mport java.util.Scanner;

class BinaryToDecimal

{

        public static void main(String args[])

        {

            Scanner s=new Scanner(System.in);

            System.out.println("Enter a binary number:");

            String n=s.nextLine();

            System.out.println(Integer.parseInt(n,2));

        }

}

mport java.util.Scanner;

class BinaryToDecimal

{

        public static void main(String args[])

        {

            Scanner s=new Scanner(System.in);

            System.out.println("Enter a binary number:");

            int n=s.nextInt();

            int decimal=0,p=0;

            while(n!=0)

            {

                decimal+=((n%10)\*Math.pow(2,p));

                n=n/10;

                p++;

            }

            System.out.println(decimal);

        }

}

import java.util.Scanner;

public class Exercise22 {

public static void main(String[] args)

{

Scanner sc = new Scanner(System.in);

long binaryNumber, decimalNumber = 0, j = 1, remainder;

System.out.print("Input a binary number: ");

binaryNumber = sc.nextLong();

while (binaryNumber != 0)

{

remainder = binaryNumber % 10;

decimalNumber = decimalNumber + remainder \* j;

j = j \* 2;

binaryNumber = binaryNumber / 10;

}

System.out.println("Decimal Number: " + decimalNumber);

}

}

Sample Output:

Input a binary number: 100

Decimal Number: 4

public class Bin2dec {

/\*\*  
\* @param args the command line arguments  
\*/  
public static void main(String[] args) {  
Scanner s = new Scanner(System.in);  
System.out.println(“Enter the Word”);  
String str = s.next();  
int number;  
double digit = 0 , result = 0;  
int check = 1;  
for(int i = 0; i < str.length() ; i++){  
if(str.charAt(i) =='1' || str.charAt(i) =='0'){  
check = 1;  
}  
else{  
check = 0;  
}  
}  
if(check == 0){  
System.out.println("Error: Invalid Binary String \"" + str + "\"");  
}  
else if(check == 1){  
number = Integer.parseInt(str);  
for(int i = 0; i < str.length() ; i++){  
digit = number % 10;  
number = number / 10;  
result = result + (digit \* pow(2,i));  
}  
System.out.println((int)result);  
}

[July 7, 2017](https://www.thecrazyprogrammer.com/2015/11/program-to-convert-binary-to-decimal-in-java.html#comment-49158)

import java.io.\*;  
class BinaryToDecimal  
{  
public static void main(String [] args) throws IOException  
{  
BufferedReader br = new BufferedReader (new InputStreamReader (System.in));  
System.out.println(“Enter a binary a number:”);  
String a = br.readLine();  
char x[] = a.toCharArray();  
int power=0;  
long decimal=0 , sum=0;  
for(int i =x.length-1 ; i>=0;i–)  
{

int z = (int)java.lang.Math.pow(2,power);  
sum = sum + ((x[i])\*(z));  
power++;  
sum=0;  
}  
System.out.println(decimal);  
}  
}

|  |  |  |
| --- | --- | --- |
| |  |  | | --- | --- | | |  | | --- | | package com.java2novice.algos;    public class BinaryToDecimal {        public int getDecimalFromBinary(int binary){            int decimal = 0;          int power = 0;          while(true){              if(binary == 0){                  break;              } else {                  int tmp = binary%10;                  decimal += tmp\*Math.pow(2, power);                  binary = binary/10;                  power++;              }          }          return decimal;      }        public static void main(String a[]){          BinaryToDecimal bd = new BinaryToDecimal();          System.out.println("11 ===> "+bd.getDecimalFromBinary(11));          System.out.println("110 ===> "+bd.getDecimalFromBinary(110));          System.out.println("100110 ===> "+bd.getDecimalFromBinary(100110));      }  } | | |
| |  | | --- | | **Output:** | | 11 ===> 3  110 ===> 6  100110 ===> 38 | |

**Method 1: Binary to Decimal conversion using Integer.parseInt() method**

import java.util.Scanner;

class BinaryToDecimal {

public static void main(String args[]){

Scanner input = new Scanner( System.in );

System.out.print("Enter a binary number: ");

String binaryString =input.nextLine();

System.out.println("Output: "+Integer.parseInt(binaryString,2));

}

}

**Output:**

Enter a binary number: 1101

Output: 13

**Method 2: Conversion without using parseInt**

public class Details {

public int BinaryToDecimal(int binaryNumber){

int decimal = 0;

int p = 0;

while(true){

if(binaryNumber == 0){

break;

} else {

int temp = binaryNumber%10;

decimal += temp\*Math.pow(2, p);

binaryNumber = binaryNumber/10;

p++;

}

}

return decimal;

}

public static void main(String args[]){

Details obj = new Details();

System.out.println("110 --> "+obj.BinaryToDecimal(110));

System.out.println("1101 --> "+obj.BinaryToDecimal(1101));

System.out.println("100 --> "+obj.BinaryToDecimal(100));

System.out.println("110111 --> "+obj.BinaryToDecimal(110111));

}

}

**Output:**

110 --> 6

1101 --> 13

100 --> 4

110111 --> 55

# Java program for linear search – Example

/\* Program: Linear Search Example

\* Written by: Chaitanya from beginnersbook.com

\* Input: Number of elements, element's values, value to be searched

\* Output:Position of the number input by user among other numbers\*/

import java.util.Scanner;

class LinearSearchExample

{

public static void main(String args[])

{

int counter, num, item, array[];

//To capture user input

Scanner input = new Scanner(System.in);

System.out.println("Enter number of elements:");

num = input.nextInt();

//Creating array to store the all the numbers

array = new int[num];

System.out.println("Enter " + num + " integers");

//Loop to store each numbers in array

for (counter = 0; counter < num; counter++)

array[counter] = input.nextInt();

System.out.println("Enter the search value:");

item = input.nextInt();

for (counter = 0; counter < num; counter++)

{

if (array[counter] == item)

{

System.out.println(item+" is present at location "+(counter+1));

/\*Item is found so to stop the search and to come out of the

\* loop use break statement.\*/

break;

}

}

if (counter == num)

System.out.println(item + " doesn't exist in array.");

}

}

Output 1:

Enter number of elements:

6

Enter 6 integers

22

33

45

1

3

99

Enter the search value:

45

45 is present at location 3

Output 2:

Enter number of elements:

4

Enter 4 integers

11

22

4

5

Enter the search value:

99

99 doesn't exist in array.

