**RECURSION**

**PROGRAMS**

PROGRAM 1

Write a program in java to accept a range from the user & display the Armstrong number till a range.

Sample input:

Enter a range: 1000

Sample output:

The Armstrong numbers are: 1 153 370 371 407

123

Algorithm

Step 1: Start

Step 2: Scanner class object declared

Step 3: A range is accepted from the user.

Step 4: An object armstrong-recursion is declared

Step 5: The following steps are to be executed from 1 to the range accepted.

Step 5.1: Each number from 1 to that range is assigned to a variable and is passed to a function sumofdigits for armstrong number checking.

Step 5.2: if the returned number is equal to the number passed then print the number is Armstrong number else print not a Armstrong number

FOR sumofdigits()

Step 1: Start.

Step 2: The number is received from the main function.

Step 3: Each digit of that number is extracted and the cube of that digit is calculated and is stored in a variable.

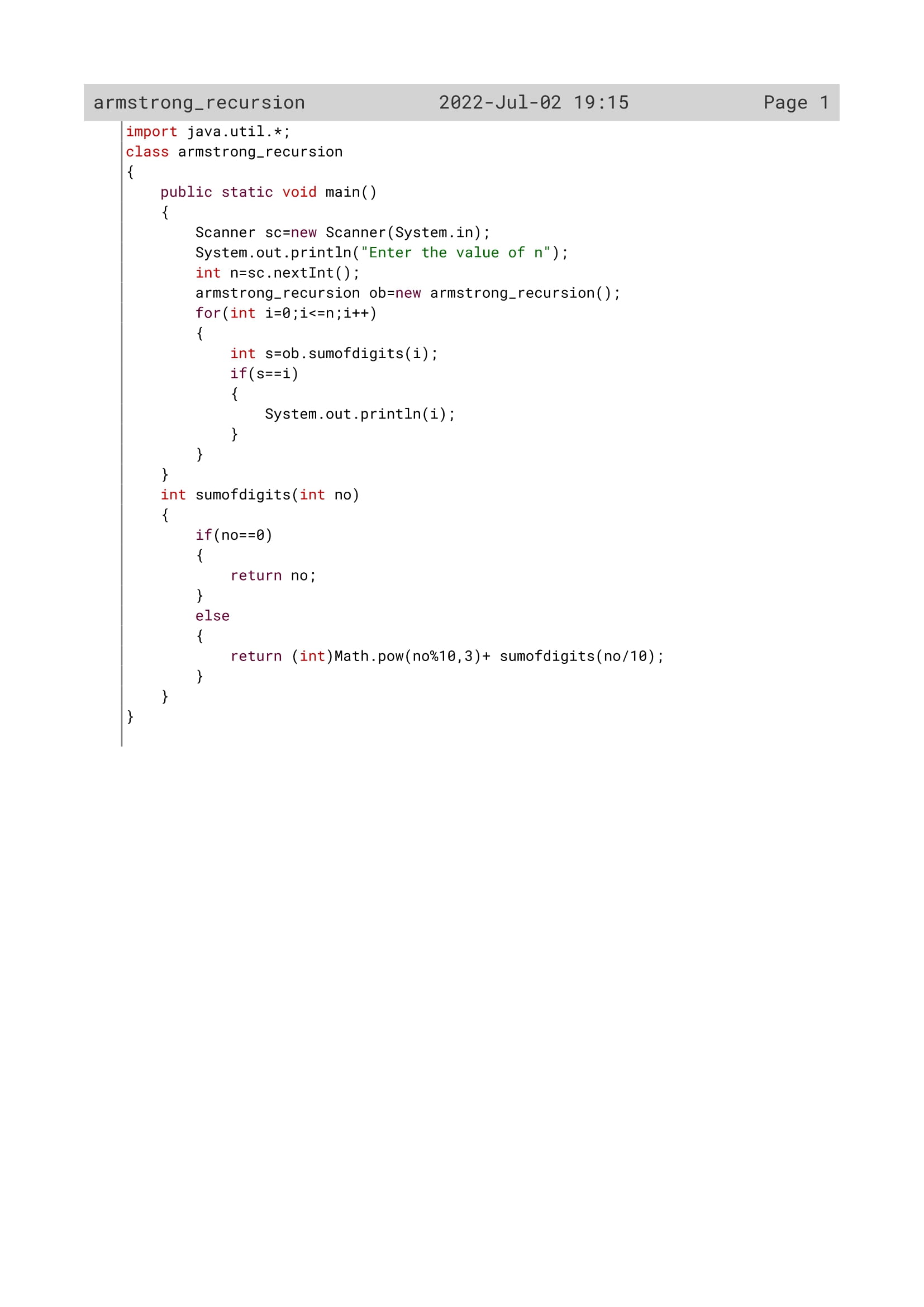
Step 4: the cubes of all the digits are added.

Step 5: if the original number becomes Zero then the sum of cubes of each digit is returned to main function.

