



UNIVERSIDAD DE COSTA RICA

SEDE DE GUANACASTE

Informática Empresarial
IF-2000 Programación I

II Proyecto Programado

Integrantes:

Álvarez Taisigüe Cris
Murillo Vásquez Erick

Carné B97785
Carné B98334

M.C.I Kenneth Sánchez Sánchez

Liberia – Guanacaste
4 de diciembre del 2019
II Ciclo

Índice

Índice	2
1. Introducción	3
2. Objetivos.....	4
2.1 Objetivo General:.....	4
2.2 Objetivo Especifico:	4
3. Descripción del problema y solución	5
3.1 Problema:.....	5
3.2 Solución:.....	6
3.2.1 Diagrama de clases	7
4. Herramienta de Desarrollo	8
4.1 Hardware:	8
4.2 Software:	8
5. Descripción de Datos	9
6. Corrida del programa con datos suministrados	10
7. Código fuente	18
7.1 Clase Persona.	18
7.2 Clase Cliente.	18
7.3 Clase Empleado.	20
7.4 Clase Operaciones.	20
7.5 Clase Prestamos.	21
7.6 Clase Tarjeta.	22
7.7 Clase PYMES.....	24
7.8 Clase Vivienda.....	25
7.9 Clase Personal.	26
7.10 Clase Fiduciaria.....	27
7.11 Clase Registral.	28
7.12 Clase Interfaz.	29
8. Problemas y Limitaciones	56
8.1 Problema:.....	56
8.2 Limitaciones:	56
9. Conclusión	57

1. Introducción

La identidad financiera ACAMP S.A, recibe gran cantidad de clientes todos los días del año. Con esto hay una clara perspectiva de la dificultad de los funcionarios de la identidad financiera para poder recibir un número de clientes exorbitante, y ser atendidos uno por uno para poder solicitar la información correspondiente de cada usuario. Asimismo, los funcionarios de la identidad financiera deberán realizar cálculos según las operaciones que el usuario desea adquirir. Además, se le es difícil la búsqueda de operaciones de los clientes por la gran cantidad de usuarios.

Con lo dicho anteriormente, La empresa ACAMP S.A necesitan de un programa que facilite los cálculos correspondientes de cada operación que ofrecen.

Con el mismo fin de solucionar y facilitar los resultados de los cálculos de cada operación que ofrece y la solicitud de la información personal del usuario y del empleado en un tiempo más eficaz. Igualmente, para agilizar el tiempo, desean que el programa tenga una implementación para poder buscar al usuario por medio el DNI a la vez se desea que se ingrese y recupere la información de los clientes.

Con el problema anterior, se elaboró un programa el cual soluciona las necesidades de la identidad financiera, con el objetivo de tener eficacia a la hora de atender el usuario y a la vez proporcionar un sistema sencillo de entender para los funcionarios de la identidad financiera ACAMP S.A.

El método utilizado para la elaboración de este programa fue desarrollado con el lenguaje de programación Java, mediante un entorno de programación de llamado Eclipse. Se busco que el código fuera claro y entendible para cualquier persona que lo interprete. Se tratará de comprender y analizar las funcionalidades del programa a continuación.

2. Objetivos

2.1 Objetivo General: Explicar el funcionamiento del código del programa realizado para facilitar las necesidades de la identidad financiera ACAMP S.A por medio de un manual al usuario.

2.2 Objetivo Especifico:

- Comprender el problema y buscar la solución a la necesidad de la identidad financiera ACAMP S.A.
- Comprender y describir los atributos y métodos propuestas en el código.
- Explicar el funcionamiento del sistema de manera sencilla.

3. Descripción del problema y solución

3.1 Problema:

La entidad financiera ACAMP S.A lo contrata porque requiere un sistema que utilizando la metodología de POO permita Capturar, Mostrar, Buscar, Almacenar y Recuperar la información de todos los clientes y su respectiva operación crediticia que desee ejecutar. La información relevante para cada operación de cada cliente está compuesta de los siguientes datos: Empleado (DNI, Nombre, Sección (Plataforma, Cajero, Servicio al Cliente). Cliente (DNI, Nombre, Tipo (Físico o Jurídico (Nombre del Accionista mayor y razón social), Monto financiamiento crediticio), Operación crediticia a realizar. Dentro del tipo de operaciones se encuentran: Tarjeta crédito, esta contiene N° de tarjeta, Monto límite de tarjeta, Tasa de interés (55%), Plazo de crédito fijo (72 meses), Monto gastado, Cuota a pagar $((\text{Monto gastado} + (\text{Monto gastado} * \text{Tasa de interés}))/\text{Plazo de crédito fijo})$, Saldo $(\text{Monto gastado} - \text{Cuota a pagar})$.