# Java program to perform binary search – Example

/\* Program: Binary Search Example

\* Written by: Chaitanya from beginnersbook.com

\* Input: Number of elements, element's values, value to be searched

\* Output:Position of the number input by user among other numbers\*/

import java.util.Scanner;

class BinarySearchExample

{

public static void main(String args[])

{

int counter, num, item, array[], first, last, middle;

//To capture user input

Scanner input = new Scanner(System.in);

System.out.println("Enter number of elements:");

num = input.nextInt();

//Creating array to store the all the numbers

array = new int[num];

System.out.println("Enter " + num + " integers");

//Loop to store each numbers in array

for (counter = 0; counter < num; counter++)

array[counter] = input.nextInt();

System.out.println("Enter the search value:");

item = input.nextInt();

first = 0;

last = num - 1;

middle = (first + last)/2;

while( first <= last )

{

if ( array[middle] < item )

first = middle + 1;

else if ( array[middle] == item )

{

System.out.println(item + " found at location " + (middle + 1) + ".");

break;

}

else

{

last = middle - 1;

}

middle = (first + last)/2;

}

if ( first > last )

System.out.println(item + " is not found.\n");

}

}

Output 1:

Enter number of elements:

7

Enter 7 integers

4

5

66

77

8

99

0

Enter the search value:

77

77 found at location 4.

Output 2:

Enter number of elements:

5

Enter 5 integers

12

3

77

890

23

Enter the search value:

99

99 is not found.

# Java program to print Floyd’s triangle – Example

/\* Program: It Prints Floyd's triangle based on user inputs

\* Written by: Chaitanya from beginnersbook.com

\* Input: Number of rows

\* output: floyd's triangle\*/

import java.util.Scanner;

class FloydTriangleExample

{

public static void main(String args[])

{

int rows, number = 1, counter, j;

//To get the user's input

Scanner input = new Scanner(System.in);

System.out.println("Enter the number of rows for floyd's triangle:");

//Copying user input into an integer variable named rows

rows = input.nextInt();

System.out.println("Floyd's triangle");

System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

for ( counter = 1 ; counter <= rows ; counter++ )

{

for ( j = 1 ; j <= counter ; j++ )

{

System.out.print(number+" ");

//Incrementing the number value

number++;

}

//For new line

System.out.println();

}

}

}

Output:

Enter the number of rows for floyd's triangle:

6

Floyd's triangle

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1

2 3

4 5 6

7 8 9 10

11 12 13 14 15

16 17 18 19 20 21

# Java program to reverse a number using for, while and recursion

import java.util.Scanner;

class ReverseNumberWhile

{

public static void main(String args[])

{

int num=0;

int reversenum =0;

System.out.println("Input your number and press enter: ");

//This statement will capture the user input

Scanner in = new Scanner(System.in);

//Captured input would be stored in number num

num = in.nextInt();

//While Loop: Logic to find out the reverse number

while( num != 0 )

{

reversenum = reversenum \* 10;

reversenum = reversenum + num%10;

num = num/10;

}

System.out.println("Reverse of input number is: "+reversenum);

}

}

Output:

Input your number and press enter:

145689

Reverse of input number is: 986541

### Program 2: Reverse a number using for Loop

import java.util.Scanner;

class ForLoopReverseDemo

{

public static void main(String args[])

{

int num=0;

int reversenum =0;

System.out.println("Input your number and press enter: ");

//This statement will capture the user input

Scanner in = new Scanner(System.in);

//Captured input would be stored in number num

num = in.nextInt();

/\* for loop: No initialization part as num is already

\* initialized and no increment/decrement part as logic

\* num = num/10 already decrements the value of num

\*/

for( ;num != 0; )

{

reversenum = reversenum \* 10;

reversenum = reversenum + num%10;

num = num/10;

}

System.out.println("Reverse of specified number is: "+reversenum);

}

}

Output:

Input your number and press enter:

56789111

Reverse of specified number is: 11198765

### Program 3: Reverse a number using recursion

import java.util.Scanner;

class RecursionReverseDemo

{

//A method for reverse

public static void reverseMethod(int number) {

if (number < 10) {

System.out.println(number);

return;

}

else {

System.out.print(number % 10);

//Method is calling itself: recursion

reverseMethod(number/10);

}

}

public static void main(String args[])

{

int num=0;

System.out.println("Input your number and press enter: ");

Scanner in = new Scanner(System.in);

num = in.nextInt();

System.out.print("Reverse of the input number is:");

reverseMethod(num);

System.out.println();

}

}

Output:

Input your number and press enter:

5678901

Reverse of the input number is:1098765

# ava program to generate random number – Example

/\* Program: Random number generator

\* Written by: Chaitanya from beginnersbook.com

\* Input: None

\* Output:Random number between o and 200\*/

import java.util.\*;

class GenerateRandomNumber {

public static void main(String[] args) {

int counter;

Random rnum = new Random();

/\* Below code would generate 5 random numbers

\* between 0 and 200.

\*/

System.out.println("Random Numbers:");

System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

for (counter = 1; counter <= 5; counter++) {

System.out.println(rnum.nextInt(200));

}

}

}

Output:

Random Numbers:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

135

173

5

17

15

# Java Program to display first n or first 100 prime numbers

import java.util.Scanner;

class PrimeNumberDemo

{

public static void main(String args[])

{

int n;

int status = 1;

int num = 3;

//For capturing the value of n

Scanner scanner = new Scanner(System.in);

System.out.println("Enter the value of n:");

//The entered value is stored in the var n

n = scanner.nextInt();

if (n >= 1)

{

System.out.println("First "+n+" prime numbers are:");

//2 is a known prime number

System.out.println(2);

}

for ( int i = 2 ; i <=n ; )

{

for ( int j = 2 ; j <= Math.sqrt(num) ; j++ )

{

if ( num%j == 0 )

{

status = 0;

break;

}

}

if ( status != 0 )

{

System.out.println(num);

i++;

}

status = 1;

num++;

}

}

}

Output:

Enter the value of n:

15

First 15 prime numbers are:

2

3

5

7

11

13

17

19

23

29

31

37

41

43

47

**Program to display first 100 prime numbers**  
To display the first 100 prime numbers, you can **either enter n value as 100 in the above program**OR write a program like this:

class PrimeNumberDemo

{

public static void main(String args[])

{

int n;

int status = 1;

int num = 3;

System.out.println("First 100 prime numbers are:");

System.out.println(2);

for ( int i = 2 ; i <=100 ; )

{

for ( int j = 2 ; j <= Math.sqrt(num) ; j++ )

{

if ( num%j == 0 )

{

status = 0;

break;

}

}

if ( status != 0 )

{

System.out.println(num);

i++;

}

status = 1;

num++;

}

}

}

Output:

First 100 prime numbers are:

2

3

5

7

11

13

17

19

23

29

31

37

41

43

47

53

59

61

67

71

73

79

83

89

97

101

103

107

109

113

127

131

137

139

149

151

157

163

167

173

179

181

191

193

197

199

211

223

227

229

233

239

241

251

257

263

269

271

277

281

283

293

307

311

313

317

331

337

347

349

353

359

367

373

379

383

389

397

401

409

419

421

431

433

439

443

449

457

461

463

467

479

487

491

499

503

509

521

523

541

# Java program to display prime numbers from 1 to 100 and 1 to n

## Program to display the prime numbers from 1 to 100

It will display the prime numbers between 1 and 100.

class PrimeNumbers

{

public static void main (String[] args)

{

int i =0;

int num =0;

//Empty String

String primeNumbers = "";

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

{

int counter=0;

for(num =i; num>=1; num--)

