**Universidad de Oriente**

**Núcleo de Anzoátegui**

**Escuela de Ingeniería y Ciencias Aplicadas**

**Departamento de Ingeniería en Computación**

**Análisis y Diseño de Algoritmos**

****

**Práctica 2**

**Profesor: Claudio Cortines Estudiante:**

**Preparador: Cesar Díaz**

**Luis Correa C.I.: 19.840.230**

**Sección: 01**

**Puerto La Cruz, Lunes 28 de Enero 2013**

**INDICE**

|  |
| --- |
| [Introducción: 2](#_Toc347072513)  [1.-Objetivo: 3](#_Toc347072514)  [2.-Metodologia: 3](#_Toc347072515)  [2.1.- Software: 3](#_Toc347072516)  [2.2.- Tabla de cálculo del software: 3](#_Toc347072517)  [2.3.- Código en lenguaje java: 4](#_Toc347072518)  [3.- Conclusiones: 16](#_Toc347072519) |

# Introducción:

El presente informe es sobre el programa de algoritmo para obtener la clave ingresada por el usuario.

Empleando el leguaje de programación “Java”, se obtiene la clave ingresada por el usuario, se podrá obtener más claves reiniciando el programa cada vez que cambie el usuario.

El usuario debe pensar en 4 dígitos, luego mediante la aplicación del programa, la computadora trata de determinar la clave ingresada por el usuario.

Una vez que el usuario ingresa al programa, la computadora procede hacer dos preguntas: cuantos dígitos tienen en común su clave con la mostrada actualmente y cuantos dígitos están en la misma posición que la clave que usted ingresó.

Mediante la comparación de las respuestas dadas por el usuario el programa determinará la clave pensada por el usuario.

## 1.-Objetivo:

Que el programa logre obtener la clave ingresada por el usuario en el mínimo de pasos posibles (si es posible en seis pasos).

## 2.-Metodologia:

Se aplicara la metodología de un software código en lenguaje java y la aplicación de una tabla de cálculo.

## 2.1.- Software:

Para realizar este programa fue utilizado el sistema operativo Windows 7, el IDE netbeans con el lenguaje de programación java.

## 2.2.- Tabla de cálculo del software:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Número pensado | Números  Generados  Por la  computadora | Respuesta 1 | Respuesta 2 | Clave1 | Clave 2 |
| 4655 |  | 0 | 0 | 0000 | 0000 |
|  | 1123 | 0 | 0 | 0000 | 0000 |
|  | 6655 | 3 | 2 | 6655 | 6655 |
|  | 5564 | 4 | 0 | 5564 | 6655 |
|  | 6554 | 4 | 1 | 5564 | 6655 |
|  | 6455 | 4 | 2 | 5564 | 6655 |
|  | 4655 | 4 | 4 | 4655 | 4655 |
| Finalizo |  |  |  |  |  |

## 2.3.- Código en lenguaje java:

* Proyecto se llama GeneradorClave se usan 11 clases son AlmacenDatos, Burbuja,:

public class AlmacenDatos extends java.awt.Point {

public String cadena;

public AlmacenDatos(String cadena,int x,int y){

this.cadena=cadena;

this.x=x; this.y=y;

}

}

public class Burbuja {

public Burbuja(){

}

public static String burbuja(String a){

int cambio=1;

char[] c=a.toCharArray();

char aux;

while( cambio == 1 ){

cambio = 0;

for(int i = 0 ; i < c.length - 1 ; i++ ){

if((int)c[i]>(int)c[i+1]){

aux = c[ i ];

c[ i ] = c[ i + 1 ];

c[ i + 1 ] = aux;

cambio=1;

}

}

}

a="";

for(int i=0;i<c.length;i++){

a=a+c[i];

}

return a;

}

}

public class Computador extends Thread{

private AlmacenDatos clave1=new AlmacenDatos("0000",0,0);

private AlmacenDatos clave2=new AlmacenDatos("0000",0,0);

private java.util.Queue<String> lista;

private boolean sw=true;

public Computador(){

Generador gen=new Generador();

gen.generar();

this.lista=gen.get();

}

private int setPregunta1(){

System.out.println("a)Cuantos digitos son correctos");

System.out.println("pero no estan en los lugares correctos");

return new java.util.Scanner(System.in).nextInt();

}

private int setPregunta2(){

System.out.println("b)Cuantos digitos correctos");

System.out.println("estan en los lugares correctos");

return new java.util.Scanner(System.in).nextInt();