Step 6: Stop

124

Source Code



125

Variable Description

|  |  |  |
| --- | --- | --- |
| Variable name | Data type | Description |
| n | Int | To accept a number from the user |
| no | Int | To act as parameter inside the method sumofdigits() |
| i | Int | Loop variable |
| s | Int | To calculate the sum of the digits |

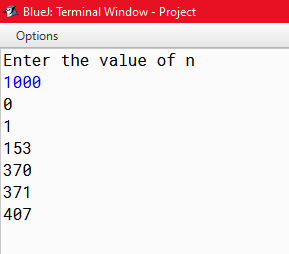
126

Function Description

|  |  |  |
| --- | --- | --- |
| Function name | Data type | description |
| sumofdigits() | Int | To calculate sum of digits |
| main() | void | To accept a number and check whether it is a Armstrong number or not. |

127

OUTPUT



128

PROGRAM 2

Write a program in java to accept a number from the user & check whether it’s a magic number or not.

For example: 289 = 2 + 8 + 9 = 19

19 = 1 + 9 = 10

10 = 1 + 0 = 1

289 is a magic number.

Sample input:

Enter a number: 55

Sample output: 55 is a magic number.

129

Algorithm

Step 1: Start

Step 2: Scanner class object declared

Step 3: A number is accepted from the user.

Step 4: An object *Magic* is declared.

Step 5: The following steps will be executed until the number stored in  *no* variable is greater than 9

Step 5.1: If the number is greater than 9 then the number is passed to *sumofdgt* function and the returned value is stored in *s* variable.

Step 5.2: again the value in *s* variable is initialized to *no* variable.

Step 6: if the eventual sum of the digits of the accepted number is one then print the Number is Magic number else print not a Magic number

Step 7: Stop.

For sumofdgt()

Step 1: Start

Step 2: the number received from main Function is initialised to *no* variable.

Step 3: repeat the following steps until the number is equal to zero

Step 3.1: Here the number is divided by 10 and the remainder obtained is added while the quotient is again passed to *sumofdgt()* Function.

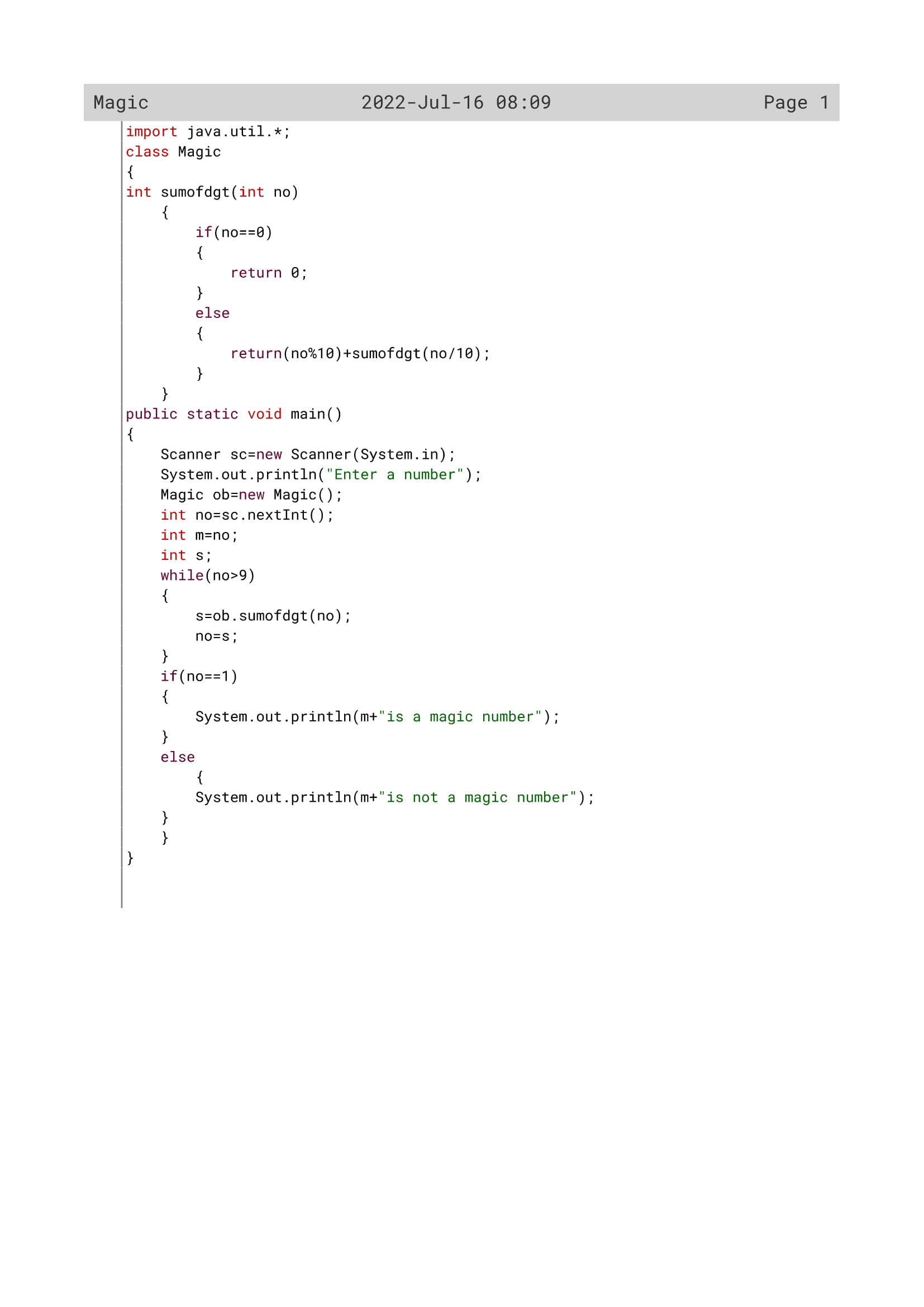
Step 3.2: Again the same steps will be followed and the remainders will be added to get The sum of digits

Step 4: when the number will become zero we will return the sum of digits to main function.