{

if(i%num==0)

{

counter = counter + 1;

}

}

if (counter ==2)

{

//Appended the Prime number to the String

primeNumbers = primeNumbers + i + " ";

}

}

System.out.println("Prime numbers from 1 to 100 are :");

System.out.println(primeNumbers);

}

}

Output:

Prime numbers from 1 to 100 are :

2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59 61 67 71 73 79 83 89 97

## Program to display prime numbers from 1 to n

It will display all the prime numbers between 1 and n (n is the number, entered by user).

import java.util.Scanner;

class PrimeNumbers2

{

public static void main (String[] args)

{

Scanner scanner = new Scanner(System.in);

int i =0;

int num =0;

//Empty String

String primeNumbers = "";

System.out.println("Enter the value of n:");

int n = scanner.nextInt();

scanner.close();

for (i = 1; i <= n; i++)

{

int counter=0;

for(num =i; num>=1; num--)

{

if(i%num==0)

{

counter = counter + 1;

}

}

if (counter ==2)

{

//Appended the Prime number to the String

primeNumbers = primeNumbers + i + " ";

}

}

System.out.println("Prime numbers from 1 to n are :");

System.out.println(primeNumbers);

}

}

Output:

Enter the value of n:

150

Prime numbers from 1 to n are :

2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59 61 67 71 73 79 83 89

97 101 103 107 109 113 127 131 137 139 149

# java program to check palindrome string using recursion

package beginnersbook.com;

import java.util.Scanner;

class PalindromeCheck

{

//My Method to check

public static boolean isPal(String s)

{ // if length is 0 or 1 then String is palindrome

if(s.length() == 0 || s.length() == 1)

return true;

if(s.charAt(0) == s.charAt(s.length()-1))

/\* check for first and last char of String:

\* if they are same then do the same thing for a substring

\* with first and last char removed. and carry on this

\* until you string completes or condition fails

\* Function calling itself: Recursion

\*/

return isPal(s.substring(1, s.length()-1));

/\* If program control reaches to this statement it means

\* the String is not palindrome hence return false.

\*/

return false;

}

public static void main(String[]args)

{

//For capturing user input

Scanner scanner = new Scanner(System.in);

System.out.println("Enter the String for check:");

String string = scanner.nextLine();

/\* If function returns true then the string is

\* palindrome else not

\*/

if(isPal(string))

System.out.println(string + " is a palindrome");

else

System.out.println(string + " is not a palindrome");

}

}

Output:

Enter the String for check:

qqaabb

qqaabb is not a palindrome

Output 2:

Enter the String for check:

cocoococ

cocoococ is a palindrome

## What is a Class in OOPs Concepts

public class Website {

//fields (or instance variable)

String webName;

int webAge;

// constructor

Website(String name, int age){

this.webName = name;

this.webAge = age;

}

public static void main(String args[]){

//Creating objects

Website obj1 = new Website("beginnersbook", 5);

Website obj2 = new Website("google", 18);

//Accessing object data through reference

System.out.println(obj1.webName+" "+obj1.webAge);

System.out.println(obj2.webName+" "+obj2.webAge);

}

}

**Output:**

beginnersbook 5

google 18

## What is a Constructor

#### Example of constructor

public class ConstructorExample {

int age;

String name;

//Default constructor

ConstructorExample(){

this.name="Chaitanya";

this.age=30;

}

//Parameterized constructor

ConstructorExample(String n,int a){

this.name=n;

this.age=a;

}

public static void main(String args[]){

ConstructorExample obj1 = new ConstructorExample();

ConstructorExample obj2 =

new ConstructorExample("Steve", 56);

System.out.println(obj1.name+" "+obj1.age);

System.out.println(obj2.name+" "+obj2.age);

}

}

**Output:**

Chaitanya 30

Steve 56

### Encapsulation

Encapsulation simply means binding object state(fields) and behavior(methods) together. If you are creating class, you are doing encapsulation.

#### Encapsulation example in Java

How to  
1) Make the instance variables private so that they cannot be accessed directly from outside the class. You can only set and get values of these variables through the methods of the class.  
2) Have getter and setter methods in the class to set and get the values of the fields.

class EmployeeCount

{

private int numOfEmployees = 0;

public void setNoOfEmployees (int count)

{

numOfEmployees = count;

}

public double getNoOfEmployees ()

{

return numOfEmployees;

}

}

public class EncapsulationExample

{

public static void main(String args[])

{

EmployeeCount obj = new EmployeeCount ();

obj.setNoOfEmployees(5613);

System.out.println("No Of Employees: "+(int)obj.getNoOfEmployees());

}

}

**Output:**

No Of Employees: 5613

#### Inheritance Example

In this example, we have a parent class Teacher and a child class MathTeacher. In the MathTeacherclass we need not to write the same code which is already present in the present class. Here we have college name, designation and does() method that is common for all the teachers, thus MathTeacher class does not need to write this code, the common data members and methods can inherited from the Teacher class.

class Teacher {

String designation = "Teacher";

String college = "Beginnersbook";

void does(){

System.out.println("Teaching");

}

}

public class MathTeacher extends Teacher{

String mainSubject = "Maths";

public static void main(String args[]){

MathTeacher obj = new MathTeacher();

System.out.println(obj.college);

System.out.println(obj.designation);

System.out.println(obj.mainSubject);

obj.does();

}

}

**Output:**

Beginnersbook

Teacher

Maths

Teaching

# Constructors in Java – A complete study!!

public class Hello {

String name;

//Constructor

Hello(){

this.name = "BeginnersBook.com";

}

public static void main(String[] args) {

Hello obj = new Hello();

System.out.println(obj.name);

}

}

**Output:**

BeginnersBook.com

#### Example: no-arg constructor

class Demo

{

public Demo()

{

System.out.println("This is a no argument constructor");

}

public static void main(String args[]) {

new Demo();

}

}

Output:  
This is a no argument constructor

### Parameterized constructor

Constructor with arguments(or you can say parameters) is known as [Parameterized constructor](https://beginnersbook.com/2014/01/parameterized-constructor-in-java-example/).