Otro tipo de operación es PYMES que consta de: N° de operación, Interés fijo PYMES (5%), Monto de crédito, Plazo en meses (84), Tasa de interés (9%), Cuota a pagar $((\text{Monto de crédito} + (\text{Monto de financiamiento crediticio} * \text{interés fijo}) + (\text{Monto de crédito} * \text{tasa de interés}))/\text{plazo en meses})$, Saldo $(\text{Monto de crédito} - \text{Cuota a pagar})$. Existe la operación Préstamo Personal, la cual posee los mismos datos que PYMES excepto el interés fijo PYMES, por lo que este cálculo $(\text{Monto de financiamiento crediticio} * \text{interés fijo})$ en la operación no va.

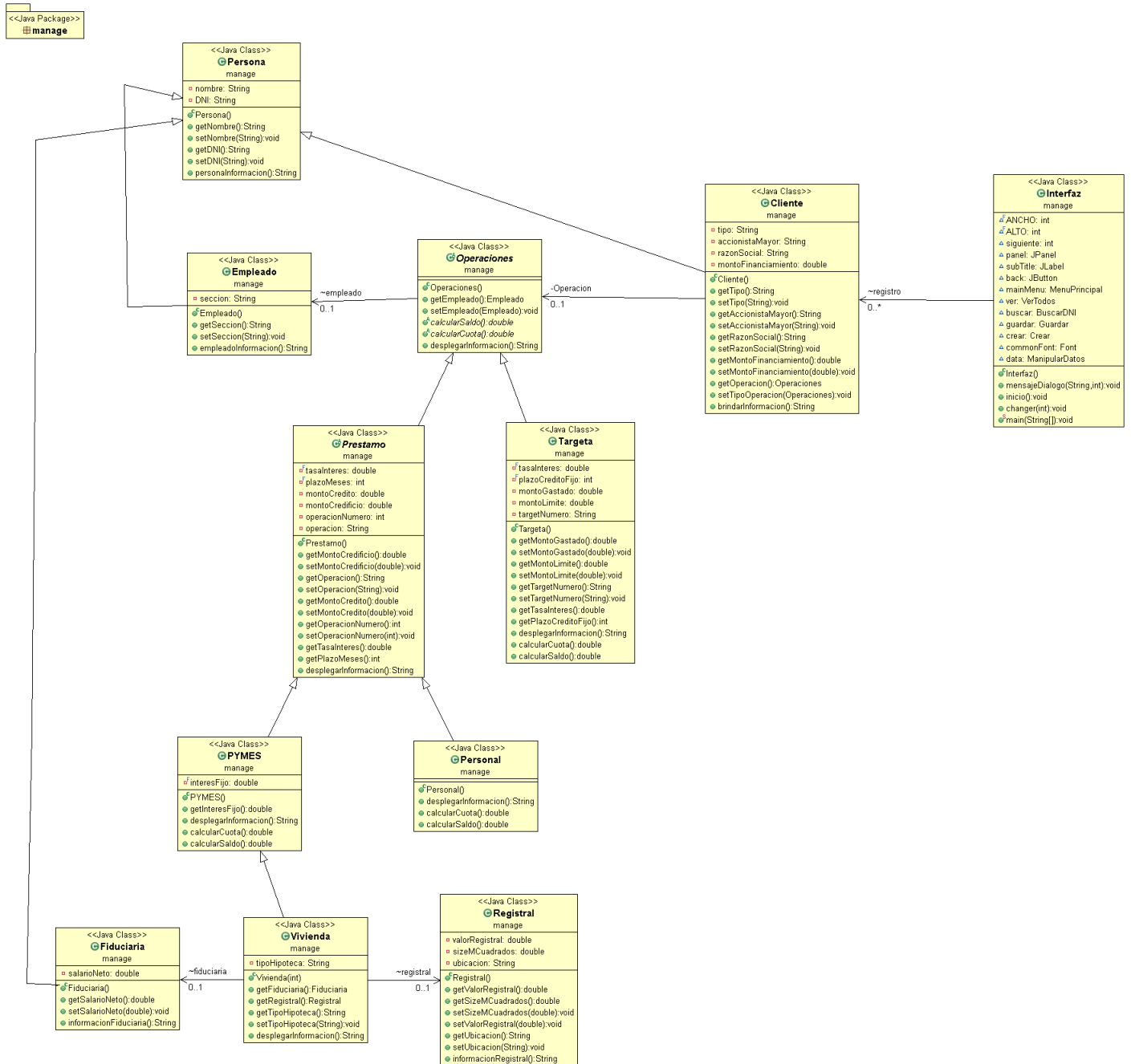
Para la operación Préstamo Vivienda se tienen todos los datos de PYMES, pero además consta de Tipo de Hipoteca que puede ser Fiduciaria (Nombre,

DNI, Salario neto) o puede ser Registral (Monto de Valor registral en ¢, Tamaño en mts, ubicación de la propiedad).