Step 5: Stop.

130

Source Code



131

Variable Description

|  |  |  |
| --- | --- | --- |
| Variable name | Data type | Description |
| no | int | To accept a number by the user |
| s | int | To send the number to the method sumofdgt() if it is greater than 9 |
| m | int | To store ‘no’s value |

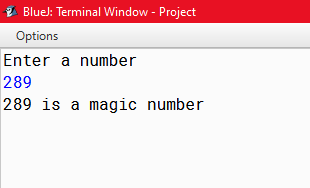
132

Function Description

|  |  |  |
| --- | --- | --- |
| Function name | Data type | description |
| sumofdgt() | Int | To calculate sum of digits |
| main() | void | To check whether the given number is a magic number or nor |

133

OUTPUT



134

PROGRAM 3

Write a program in java to accept a number from the user & check whether it’s a harshad number or not. A harshad number is an integer that is divisible by the sum of its digits. Sample input:

Enter a number: 81

Sample output:

81 is a harshad number.

135

Algorithm

Step 1: Start

Step 2: Scanner class object declared

Step 3: A number is accepted from the user.

Step 4: An object *Harshad* is declared.

Step 5: the number is passed to a function *sumofdgt* and the returned value is stored in *y* variable.

Step 6: The returned number is divided by the value stored in *y* variable and if the remainder is zero the print the number is Harshad number else print not a Harshad number

Step 7: Stop.

**ALGORITHM FOR sumofdgt ()**

Step 1: Start

Step 2: the number received from main Function is initialised to *no* variable.

Step 3: repeat the following steps until the number is equal to zero

Step 3.1: Here the number is divided by 10 and the remainder obtained is added while the quotient is again passed to *sumofdgt()* Function.

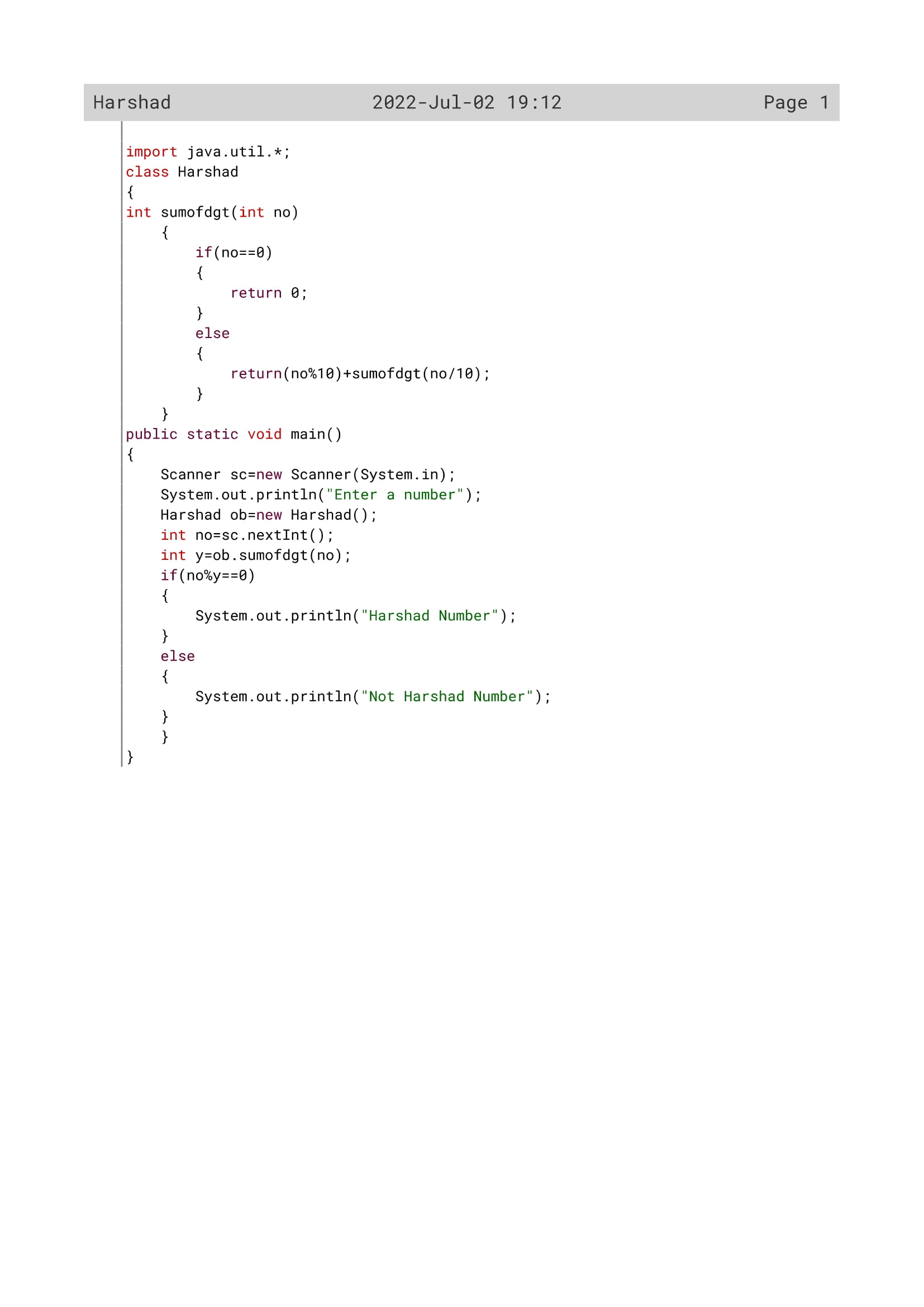
Step 3.2: Again the same steps will be followed And the remainders will be added to get The sum of digits