#### Example: parameterized constructor

In this example we have a parameterized constructor with two parameters id and name. While creating the objects obj1 and obj2 I have passed two arguments so that this constructor gets invoked after creation of obj1 and obj2.

public class Employee {

int empId;

String empName;

//parameterized constructor with two parameters

Employee(int id, String name){

this.empId = id;

this.empName = name;

}

void info(){

System.out.println("Id: "+empId+" Name: "+empName);

}

public static void main(String args[]){

Employee obj1 = new Employee(10245,"Chaitanya");

Employee obj2 = new Employee(92232,"Negan");

obj1.info();

obj2.info();

}

}

**Output:**

Id: 10245 Name: Chaitanya

Id: 92232 Name: Negan

#### Example2: parameterized constructor

In this example, we have two constructors, a default constructor and a parameterized constructor. When we do not pass any parameter while creating the object using new keyword then default constructor is invoked, however when you pass a parameter then parameterized constructor that matches with the passed parameters list gets invoked.

class Example2

{

private int var;

//default constructor

public Example2()

{

this.var = 10;

}

//parameterized constructor

public Example2(int num)

{

this.var = num;

}

public int getValue()

{

return var;

}

public static void main(String args[])

{

Example2 obj = new Example2();

Example2 obj2 = new Example2(100);

System.out.println("var is: "+obj.getValue());

System.out.println("var is: "+obj2.getValue());

}

}

**Output:**

var is: 10

var is: 100

## What if you implement only parameterized constructor in class

class Example3

{

private int var;

public Example3(int num)

{

var=num;

}

public int getValue()

{

return var;

}

public static void main(String args[])

{

Example3 myobj = new Example3();

System.out.println("value of var is: "+myobj.getValue());

}

}

## Super()

Whenever a child class constructor gets invoked it implicitly invokes the constructor of parent class. You can also say that the compiler inserts a super(); statement at the beginning of child class constructor.

class MyParentClass {

MyParentClass(){

System.out.println("MyParentClass Constructor");

}

}

class MyChildClass extends MyParentClass{

MyChildClass() {

System.out.println("MyChildClass Constructor");

}

public static void main(String args[]) {

new MyChildClass();

}

}

**Output:**

MyParentClass Constructor

MyChildClass Constructor

## Java Copy Constructor

A copy constructor is used for copying the values of one object to another object.

class JavaExample{

String web;

JavaExample(String w){

web = w;

}

/\* This is the Copy Constructor, it

\* copies the values of one object

\* to the another object (the object

\* that invokes this constructor)

\*/

JavaExample(JavaExample je){

web = je.web;

}

void disp(){

System.out.println("Website: "+web);

}

public static void main(String args[]){

JavaExample obj1 = new JavaExample("BeginnersBook");

/\* Passing the object as an argument to the constructor

\* This will invoke the copy constructor

\*/

JavaExample obj2 = new JavaExample(obj1);

obj1.disp();

obj2.disp();

}

}

Output:

Website: BeginnersBook

Website: BeginnersBook

## Inheritance Example

class Teacher {

String designation = "Teacher";

String collegeName = "Beginnersbook";

void does(){

System.out.println("Teaching");

}

}

public class PhysicsTeacher extends Teacher{

String mainSubject = "Physics";

public static void main(String args[]){

PhysicsTeacher obj = new PhysicsTeacher();

System.out.println(obj.collegeName);

System.out.println(obj.designation);

System.out.println(obj.mainSubject);

obj.does();

}

}

Output:

Beginnersbook

Teacher

Physics

Teaching

class Teacher {

private String designation = "Teacher";

private String collegeName = "Beginnersbook";

public String getDesignation() {

return designation;

}

protected void setDesignation(String designation) {

this.designation = designation;

}

protected String getCollegeName() {

return collegeName;

}

protected void setCollegeName(String collegeName) {

this.collegeName = collegeName;

}

void does(){

System.out.println("Teaching");

}

}

public class JavaExample extends Teacher{

String mainSubject = "Physics";

public static void main(String args[]){

JavaExample obj = new JavaExample();

/\* Note: we are not accessing the data members

\* directly we are using public getter method

\* to access the private members of parent class

\*/

System.out.println(obj.getCollegeName());

System.out.println(obj.getDesignation());

System.out.println(obj.mainSubject);

obj.does();

}

}

The output is:

Beginnersbook

Teacher

Physics

Teaching

class ParentClass{

//Parent class constructor

ParentClass(){

System.out.println("Constructor of Parent");

}

void disp(){

System.out.println("Parent Method");

}

}

class JavaExample extends ParentClass{

JavaExample(){

System.out.println("Constructor of Child");

}

void disp(){

System.out.println("Child Method");

//Calling the disp() method of parent class

super.disp();

}

public static void main(String args[]){

//Creating the object of child class

JavaExample obj = new JavaExample();

obj.disp();

}

}

The output is :

Constructor of Parent

Constructor of Child

Child Method

Parent Method

# Super keyword in java with example

//Parent class or Superclass or base class

class Superclass

{

int num = 100;

}

//Child class or subclass or derived class

class Subclass extends Superclass

{

/\* The same variable num is declared in the Subclass

\* which is already present in the Superclass

\*/

int num = 110;

void printNumber(){

System.out.println(num);

}

public static void main(String args[]){

Subclass obj= new Subclass();

obj.printNumber();

}

}

Output:  
110

**Accessing the num variable of parent class:**  
By calling a variable like this, we can access the variable of parent class if both the classes (parent and child) have same variable.

super.variable\_name

Let’s take the same example that we have seen above, this time in print statement we are passing super.num instead of num.

class Superclass

{

int num = 100;

}

class Subclass extends Superclass

{

int num = 110;

void printNumber(){

/\* Note that instead of writing num we are

\* writing super.num in the print statement

\* this refers to the num variable of Superclass

\*/

System.out.println(super.num);

}

public static void main(String args[]){

Subclass obj= new Subclass();

obj.printNumber();

}

}

Output:  
100  
As you can see by using super.num we accessed the num variable of parent class.

### 2) Use of super keyword to invoke constructor of parent class

When we create the object of sub class, the new keyword invokes the [constructor](https://beginnersbook.com/2013/03/constructors-in-java/) of child class, which implicitly invokes the constructor of parent class. So the order to execution when we create the object of child class is: parent class constructor is executed first and then the child class constructor is executed. It happens because compiler itself adds super()(this invokes the no-arg constructor of parent class) as the first statement in the constructor of child class.

Let’s see an example to understand what I have explained above:

class Parentclass

{

Parentclass(){

System.out.println("Constructor of parent class");

}

}

class Subclass extends Parentclass

{

Subclass(){

/\* Compile implicitly adds super() here as the

\* first statement of this constructor.

\*/

System.out.println("Constructor of child class");

}

Subclass(int num){

/\* Even though it is a parameterized constructor.

\* The compiler still adds the no-arg super() here

\*/

System.out.println("arg constructor of child class");

}

void display(){

System.out.println("Hello!");

}

public static void main(String args[]){

/\* Creating object using default constructor. This

\* will invoke child class constructor, which will

\* invoke parent class constructor

\*/

Subclass obj= new Subclass();

//Calling sub class method

obj.display();

/\* Creating second object using arg constructor

\* it will invoke arg constructor of child class which will

\* invoke no-arg constructor of parent class automatically

\*/

Subclass obj2= new Subclass(10);

obj2.display();

}

}

**Output:**

Constructor of parent class

Constructor of child class

Hello!

Constructor of parent class

arg constructor of child class

Hello!

