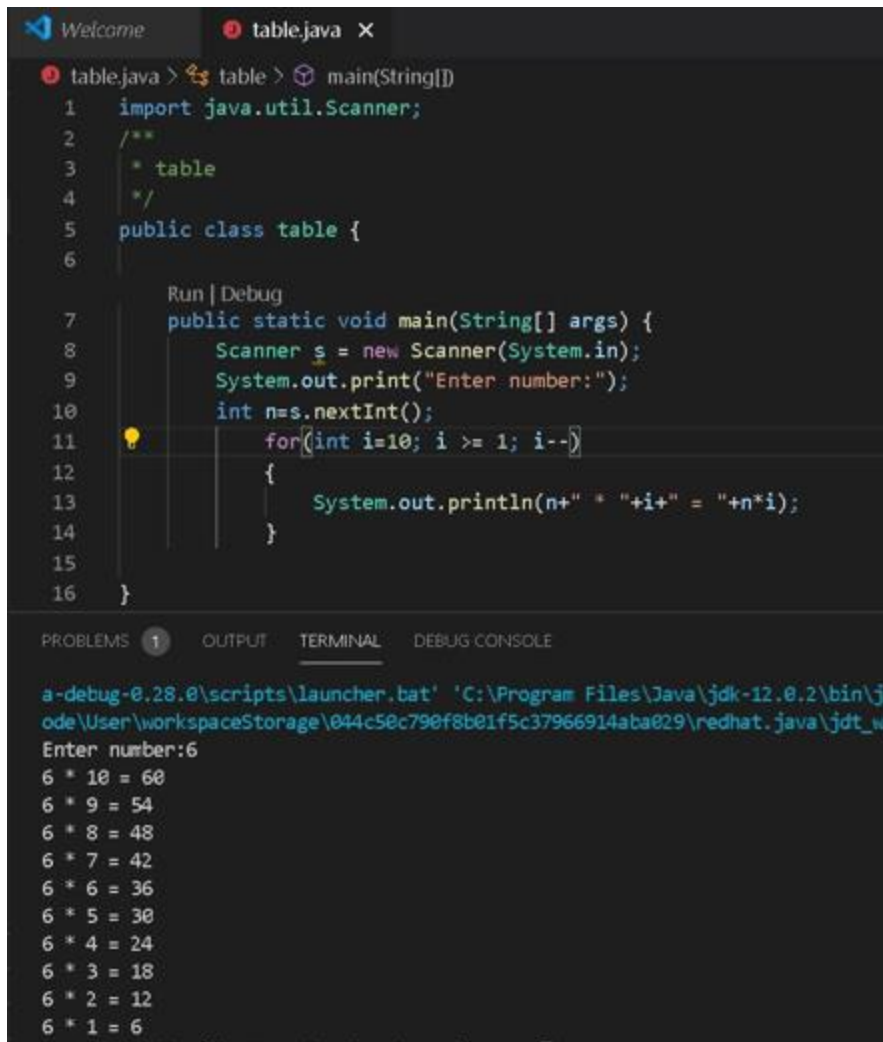
PROBLEMS 1 | OUTPUT | TERMINAL | DEBUG CONSOLE

```
a-debug-0.28.0\scripts\launcher.bat' 'C:\Program Files\Java\jdk-12.0.2\bin\java.exe' '-Dfile
ode\User\workspaceStorage\044c50c79ef8b01f5c37966914aba029\redhat.java\jdt_ws\question_5_f7
Enter number:10
10 * 10 = 100
10 * 9 = 90
10 * 8 = 80
10 * 7 = 70
10 * 6 = 60
10 * 5 = 50
10 * 4 = 40
10 * 3 = 30
10 * 2 = 20
10 * 1 = 10
```

```
table.java > table > main(String[])
1 import java.util.Scanner;
2 /**
3  * table
4  */
5 public class table {
6
7     Run | Debug
8     public static void main(String[] args) {
9         Scanner s = new Scanner(System.in);
10        System.out.print("Enter number:");
11        int n=s.nextInt();
12        for((int i=10; i >= 1; i--))
13        {
14            System.out.println(n+ " * "+i+ " = "+n*i);
15        }
16    }
```

PROBLEMS 1 OUTPUT TERMINAL DEBUG CONSOLE

```
a-debug-0.28.0\scripts\launcher.bat' 'C:\Program Files\Java\jdk-12.0.2\bin\java.exe' '-Dfile.encoding=UTF-8' 'C:\Users\User\workspaceStorage\044c50c790f8b01f5c37966914aba029\redhat.java\jdt_ws\question_5_f7da850'
Enter number:3
3 * 10 = 30
3 * 9 = 27
3 * 8 = 24
3 * 7 = 21
3 * 6 = 18
3 * 5 = 15
3 * 4 = 12
3 * 3 = 9
3 * 2 = 6
3 * 1 = 3
```



```
table.java > table > main(String[])
1 import java.util.Scanner;
2 /**
3  * table
4  */
5 public class table {
6
7     public static void main(String[] args) {
8         Scanner s = new Scanner(System.in);
9         System.out.print("Enter number:");
10        int n=s.nextInt();
11        for(int i=10; i >= 1; i--)
12        {
13            System.out.println(n+" * "+i+" = "+n*i);
14        }
15    }
16 }
```

PROBLEMS 1 OUTPUT TERMINAL DEBUG CONSOLE

```
a-debug-0.28.0\scripts\launcher.bat' 'C:\Program Files\Java\jdk-12.0.2\bin\java.exe' -Djava.class.path=C:\Program Files\Java\jdk-12.0.2\bin\java\jdt_ws\redhat.java\jdt_ws\
Enter number:6
6 * 10 = 60
6 * 9 = 54
6 * 8 = 48
6 * 7 = 42
6 * 6 = 36
6 * 5 = 30
6 * 4 = 24
6 * 3 = 18
6 * 2 = 12
6 * 1 = 6
```

## Explanation :

---

In this program we take input from user.

The number n will be that user input number. The i will be counter. The i will start from 10.and ends at 1.

Here the counter will go in reverse order. first the number n will be multiply from i values.(N\*i).

After that the number n will be same and the( i-1.10-1=9) .and the loop run until the value of i comes to 1.

---