Step 4: when the number will become zero we will return the sum of digits to main function

Step 5: Stop

136

Source Code



137

Variable Description

|  |  |  |
| --- | --- | --- |
| Variable name | Data type | Description |
| no | int | To accept a number by the user |
| y | Int | To check whether the number is divisible by the sum of its digits or not |

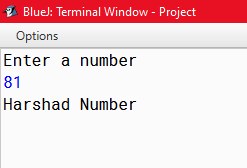
138

Function Description

|  |  |  |
| --- | --- | --- |
| Function name | Data type | description |
| sumofdgt() | Int | To calculate sum of digits |
| main() | void | To check whether the given number is a harshad number or not |

139

OUTPUT



140

PROGRAM 4

Write a program in java to accept a decimal number and convert it into its binary equivalent.

Sample input:

Enter a decimal number: 35

Sample output:

Binary equivalent is 10011

141

Algorithm

Step 1: Start

Step 2: Scanner class object declared

Step 3: A decimal number is accepted from the user.

Step 4: An object *binary* is declared

Step 5: the number and a String enclosed in double quotes are passed to a function *recbin* and the returned value is stored in *res* variable.

Step 6: The returned number is displayed as Binary equivalent of the accepted Number.

Step 7: Stop.

ALGORITHM FOR recbin()

Step 1: Start

Step 2: the number received from main function is initialised to *n* variable and the string is initialised to *s* variable

Step 3: repeat the following steps until the number is equal to zero

Step 3.1: Here the number is divided by 2 and the Remainder obtained is converted to String and is stored in *s* variable while The quotient is again passed to *recbin* Function

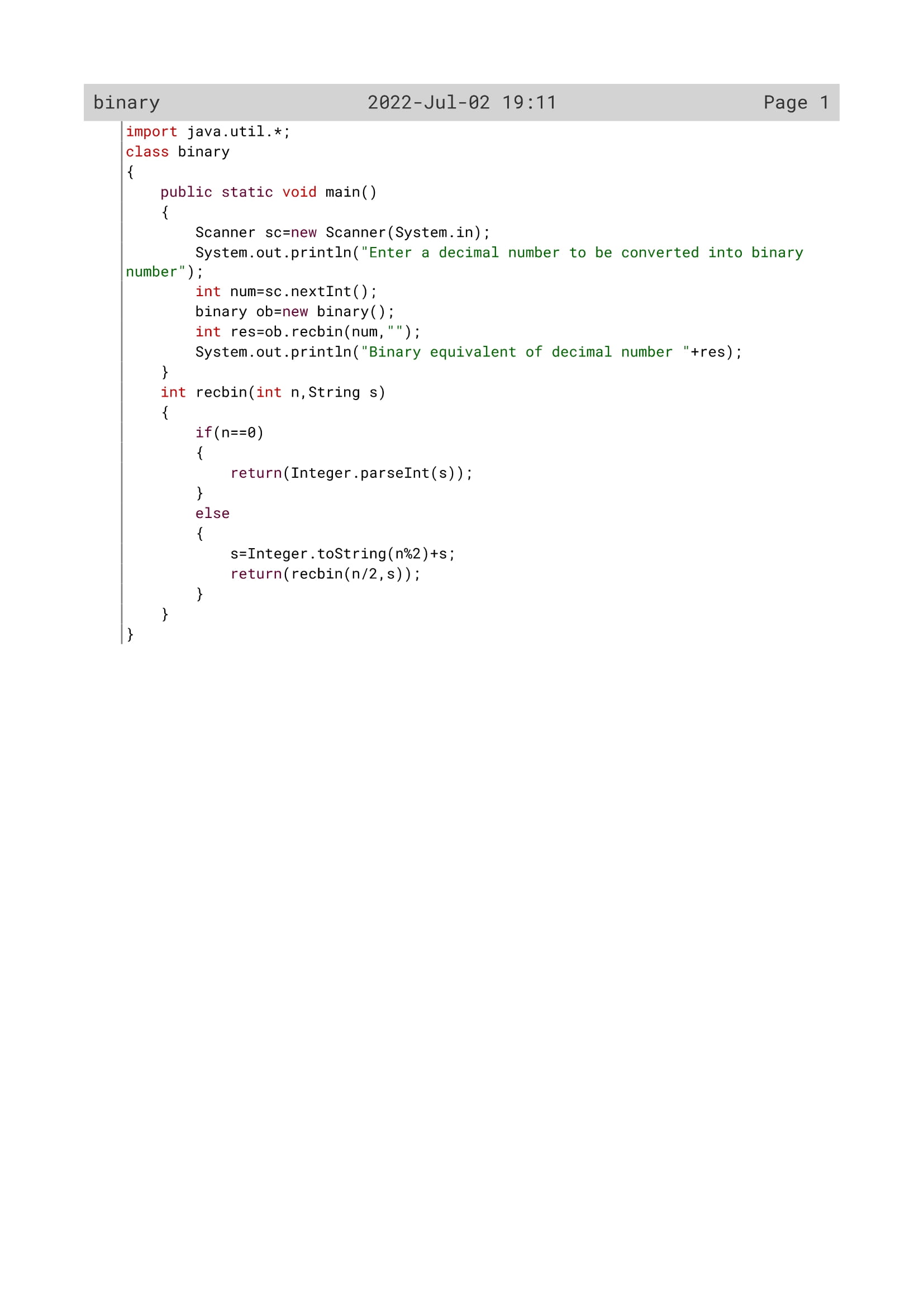
Step 4: Again the same steps will be followed The remainders will be converted to String and is added to s variable to get the binary equivalent