#### Parameterized super() call to invoke parameterized constructor of parent class

We can call super() explicitly in the constructor of child class, but it would not make any sense because it would be redundant. It’s like explicitly doing something which would be implicitly done otherwise.  
However when we have a constructor in parent class that takes arguments then we can use parameterized super, like super(100); to invoke [parameterized constructor](https://beginnersbook.com/2014/01/parameterized-constructor-in-java-example/) of parent class from the constructor of child class.  
Let’s see an example to understand this:

class Parentclass

{

//no-arg constructor

Parentclass(){

System.out.println("no-arg constructor of parent class");

}

//arg or parameterized constructor

Parentclass(String str){

System.out.println("parameterized constructor of parent class");

}

}

class Subclass extends Parentclass

{

Subclass(){

/\* super() must be added to the first statement of constructor

\* otherwise you will get a compilation error. Another important

\* point to note is that when we explicitly use super in constructor

\* the compiler doesn't invoke the parent constructor automatically.

\*/

super("Hahaha");

System.out.println("Constructor of child class");

}

void display(){

System.out.println("Hello");

}

public static void main(String args[]){

Subclass obj= new Subclass();

obj.display();

}

}

**Output:**

parameterized constructor of parent class

Constructor of child class

Hello

class Parentclass

{

//Overridden method

void display(){

System.out.println("Parent class method");

}

}

class Subclass extends Parentclass

{

//Overriding method

void display(){

System.out.println("Child class method");

}

void printMsg(){

//This would call Overriding method

display();

//This would call Overridden method

super.display();

}

public static void main(String args[]){

Subclass obj= new Subclass();

obj.printMsg();

}

}

**Output:**

Child class method

Parent class method

class Parentclass

{

void display(){

System.out.println("Parent class method");

}

}

class Subclass extends Parentclass

{

void printMsg(){

/\* This would call method of parent class,

\* no need to use super keyword because no other

\* method with the same name is present in this class

\*/

display();

}

public static void main(String args[]){

Subclass obj= new Subclass();

obj.printMsg();

}

}

**Output:**

Parent class method

# Encapsulation in Java with example

class EncapsulationDemo{

private int ssn;

private String empName;

private int empAge;

//Getter and Setter methods

public int getEmpSSN(){

return ssn;

}

public String getEmpName(){

return empName;

}

public int getEmpAge(){

return empAge;

}

public void setEmpAge(int newValue){

empAge = newValue;

}

public void setEmpName(String newValue){

empName = newValue;

}

public void setEmpSSN(int newValue){

ssn = newValue;

}

}

public class EncapsTest{

public static void main(String args[]){

EncapsulationDemo obj = new EncapsulationDemo();

obj.setEmpName("Mario");

obj.setEmpAge(32);

obj.setEmpSSN(112233);

System.out.println("Employee Name: " + obj.getEmpName());

System.out.println("Employee SSN: " + obj.getEmpSSN());

System.out.println("Employee Age: " + obj.getEmpAge());

}

}

**Output:**

Employee Name: Mario

Employee SSN: 112233

Employee Age: 32

# Exception handling in java with examples

If an exception occurs, which has not been handled by programmer then program execution gets terminated and a system generated error message is shown to the user. For example look at the system generated exception below:

### Advantage of exception handling

Exception handling ensures that the flow of the program doesn’t break when an exception occurs. For example, if a program has bunch of statements and an exception occurs mid way after executing certain statements then the statements after the exception will not execute and the program will terminate abruptly.  
By handling we make sure that all the statements execute and the flow of program doesn’t break.

## Difference between error and exception

**Errors** indicate that something severe enough has gone wrong, the application should crash rather than try to handle the error.

**Exceptions** are events that occurs in the code. A programmer can handle such conditions and take necessary corrective actions. Few examples:  
NullPointerException – When you try to use a reference that points to null.  
ArithmeticException – When bad data is provided by user, for example, when you try to divide a number by zero this exception occurs because dividing a number by zero is undefined.  
ArrayIndexOutOfBoundsException – When you try to access the elements of an array out of its bounds, for example array size is 5 (which means it has five elements) and you are trying to access the 10th element.

## Types of exceptions

There are two types of exceptions in Java:  
1)Checked exceptions  
2)Unchecked exceptions

I have covered this in detail in a separate tutorial: [Checked and Unchecked exceptions in Java](https://beginnersbook.com/2013/04/java-checked-unchecked-exceptions-with-examples/).

### Checked exceptions

All exceptions other than Runtime Exceptions are known as Checked exceptions as the compiler checks them during compilation to see whether the programmer has handled them or not. If these exceptions are not handled/declared in the program, you will get compilation error. For example, SQLException, IOException, ClassNotFoundException etc.

### Unchecked Exceptions

Runtime Exceptions are also known as Unchecked Exceptions. These exceptions are not checked at compile-time so compiler does not check whether the programmer has handled them or not but it’s the responsibility of the programmer to handle these exceptions and provide a safe exit. For example, ArithmeticException, NullPointerException, ArrayIndexOutOfBoundsException etc.

## A Simple Java String Example

public class Example{

public static void main(String args[]){

//creating a string by java string literal

String str = "Beginnersbook";

char arrch[]={'h','e','l','l','o'};

//converting char array arrch[] to string str2

String str2 = new String(arrch);

//creating another java string str3 by using new keyword

String str3 = new String("Java String Example");

//Displaying all the three strings

System.out.println(str);

System.out.println(str2);

System.out.println(str3);

}

}

Output:

Beginnersbook

hello

Java String Example

## Java AWT Example

We can create a GUI using Frame in two ways:  
1) By extending Frame class  
2) By creating the instance of Frame class  
Lets have a look at the example of each one.

## AWT Example 1: creating Frame by extending Frame class

import java.awt.\*;

/\* We have extended the Frame class here,

 \* thus our class "SimpleExample" would behave

 \* like a Frame

 \*/

public class SimpleExample extends Frame{

    SimpleExample(){

        Button b=new Button("Button!!");

        // setting button position on screen

        b.setBounds(50,50,50,50);

        //adding button into frame

        add(b);

        //Setting Frame width and height

        setSize(500,300);

        //Setting the title of Frame

        setTitle("This is my First AWT example");

        //Setting the layout for the Frame

        setLayout(new FlowLayout());

        /\* By default frame is not visible so

         \* we are setting the visibility to true

         \* to make it visible.