}

@Override

public void run(){

while(sw){

String cad=""; int pregunta1=0,pregunta2=0;

boolean s=true;

while(s){

Posible pos=new Posible(this.lista);

pos.posible();

cad=pos.get();

Pregunta1 p1=new Pregunta1(clave1.cadena,cad,clave1.x);

p1.pregunta1();

Pregunta2 p2=new Pregunta2(clave2.cadena,cad,clave2.y);

p2.pregunta2();

s=p1.get()||p2.get();

}

mostrar();

System.out.println("este es el codigo "+cad);

pregunta1=setPregunta1();

pregunta2=setPregunta2();

System.out.println(this.lista.remove(cad));

if(pregunta1==0){

Eliminador el=new Eliminador(this.lista);

el.generar(cad);

this.lista=el.get();

}

else if(pregunta1==4){

ReElimina el=new ReElimina(this.lista);

el.generar(cad);

this.lista=el.get();

if(pregunta2==4){

System.out.println("su cadena es "+cad); sw=false;

}

almacenaCal1(new AlmacenDatos(cad,pregunta1,pregunta2));

almacenaCal2(new AlmacenDatos(cad,pregunta1,pregunta2));

}

else{

almacenaCal1(new AlmacenDatos(cad,pregunta1,pregunta2));

almacenaCal2(new AlmacenDatos(cad,pregunta1,pregunta2));

}

}

}

public void almacenaCal1(AlmacenDatos a){

if(a.x>this.clave1.x){

this.clave1=a;

}

}

public void almacenaCal2(AlmacenDatos a){

if(a.y>this.clave2.y){

this.clave2=a;

}

if(this.clave1.x==this.clave2.y){ this.clave1.cadena=this.clave2.cadena; }

}

public void mostrar(){

System.out.println("mejor de a) "+this.clave1.cadena);

System.out.println("mejor de b) "+this.clave2.cadena);

}

}

public class Eliminador extends Thread{

private java.util.Queue<String> lista;

private String cadena;

private boolean sw=false;

public Eliminador(java.util.Queue<String> lista){

this.lista=lista;

}

public synchronized void generar(String cadena){

if(sw){ try{wait();}catch(Exception e){} }

this.cadena=cadena;

sw=true;

start();

notify();

}

public synchronized java.util.Queue<String> get(){

if(sw){ try{ wait();}catch(Exception e){} }

notify();

return lista;

}

@Override

public void run(){

int[] numero=new int[4];

for(int i=0;i<4;i++){

numero[i]=Integer.parseInt(cadena.substring(i, i+1));

}

java.util.Queue<String> el=new java.util.LinkedList<String>();

for(String a:lista){

for(int i=0;i<4;i++){

for(int j=0;j<4;j++){

if(Integer.parseInt(a.substring(i, i+1))==numero[j]){

el.add(a);

j=4;

i=4;

}

}

}

}

for(String i:el){

lista.remove(i);

}

/\*

for(String i:lista){

System.out.println(i);

}

\*/

sw=false;

}

}

public class Generador extends Thread {

private java.util.Queue<String> lista=new java.util.LinkedList<String>();

private boolean sw=false;

public Generador(){

}

public void setLista(java.util.Queue<String> lista){

this.lista=lista;

}

public synchronized void generar(){

if(sw!=false){ try{wait();}catch(Exception e){} }

sw=true;

start();

notify();

}

public synchronized java.util.Queue<String> get(){

if(sw==true){ try{wait();}catch(Exception e){} }

sw=false;

notify();

return lista;

}

@Override

public void run(){

//1296

String cadena="123456";

for(int i=0;i<6;i++){

for(int j=0;j<6;j++){

for(int k=0; k<6;k++){

for(int g=0;g<6;g++){

lista.add(cadena.substring(i, i+1)

+cadena.substring(j, j+1)

+cadena.substring(k, k+1)

+cadena.substring(g, g+1));

}

}

}

}

sw=false;

}

}

public class Posible extends Thread{

private java.util.Queue<String> lista;

public boolean sw=false;

private String cadena;

public Posible(java.util.Queue<String> lista){

this.lista=lista;

}

public synchronized void posible(){

if(sw){try{wait();}catch(Exception e){}}

sw=true;

start();

notify();

}

@Override

public void run(){

try

{

java.util.Random rand=new java.util.Random();

int entero=rand.nextInt();

if(entero<0){ entero\*=-1; }

entero=entero%lista.size();

int cont1=0;

for(String a: lista){

if(cont1==entero){

this.cadena=a;

sw=false;

return;