Step 5: when the number will become zero then return the binary equivalent.

Step 6: Stop

142

Source Code



143

Variable Description

|  |  |  |
| --- | --- | --- |
| Variable name | Data type | Description |
| Num | Int | To accept the number by the user |
| res | Int | The display the number to the user |
| n | Int | To act as parameter inside recbin() |
| s | string | To convert the number from integer to string |

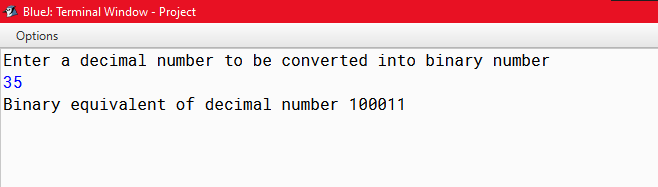
144

Function Description

|  |  |  |
| --- | --- | --- |
| Function name | Data type | description |
| recbin() | Int | To convert a decimal number to its binary equivalent |
| main() | void | To accept a decimal number and print its binary equivalent |

145

OUTPUT



146

PROGRAM 5

Write a program in java to accept a decimal number and convert it into its octal equivalent.

Sample input:

Enter a decimal number: 89

Sample output:

Octal equivalent is 131

147

Algorithm

Step 1: Start

Step 2: Scanner class object declared

Step 3: A decimal number is accepted from the user.

Step 4: An object *octal* is declared

Step 5: the number and a String enclosed in double quotes are passed to a function *recbin* and the returned value is stored in *res* variable.

Step 6: The returned number is displayed as Octal equivalent of the accepted Number.

Step 7: Stop.

ALGORITHM FOR recbin ()

Step 1: Start

Step 2: the number received from main function is initialised to *n* variable and the string is initialised to *s* variable

Step 3: repeat the following steps until the number is equal to zero

Step 3.1: Here the number is divided by 8 and the Remainder obtained is converted to String and is stored in *s* variable while The quotient is again passed to *recbin* Function