         \*/

        setVisible(true);

    }

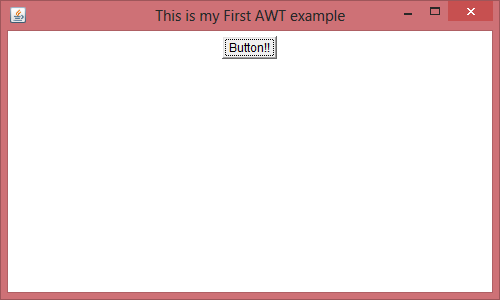
    public static void main(String args[]){

         // Creating the instance of Frame

         SimpleExample fr=new SimpleExample();

    }

}

**Output:**  


## AWT Example 2: creating Frame by creating instance of Frame class

import java.awt.\*;

public class Example2 {

Example2()

   {

      //Creating Frame

      Frame fr=new Frame();

      //Creating a label

      Label lb = new Label("UserId: ");

      //adding label to the frame

      fr.add(lb);

      //Creating Text Field

      TextField t = new TextField();

      //adding text field to the frame

      fr.add(t);

      //setting frame size

      fr.setSize(500, 300);

      //Setting the layout for the Frame

      fr.setLayout(new FlowLayout());

      fr.setVisible(true);

   }

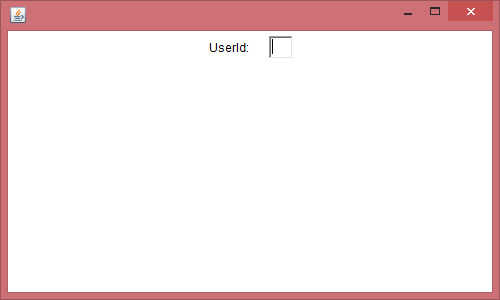
   public static void main(String args[])

   {

       Example2 ex = new Example2();

   }

}

**Output:**  


Calculator

# **How to Make a Calculator in Java Using Swing**

This tutorial is about how to make a calculator in Java. Below I have shared the simple calculator program in java using swing. It is a simple calculator in Java which can perform basic arithmetic operations like addition, subtraction, multiplication and division of two numbers. If you are getting any difficulty to understand or run this program then comment below, I will try to solve problem.

## Program to Make a Calculator in Java Using Swing



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99  100  101  102  103  104  105  106  107  108  109  110  111  112  113  114  115  116  117  118  119  120  121  122  123  124  125  126  127  128  129  130  131  132  133  134  135  136  137  138  139  140  141  142  143  144  145  146  147  148  149  150  151  152  153  154  155  156  157  158  159  160  161  162  163  164  165  166  167  168  169  170  171  172  173  174  175  176  177  178  179  180  181  182  183  184  185  186  187  188  189  190  191  192  193  194  195  196  197  198  199  200  201  202  203  204  205  206  207  208  209  210 |  |  |  | import javax.swing.\*;  import java.awt.event.\*;      class Calc implements ActionListener  {      JFrame f;      JTextField t;      JButton b1,b2,b3,b4,b5,b6,b7,b8,b9,b0,bdiv,bmul,bsub,badd,bdec,beq,bdel,bclr;        static double a=0,b=0,result=0;      static int operator=0;        Calc()      {          f=new JFrame("Calculator");          t=new JTextField();          b1=new JButton("1");          b2=new JButton("2");          b3=new JButton("3");          b4=new JButton("4");          b5=new JButton("5");          b6=new JButton("6");          b7=new JButton("7");          b8=new JButton("8");          b9=new JButton("9");          b0=new JButton("0");          bdiv=new JButton("/");          bmul=new JButton("\*");          bsub=new JButton("-");          badd=new JButton("+");          bdec=new JButton(".");          beq=new JButton("=");          bdel=new JButton("Delete");          bclr=new JButton("Clear");            t.setBounds(30,40,280,30);          b7.setBounds(40,100,50,40);          b8.setBounds(110,100,50,40);          b9.setBounds(180,100,50,40);          bdiv.setBounds(250,100,50,40);            b4.setBounds(40,170,50,40);          b5.setBounds(110,170,50,40);          b6.setBounds(180,170,50,40);          bmul.setBounds(250,170,50,40);            b1.setBounds(40,240,50,40);          b2.setBounds(110,240,50,40);          b3.setBounds(180,240,50,40);          bsub.setBounds(250,240,50,40);            bdec.setBounds(40,310,50,40);          b0.setBounds(110,310,50,40);          beq.setBounds(180,310,50,40);          badd.setBounds(250,310,50,40);            bdel.setBounds(60,380,100,40);          bclr.setBounds(180,380,100,40);            f.add(t);          f.add(b7);          f.add(b8);          f.add(b9);          f.add(bdiv);          f.add(b4);          f.add(b5);          f.add(b6);          f.add(bmul);          f.add(b1);          f.add(b2);          f.add(b3);          f.add(bsub);          f.add(bdec);          f.add(b0);          f.add(beq);          f.add(badd);          f.add(bdel);          f.add(bclr);            f.setLayout(null);          f.setVisible(true);          f.setSize(350,500);          f.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);          f.setResizable(false);            b1.addActionListener(this);          b2.addActionListener(this);          b3.addActionListener(this);          b4.addActionListener(this);          b5.addActionListener(this);          b6.addActionListener(this);          b7.addActionListener(this);          b8.addActionListener(this);          b9.addActionListener(this);          b0.addActionListener(this);          badd.addActionListener(this);          bdiv.addActionListener(this);          bmul.addActionListener(this);          bsub.addActionListener(this);          bdec.addActionListener(this);          beq.addActionListener(this);          bdel.addActionListener(this);          bclr.addActionListener(this);      }        public void actionPerformed(ActionEvent e)      {          if(e.getSource()==b1)              t.setText(t.getText().concat("1"));            if(e.getSource()==b2)              t.setText(t.getText().concat("2"));            if(e.getSource()==b3)              t.setText(t.getText().concat("3"));            if(e.getSource()==b4)              t.setText(t.getText().concat("4"));            if(e.getSource()==b5)              t.setText(t.getText().concat("5"));            if(e.getSource()==b6)              t.setText(t.getText().concat("6"));            if(e.getSource()==b7)              t.setText(t.getText().concat("7"));            if(e.getSource()==b8)              t.setText(t.getText().concat("8"));            if(e.getSource()==b9)              t.setText(t.getText().concat("9"));            if(e.getSource()==b0)              t.setText(t.getText().concat("0"));            if(e.getSource()==bdec)              t.setText(t.getText().concat("."));            if(e.getSource()==badd)          {              a=Double.parseDouble(t.getText());              operator=1;              t.setText("");          }            if(e.getSource()==bsub)          {              a=Double.parseDouble(t.getText());              operator=2;              t.setText("");          }            if(e.getSource()==bmul)          {              a=Double.parseDouble(t.getText());              operator=3;              t.setText("");          }            if(e.getSource()==bdiv)          {              a=Double.parseDouble(t.getText());              operator=4;              t.setText("");          }            if(e.getSource()==beq)          {              b=Double.parseDouble(t.getText());                switch(operator)              {                  case 1: result=a+b;                      break;                    case 2: result=a-b;                      break;                    case 3: result=a\*b;                      break;                    case 4: result=a/b;                      break;                    default: result=0;              }                t.setText(""+result);          }            if(e.getSource()==bclr)              t.setText("");            if(e.getSource()==bdel)          {              String s=t.getText();              t.setText("");              for(int i=0;i<s.length()-1;i++)              t.setText(t.getText()+s.charAt(i));          }      }        public static void main(String...s)      {          new Calc();      }  } |