}

cont1++;

}

cadena="";

sw=false;

}

catch(Exception e){ System.out.println("error del programa");}

}

public synchronized String get(){

if(sw){try{wait();}catch(Exception e){}}

notify();

sw=false;

return cadena;

}

}

public abstract class Pregunta extends Thread{

public String cadena;

public String nuevo;

public int entero=0;

public boolean reto=true;

public boolean sw=false;

public Pregunta(String cadena,String nuevo,int entero){

this.cadena=cadena;

this.nuevo=nuevo;

this.entero=entero;

}

}

public class Pregunta1 extends Pregunta{

public Pregunta1(String cadena,String nuevo,int entero){

super(cadena,nuevo,entero);

}

public synchronized void pregunta1(){

if(sw){ try{}catch(Exception e){} }

sw=true;

start();

notify();

}

public synchronized boolean get(){

if(sw){try{wait();} catch(Exception e){}}

notify();

return reto;

}

@Override

public void run(){

this.cadena=new Burbuja().burbuja(cadena);

this.nuevo=new Burbuja().burbuja(nuevo);

int cont=0;

for(int i=0;i<4;i++){

if(Integer.parseInt(cadena.substring(i, i+1))==Integer.parseInt(nuevo.substring(i, i+1))){

cont++;

}

}

if(cont>=entero){this.reto=false; }

else{

cont=0;

for(int i=cadena.length()-1;i>=0;i--){

if(Integer.parseInt(cadena.substring(i, i+1))==Integer.parseInt(nuevo.substring(i, i+1))){

cont++;

}

}

if(cont>=entero){this.reto=false; }

}

sw=false;

}

}

public class Pregunta2 extends Pregunta{

public Pregunta2(String cadena,String nuevo,int entero){

super(cadena,nuevo,entero);

}

public synchronized void pregunta2(){

if(sw){ try{}catch(Exception e){} }

sw=true;

start();

notify();

}

public synchronized boolean get(){

if(sw){try{wait();} catch(Exception e){}}

notify();

return reto;

}

@Override

public void run(){

int cont=0;

for(int i=0;i<4;i++){

if(Integer.parseInt(cadena.substring(i, i+1))==Integer.parseInt(nuevo.substring(i, i+1))){

cont++;

}

}

if(cont>=entero){this.reto=false; }

else{

cont=0;

for(int i=cadena.length()-1;i>=0;i--){

if(Integer.parseInt(cadena.substring(i, i+1))==Integer.parseInt(nuevo.substring(i, i+1))){

cont++;

}

}

if(cont>=entero){this.reto=false; }

}

sw=false;

}

}

public class ReElimina extends Thread{

private java.util.Queue<String> lista;

private java.util.Queue<String> nueva=new java.util.LinkedList<String>();

private String cadena;

private boolean sw=false;

public ReElimina(java.util.Queue<String> lista){

this.lista=lista;

}

public synchronized void generar(String cadena){

if(sw){ try{wait();}catch(Exception e){} }

this.cadena=cadena;

sw=true;

start();

notify();

}

public synchronized java.util.Queue<String> get(){

if(sw){ try{ wait();}catch(Exception e){} }

notify();

return nueva;

}

@Override

public void run(){

int cont=0;

cadena=new Burbuja().burbuja(cadena);

for(String cad:lista){

cad=new Burbuja().burbuja(cad);cont=0;

if(Integer.parseInt(cad)==Integer.parseInt(cadena)){

this.nueva.add(cad);

}

}

sw=false;

}

}

## 3.- Conclusiones:

* Mediante la pregunta 1 (cuantos dígitos en la clave mostrada coinciden con los de su clave sin importar que coincidan en posición), se podrá lograr reducir la cantidad de pruebas a realizar.
* Si la respuesta es 0, la computadora no volverá a repetir una clave con esos dígitos.
* Si la respuesta es 1,2 ó 3, la computadora generará claves con cantidad de dígitos iguales a la clave con mayor valor de la respuesta dada por el usuario.
* Si la respuesta es 4, el computador solo debe realizar pruebas con los dígitos presentes en la clave mostrada.
* Mediante la pregunta 2 (cuantos dígitos son iguales a los que ingreso y están en sus lugares correctos), me permitía saber cuántos dígitos debía repetir en posición durante cada calculo.
* Si la respuesta era 1,2 ó 3, el computador generará claves con dígitos que tengan en común la posición con la clave que con mayor valor en respuesta.