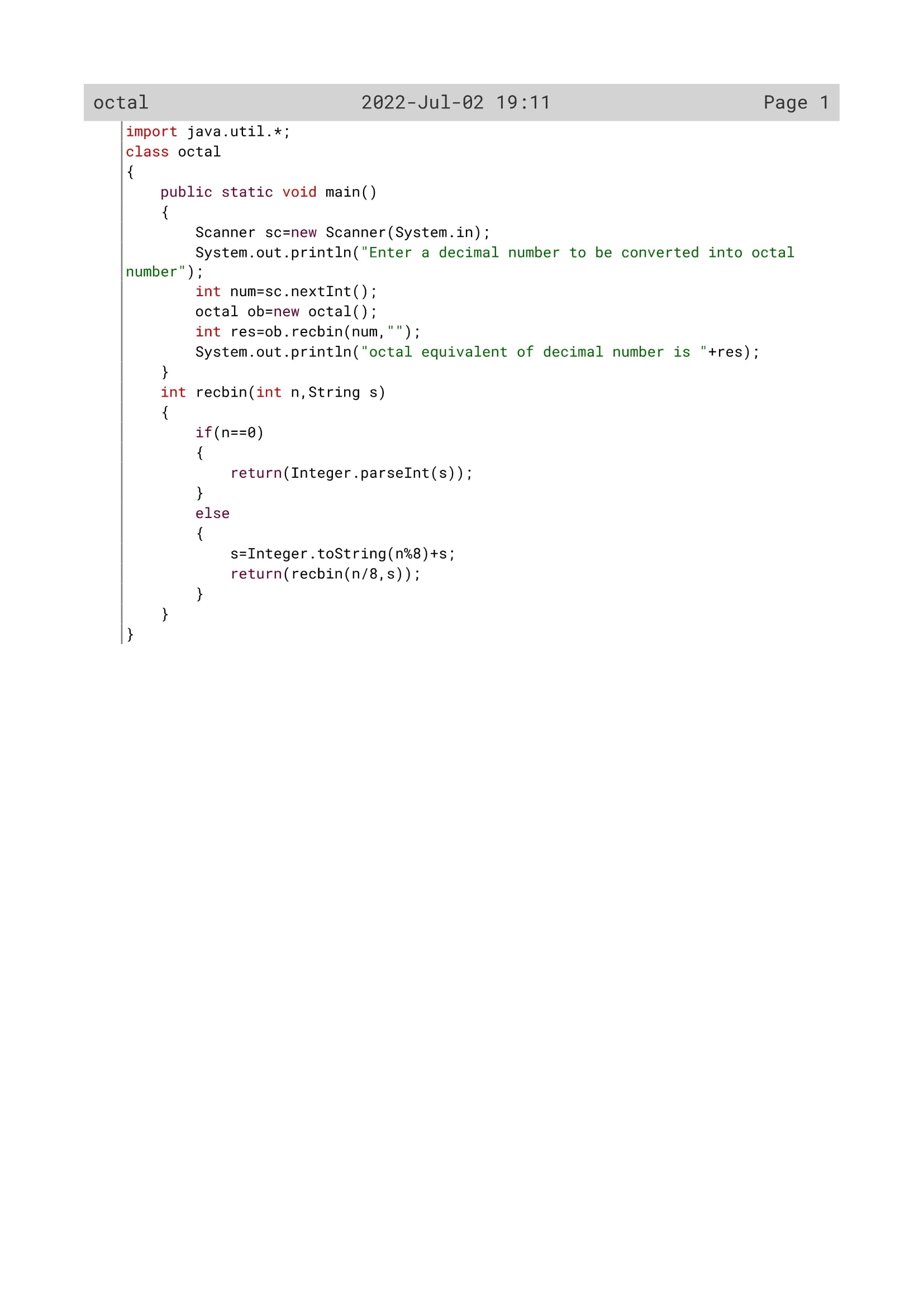
Step 4: Again the same steps will be followed The remainders will be converted to String and is added to s variable to get the binary equivalent

Step 5: when the number will become zero then return the binary equivalent.

Step 6: Stop

148

Source Code



149

Variable Description

|  |  |  |
| --- | --- | --- |
| Variable name | Data type | Description |
| Num | Int | To accept the number by the user |
| res | Int | The display the number to the user |
| n | Int | To act as parameter inside recbin() |
| s | string | To convert the number from integer to string |

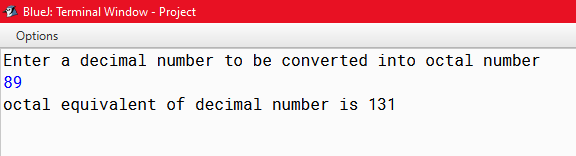
150

Function Description

|  |  |  |
| --- | --- | --- |
| Function name | Data type | description |
| recbin() | Int | To convert a decimal number to its octal equivalent |
| main() | void | To accept a decimal number and print its octal equivalent |

151

OUTPUT



152

PROGRAM 6

Write a program in java to accept a number and check whether it is a happy number or not. A number is said to be a Happy number if the sum of the square of the digits are calculated till a single digit is obtained by recursively adding the sum of the square of the digits. If the single digit comes to be 1 then the number is a Happy number.

Example:

28 = (2)^2 + (8)^2 = 4 + 64 = 68

68 = (6)^2 + (8)^2 = 36 + 64 = 100

100 = (1)^2 + (0)^2 + (0)^2 = 1 + 0 + 0 = 1

Sample input:

Enter a number: 28

Sample output:

28 is a happy number

153

Algorithm

Step 1: Start

Step 2: Scanner class object declared

Step 3: A number is accepted from the user.

Step 4: An object *Happy* is declared.

Step 5: The following steps will be executed until the number stored in  *no* variable  is greater than 9

Step 5.1: If the number is greater than 9 then thenumber is passed to *sumofdgt* functionand the returned value is stored in *s* variable.

Step 5.2: again the value in *s* variable is initialized to *no* variable.

Step 6: if the eventual sum of square of all theDigits of the accepted number is onethen print the Number is happy number else print not a happy number

Step 7: Stop.

ALGORITHM FOR sumofdgt ()

Step 1: Start

Step 2: the number received from main Function is initialised to *no* variable.

Step 3: repeat the following steps until the Number is equal to zero

Step 3.1: Here the number is divided by 10 and the square of the remainder is addedwhile the quotient is again passed to *sumofdgt* Function.