**import** javax.swing.\*;

**import** javax.swing.border.\*;

**import** java.awt.\*;

**import** java.awt.event.\*;

**import** javax.script.ScriptEngineManager;

**import** javax.script.ScriptEngine;

**import** javax.script.ScriptException;

**class** Calculator **extends** JFrame **implements** ActionListener

{

**private** **static** **final** **int** FRAME\_WIDTH = 300;

**private** **static** **final** **int** FRAME\_HEIGHT = 220;

**private** **static** **final** **int** FRAME\_X = 150;

**private** **static** **final** **int** FRAME\_Y = 100;

**private** JPanel buttonPanel;

**private** JPanel inputOutputPanel;

**private** JTextField info;

**private** **boolean** editable = **true**;

**public** Calculator()

{

Container contentPane = getContentPane();

contentPane.setLayout(**new** FlowLayout());

buttonPanel = **new** JPanel();

buttonPanel.setLayout(**new** GridLayout(4,4,5,5));

*//for padding*

buttonPanel.setBorder(**new** EmptyBorder(10,10,10,10));

inputOutputPanel = **new** JPanel();

inputOutputPanel.setLayout(**new** FlowLayout());

*//for padding*

inputOutputPanel.setBorder(**new** EmptyBorder(10,10,10,10));

setTitle("Calculator using Java");

setSize(FRAME\_WIDTH,FRAME\_HEIGHT);

setResizable(**false**);

setLocation(FRAME\_X,FRAME\_Y);

info = **new** JTextField();

info.setFont(**new** Font("SansSerif", Font.PLAIN, 16));

info.setBackground(Color.white);

info.setBorder(BorderFactory.createLineBorder(Color.black));

info.setPreferredSize(**new** Dimension(270, 35));

info.addActionListener(**this**);

inputOutputPanel.add(info);

contentPane.add(inputOutputPanel);

*//buttons in calculator*

String buttons[] = {"7", "8", "9", "/", "4", "5", "6", "\*", "1", "2","3","-","0",".","=", "+",};

*//adding buttons for buttons array*

**for** (String i : buttons)

{

JButton button = **new** JButton(i);

button.addActionListener(**this**);

buttonPanel.add(button);

}

contentPane.add(buttonPanel);

setDefaultCloseOperation(EXIT\_ON\_CLOSE);

}

**public** **void** actionPerformed(ActionEvent event)

{

*//source is from button*

**if** (event.getSource() **instanceof** JButton)

{

JButton clickedButton = (JButton) event.getSource();

*//if = is clicked then result will be displayed*

**if**(clickedButton.getText().equals("="))

{

addOutput();

}

*//else we will take input*

**else**

{

addInput(clickedButton.getText());

}

}

*//source is from JTextField*

**else**

{

addOutput();

}

}

**public** **void** addInput(String line)

{

*//appending pressed keys to TextField*

*//editable will be false if result is displayed*

**if**(editable)

{

info.setText(info.getText()+line);

}

*//result is on the screen*

*//editable is false*

*//no appending*

*//we will add the text*

**else**

{

info.setText(line);

editable = **true**;

}

}

**public** **void** addOutput()

{

**double** output = 0;

*//converting string into expression*

**try**

{

ScriptEngineManager mgr = **new** ScriptEngineManager();

ScriptEngine engine = mgr.getEngineByName("JavaScript");

info.setText(engine.eval(info.getText()).toString());

}

**catch**(ScriptException e)

{

info.setText("Syntax error");

}

editable = **false**;

}

*//main method*

**public** **static** **void** main (String [] args)

{

Calculator calculator = **new** Calculator();

calculator.setVisible(**true**);

}

}

|  |
| --- |
| package swing\_1; |

|  |  |
| --- | --- |
| 002 |  |

|  |  |
| --- | --- |
| 003 | //The java Template Calculator TODO |

|  |  |
| --- | --- |
| 004 |  |

|  |  |
| --- | --- |
| 005 | import java.awt.EventQueue; |

|  |  |
| --- | --- |
| 006 | import java.awt.GridLayout; |

|  |  |
| --- | --- |
| 007 | import java.awt.BorderLayout; |

|  |  |
| --- | --- |
| 008 | import java.awt.event.ActionListener; |

|  |  |
| --- | --- |
| 009 | import java.awt.event.ActionEvent; |

|  |  |
| --- | --- |
| 010 | import javax.swing.JFrame; |

|  |  |
| --- | --- |
| 011 | import javax.swing.JPanel; |

|  |  |
| --- | --- |
| 012 | import javax.swing.JTextField; |

|  |  |
| --- | --- |
| 013 | import javax.swing.JButton; |

|  |  |
| --- | --- |
| 014 | import java.awt.Container; |

|  |  |
| --- | --- |
| 015 |  |

|  |  |
| --- | --- |
| 016 | public class JavaCalculator implements ActionListener{ |

|  |  |
| --- | --- |
| 017 |  |

|  |  |
| --- | --- |
| 018 | JFrame guiFrame; |

|  |  |
| --- | --- |
| 019 | JPanel buttonPanel; |

|  |  |
| --- | --- |
| 020 | JTextField numberCalc; |

|  |  |
| --- | --- |
| 021 | int calcOperation = 0; |

|  |  |
| --- | --- |
| 022 | int currentCalc; |

|  |  |
| --- | --- |
| 023 |  |

|  |  |
| --- | --- |
| 024 | public static void main(String[] args) { |

|  |  |
| --- | --- |
| 025 |  |

|  |  |
| --- | --- |
| 026 | EventQueue.invokeLater(new Runnable() |

|  |  |
| --- | --- |
| 027 | { |

|  |  |
| --- | --- |
| 028 |  |

|  |  |
| --- | --- |
| 029 | public void run() |

|  |  |
| --- | --- |
| 030 | { |

|  |  |
| --- | --- |
| 031 |  |

|  |  |
| --- | --- |
| 032 | new JavaCalculator(); |

|  |  |
| --- | --- |
| 033 | } |

|  |  |
| --- | --- |
| 034 | }); |

|  |  |
| --- | --- |
| 035 |  |

|  |  |
| --- | --- |
| 036 | } |

|  |  |
| --- | --- |
| 037 |  |

|  |  |
| --- | --- |
| 038 | public JavaCalculator() |

|  |  |
| --- | --- |
| 039 | { |

|  |  |
| --- | --- |
| 040 | guiFrame = new JFrame(); |

|  |  |
| --- | --- |
| 041 | guiFrame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE); |

|  |  |
| --- | --- |
| 042 | guiFrame.setTitle("Simple Calculator"); |

|  |  |
| --- | --- |
| 043 | guiFrame.setSize(300,300); |

|  |  |
| --- | --- |
| 044 | guiFrame.setLocationRelativeTo(null); |

|  |  |
| --- | --- |
| 045 | numberCalc = new JTextField(); |

|  |  |
| --- | --- |
| 046 | numberCalc.setHorizontalAlignment(JTextField.RIGHT); |

