/\*\*\*\*\*\*\*\*\*\*\*Fibonacci matrix doubling\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* This program accepts user input value n.It finds binary representation of number

\* and finds 2s power by using bitwise and operation with 10,100,1000.....When a

\* number has got any power of two the program multiplies matrix with itself then

\* multiplies it with resultant matrix. If at particular bit position it has not got 2s power,

\* matrix will be multiplied by itself. This way it achieves nth fibonacci value in log(n)time.

\*/

package eclipsePackage;

import java.math.BigInteger;

import java.util.\*;

import java.util.concurrent.TimeUnit;

public class Fibonacci\_Matrix\_Doubling {

public static void main(String[] args) {

long n =0;

BigInteger value = BigInteger.ZERO;

boolean Again = true;

while(Again)

{

try

{

System.out.println("Please enter n to find fibonacci number: ");

Scanner input\_number = new Scanner(System.in);

n= input\_number.nextLong();

input\_number.close();

Again = false;

}

catch(InputMismatchException ime)

{

System.err.println("Input error. Please enter number. Try Again.");

}

}

value = find\_number(n);

System.out.println("The nth number is:"+value);

}

public static BigInteger find\_number(long n)

{

long time2=0,time1=0;

int i=0, m=2;

long []index = new long[1000];

BigInteger[][] Matrix = new BigInteger[2][2];

Matrix[0][0]= BigInteger.ZERO;

Matrix[0][1]= BigInteger.ONE;

Matrix[1][0]= BigInteger.ONE;

Matrix[1][1]= BigInteger.ONE;

BigInteger[][] Result = new BigInteger [2][2];

Result[0][0]= BigInteger.ZERO;

Result[0][1]= BigInteger.ONE;

Result[1][0]= BigInteger.ONE;

Result[1][1]= BigInteger.ONE;

if ( n ==1 || n == 0)

return BigInteger.valueOf(n);

else

{

String binary\_value= Long.toBinaryString(n);

int len= binary\_value.length();

time1=System.currentTimeMillis();

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

{

index[i] = ((1<<i)&n);

if ( index[i] == m )

{

Matrix = multiplication (Matrix,Matrix);

Result = multiplication(Result,Matrix);

}

else

Matrix = multiplication(Matrix,Matrix);

m=m\*2;

}

time2=System.currentTimeMillis();

long seconds1= TimeUnit.MILLISECONDS.toSeconds(time2-time1) ;

System.out.println("Time taken to generate "+n+ "th number" +" "+(time2-time1)+"milliseconds");

if ( n %2 == 0)

return Result[0][0];

else

return Result[0][1];

}

}

public static BigInteger[][] multiplication(BigInteger C[][], BigInteger D[][])

{

BigInteger[][] result = new BigInteger[2][2];

result[0][0]= BigInteger.ZERO;

result[0][1]= BigInteger.ZERO;

result[1][0]= BigInteger.ZERO;

result[1][1]= BigInteger.ZERO;

BigInteger p= C[0][0].multiply(D[0][0]);

BigInteger q= C[0][1].multiply(D[1][0]);

BigInteger r= C[0][0].multiply(D[0][1]);

BigInteger s= C[0][1].multiply(D[1][1]);

BigInteger v= C[1][0].multiply(D[0][1]);

BigInteger w= C[1][1].multiply(D[1][1]);

result[0][0] = p.add(q);

result[0][1] = r.add(s);

result[1][0] = result[0][1];

result[1][1] = v.add(w);

return result;

}

}