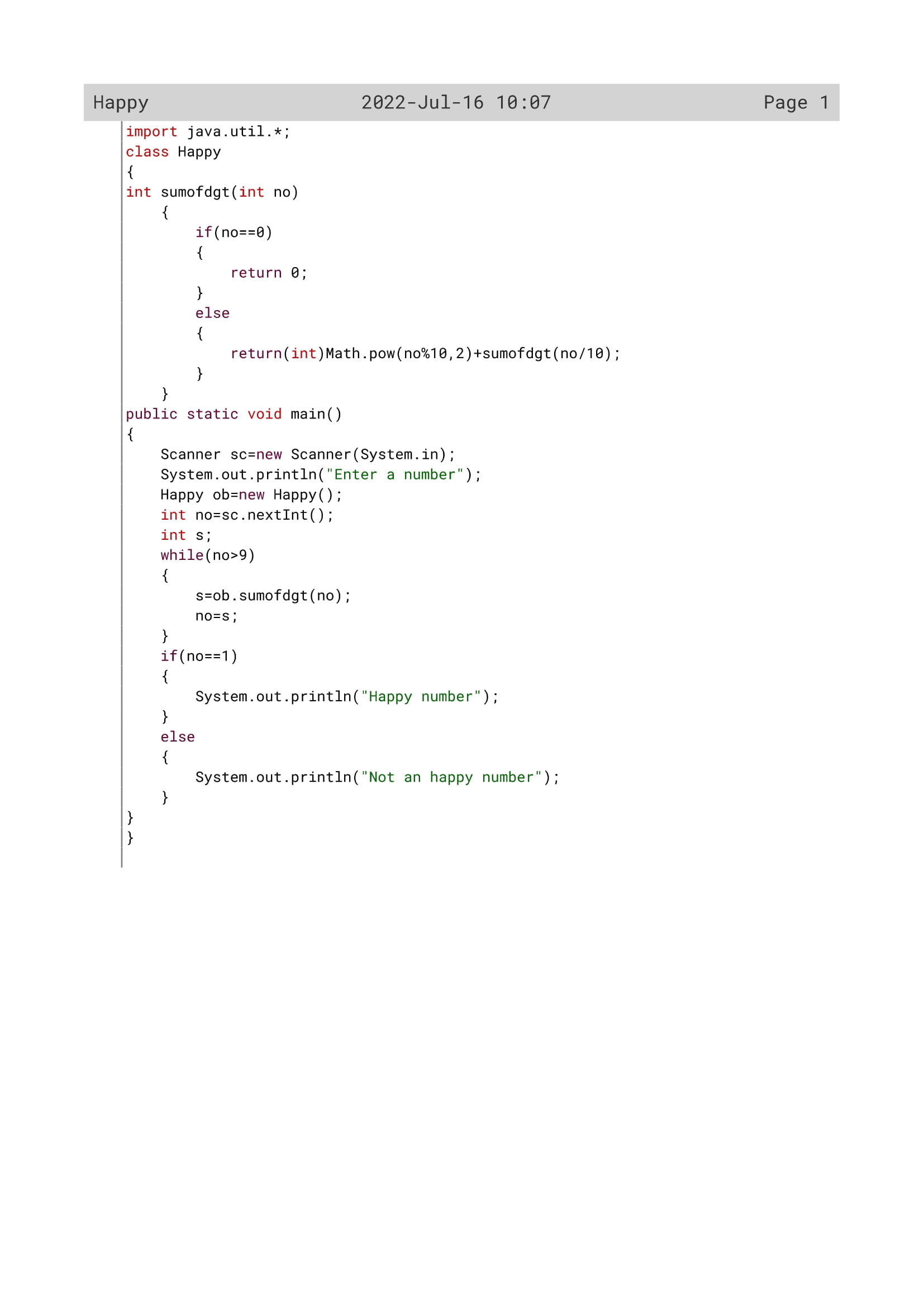
Step 3.2: Again the same steps will be followed And the remainders will be added to get The sum of digits

Step 4: when the number will become zero wewill return the sum of digits to main function.

Step 5: Stop.

154

Source Code



155

Variable Description

|  |  |  |
| --- | --- | --- |
| Variable name | Data type | Description |
| no | int | To accept a number by the user |
| s | Int | To send the number to the method sumofdgt() if it is greater than 9 |

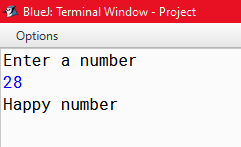
156

Function Description

|  |  |  |
| --- | --- | --- |
| Function name | Data type | description |
| sumofdgt() | Int | To calculate sum of digits |
| main() | void | To check whether the given number is a happy number or not |

157

OUTPUT



158

PROGRAM 7

Write a program in java to implement binary search.

159

Algorithm

Step 1: Start.

Step 2: A flag variable is initialized with 0.

Step 3: A function called binarySearch is defined with the parameters arr[], first, last and key.

Step 4: A mid variable is declared which is assigned with (first+last)/2.

Step 5: If first is greater than last flag=0.

Step 6: if arr[mid]is less than key the function is called again with mid-1 in place of mid.

Step 7: The flag variable is returned.

Step 8: public static void main.

Step 9: Object of scanner class is defined.

Step 10: The size of the array is taken in variable n.

Step 11: The elements of the array are taken using a loop.

Step 12: An object obj is defined of class BinarySearch.

Step 13: The function is called giving the required arguments.

Step 14: If flag variable is 1 , element is found else it is not found.

Step 15: Stop.