|  |  |
| --- | --- |
| 047 | numberCalc.setEditable(false); |

|  |  |
| --- | --- |
| 048 | guiFrame.add(numberCalc, BorderLayout.NORTH); |

|  |  |
| --- | --- |
| 049 | buttonPanel = new JPanel(); |

|  |  |
| --- | --- |
| 050 | buttonPanel.setLayout(new GridLayout(4,4)); |

|  |  |
| --- | --- |
| 051 | guiFrame.add(buttonPanel, BorderLayout.CENTER); |

|  |  |
| --- | --- |
| 052 |  |

|  |  |
| --- | --- |
| 053 | for (int i=0;i<10;i++) |

|  |  |
| --- | --- |
| 054 | { |

|  |  |
| --- | --- |
| 055 | addNumberButton(buttonPanel, String.valueOf(i)); |

|  |  |
| --- | --- |
| 056 | } |

|  |  |
| --- | --- |
| 057 |  |

|  |  |
| --- | --- |
| 058 | addActionButton(buttonPanel, 1, "+"); |

|  |  |
| --- | --- |
| 059 | addActionButton(buttonPanel, 2, "-"); |

|  |  |
| --- | --- |
| 060 | addActionButton(buttonPanel, 3, "\*"); |

|  |  |
| --- | --- |
| 061 | addActionButton(buttonPanel, 4, "/"); |

|  |  |
| --- | --- |
| 062 | addActionButton(buttonPanel, 5, "^2"); |

|  |  |
| --- | --- |
| 063 |  |

|  |  |
| --- | --- |
| 064 | JButton equalsButton = new JButton("="); |

|  |  |
| --- | --- |
| 065 | equalsButton.setActionCommand("="); |

|  |  |
| --- | --- |
| 066 | equalsButton.addActionListener(new ActionListener() |

|  |  |
| --- | --- |
| 067 | { |

|  |  |
| --- | --- |
| 068 | public void actionPerformed(ActionEvent event) |

|  |  |
| --- | --- |
| 069 | { |

|  |  |
| --- | --- |
| 070 | if (!numberCalc.getText().isEmpty()) |

|  |  |
| --- | --- |
| 071 | { |

|  |  |
| --- | --- |
| 072 | int number = Integer.parseInt(numberCalc.getText()); |

|  |  |
| --- | --- |
| 073 | if (calcOperation == 1) |

|  |  |
| --- | --- |
| 074 | { |

|  |  |
| --- | --- |
| 075 | int calculate = currentCalc  + number; |

|  |  |
| --- | --- |
| 076 | numberCalc.setText(Integer.toString(calculate)); |

|  |  |
| --- | --- |
| 077 | } |

|  |  |
| --- | --- |
| 078 | else if (calcOperation == 2) |

|  |  |
| --- | --- |
| 079 | { |

|  |  |
| --- | --- |
| 080 | int calculate = currentCalc  - number; |

|  |  |
| --- | --- |
| 081 | numberCalc.setText(Integer.toString(calculate)); |

|  |  |
| --- | --- |
| 082 | } |

|  |  |
| --- | --- |
| 083 | else if (calcOperation == 3) |

|  |  |
| --- | --- |
| 084 | { |

|  |  |
| --- | --- |
| 085 | int calculate = currentCalc  \* number; |

|  |  |
| --- | --- |
| 086 | numberCalc.setText(Integer.toString(calculate)); |

|  |  |
| --- | --- |
| 087 | } |

|  |  |
| --- | --- |
| 088 | else if (calcOperation == 4) |

|  |  |
| --- | --- |
| 089 | { |

|  |  |
| --- | --- |
| 090 | int calculate = currentCalc  / number; |

|  |  |
| --- | --- |
| 091 | numberCalc.setText(Integer.toString(calculate)); |

|  |  |
| --- | --- |
| 092 | } |

|  |  |
| --- | --- |
| 093 | else if (calcOperation == 5) |

|  |  |
| --- | --- |
| 094 | { |

|  |  |
| --- | --- |
| 095 | int calculate = currentCalc  \* currentCalc; |

|  |  |
| --- | --- |
| 096 | numberCalc.setText(Integer.toString(calculate)); |

|  |  |
| --- | --- |
| 097 | } |

|  |  |
| --- | --- |
| 098 | } |

|  |  |
| --- | --- |
| 099 | } |

|  |  |
| --- | --- |
| 100 | }); |

|  |  |
| --- | --- |
| 101 |  |

|  |  |
| --- | --- |
| 102 | buttonPanel.add(equalsButton); |

|  |  |
| --- | --- |
| 103 | guiFrame.setVisible(true); |

|  |  |
| --- | --- |
| 104 | } |

|  |  |
| --- | --- |
| 105 |  |

|  |  |
| --- | --- |
| 106 | private void addNumberButton(Container parent, String name) |

|  |  |
| --- | --- |
| 107 | { |

|  |  |
| --- | --- |
| 108 | JButton but = new JButton(name); |

|  |  |
| --- | --- |
| 109 | but.setActionCommand(name); |

|  |  |
| --- | --- |
| 110 | but.addActionListener(this); |

|  |  |
| --- | --- |
| 111 | parent.add(but); |

|  |  |
| --- | --- |
| 112 | } |

|  |  |
| --- | --- |
| 113 |  |

|  |  |
| --- | --- |
| 114 | private void addActionButton(Container parent, int action, String text) |

|  |  |
| --- | --- |
| 115 | { |

|  |  |
| --- | --- |
| 116 | JButton but = new JButton(text); |

|  |  |
| --- | --- |
| 117 | but.setActionCommand(text); |

|  |  |
| --- | --- |
| 118 | OperatorAction addAction = new OperatorAction(1); |

|  |  |
| --- | --- |
| 119 | but.addActionListener(addAction); |

|  |  |
| --- | --- |
| 120 | parent.add(but); |

|  |  |
| --- | --- |
| 121 | } |

|  |  |
| --- | --- |
| 122 |  |

|  |  |
| --- | --- |
| 123 | public void actionPerformed(ActionEvent event) |

|  |  |
| --- | --- |
| 124 | { |

|  |  |
| --- | --- |
| 125 | String action = event.getActionCommand(); |

|  |  |
| --- | --- |
| 126 |  |

|  |  |
| --- | --- |
| 127 | numberCalc.setText(action); |

|  |  |
| --- | --- |
| 128 | } |

|  |  |
| --- | --- |
| 129 |  |

|  |  |
| --- | --- |
| 130 | private class OperatorAction implements ActionListener |

|  |  |
| --- | --- |
| 131 | { |

|  |  |
| --- | --- |
| 132 | private int operator; |

|  |  |
| --- | --- |
| 133 |  |

|  |  |
| --- | --- |
| 134 | public OperatorAction(int operation) |

|  |  |
| --- | --- |
| 13 | { |

|  |  |
| --- | --- |
|  | operator = operation; |

|  |  |
| --- | --- |
|  | } |

|  |  |
| --- | --- |
|  |  |

|  |  |
| --- | --- |
|  | public void actionPerformed(ActionEvent event) |

|  |  |
| --- | --- |
|  | { |

|  |  |
| --- | --- |
|  | currentCalc = Integer.parseInt(numberCalc.getText()); |

|  |  |
| --- | --- |
|  | calcOperation = operator; |

|  |  |
| --- | --- |
|  | } |

|  |  |
| --- | --- |
|  | } |

|  |  |
| --- | --- |
|  | } |