Algorithm for binarySearch()

Step 1: Start

Step 2: mid value is calculated

Step 3: If first is greater than last flag changes to 0 else if arr[mid] value is less than key value then function is recalled in itself with mid value increased by 1 in first and if mid value is greater than key then function is recalled with mid value decreased by 1 in last

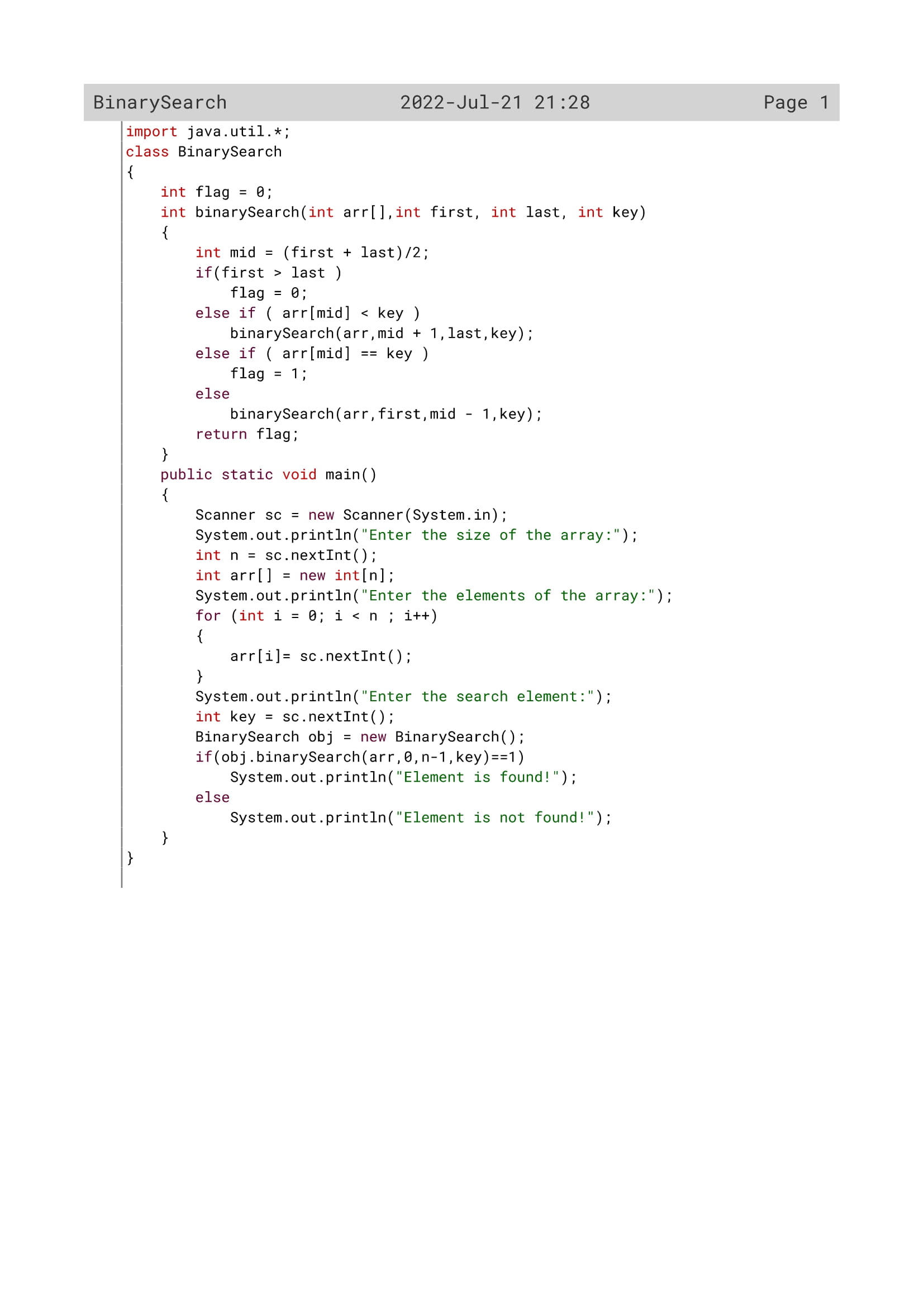
Step 4: if arr[mid] is equal to key flag value changes to 1

Step 5: flag value is returned

Step 6: Stop

160

Source Code



161

Variable Description

|  |  |  |
| --- | --- | --- |
| Variable | Data type | Description |
| flag | int | Indicator variable. |
| first | int | To initialize with the first element of array. |
| last | int | To initialize with the last element of array |
| key | int | To get the key |
| mid | int | To initialize with the middle element of array |
| arr | int | To take all the elements of array |
| n | int | The element to be searched |

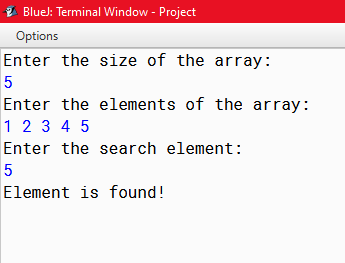
162

Function Description

|  |  |  |
| --- | --- | --- |
| Function name | Data type | description |
| binarysearch() | Int | To search element. |
| main() | void | To set the flow of program and take elements from user. |

163

OUTPUT



164