3.2 Solución:

Nuestra solución consiste en la implementación de la programación orientada a objetos, por medio de clases heredadas, agregada y abstractas, sin descuidar el encapsulamiento, polimorfismo para facilitar y respetar el POO, utilizando ligamentos dinámicos, array de objetos y a la vez constantes y variables para el almacenamiento de distintos valores a presentar por cada categoría que se nos especificó, es decir que para cada categoría existirá una clase con sus respectivos atributos. A la vez, se utilizó una interfaz sencilla pero atractiva para el usuario que la utilice así mismismo, cumple con el objetivo principal del problema el cual es: crear un registro de cliente, insertar o agregar registro de cliente, buscar un cliente específico (DNI) y mostrar su información, ver todos los registros de los clientes, guardar la lista de todos los clientes y recuperar la lista de todos los clientes.

3.2.1 Diagrama de clases



4. Herramienta de Desarrollo

4.1 Hardware:

Computador: HP 240 G5 Notebook PC

Procesador: Intel(R) Celeron(R) CPU N3060

Velocidad del procesador: 1.60 GHz

Tipo de Sistema: 64 bits

Memoria RAM: 4 Gb

Memoria de almacenamiento:

4.2 Software:

Sistema Operativo: Windows 10 home Single Language

Versión: 1803

Fabricante del sistema: Microsoft Corporation

Office: Microsoft Office Professional Plus 2019

Editor de texto: Word

Versión de editor de texto: 1905

Programa Utilizado: Eclipse 2019-09

Versión de Java: Jdk 1.8.0_212

5. Descripción de Datos

A continuación, se describirán los datos más importantes del programa.

calcularSaldo(): Este método que es de índole abstracto, se encargará de realizar todos los cálculos de las clases heredadas de Operaciones es decir, este método será sobrecargado.

calcularCuota(): Este método es similar al anterior, puesto que es abstracto y se encargará de sobrescribir los cálculos correlacionados con las diferentes clases que se heredan de la clase madre Operaciones.

desplegarInformacion(): Este método es importante debido que se encargará de mostrar todos los datos del empleado y los criterios del servicio seleccionado. Lo dicho anteriormente, se infiere que este método será sobrecargado dependiendo de la clase en que se vaya a utilizar. A la vez, en este método para poder desplegar toda la información correspondiente de cada clase necesaria, se utilizó polimorfismo por medio la palabra reservada llamada “super” asimismo se evidencia la agregación al utilizar a un objeto para poder manipular los métodos correspondientes de la clase en que pertenece el objeto.

brindarInformacion(): Este método perteneciente a la clase empleado, se encargará de ofrecer los datos correspondientes del usuario en concreto.

tasalInteres: Este es una constante de tipo double, el cual almacenará el interés correspondiente de la tarjeta.

plazoCreditoFijo: De igual manera, este es una constante de tipo double, el cual almacenará el número 72 correspondiente a los meses de plazos fijos correspondiente de la tarjeta.

montoCredito: Esta variable es de tipo double, el cual se guardará el monto del crediticio, esta variable será heredada a las clases PYMES y a Personal y a la vez a sus clases hijas, esta variable no está inicializada debido a que se inicializa dependiendo de la clase hija en la que se utilice.

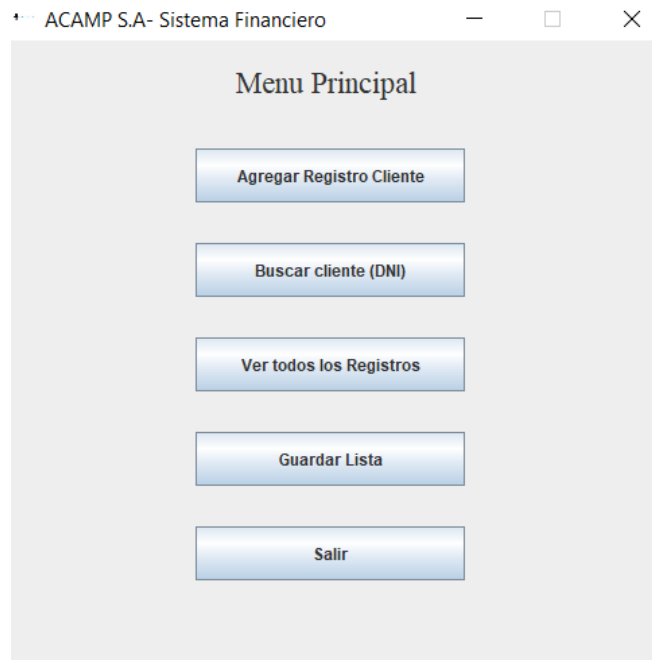
montoCredificio: De igual manera a la anterior variable, la variable montoCredificio es de tipo double, el cual se guardará el monto del crediticio, esta variable será heredada a las clases PYMES y a Personal y a la vez a sus clases hijas, esta variable no está inicializada debido a que se inicializa dependiendo de la clase hija en la que se vaya a utilizar.

Atributos y métodos de la clase Interfaz: Todas las variables y métodos de esta clase son de suma importancia para la elaboración de la interfaz gráfica. Es importante informar, que el fichero de la clase Interfaz cuenta con otras clases para la elaboración de la interfaz.

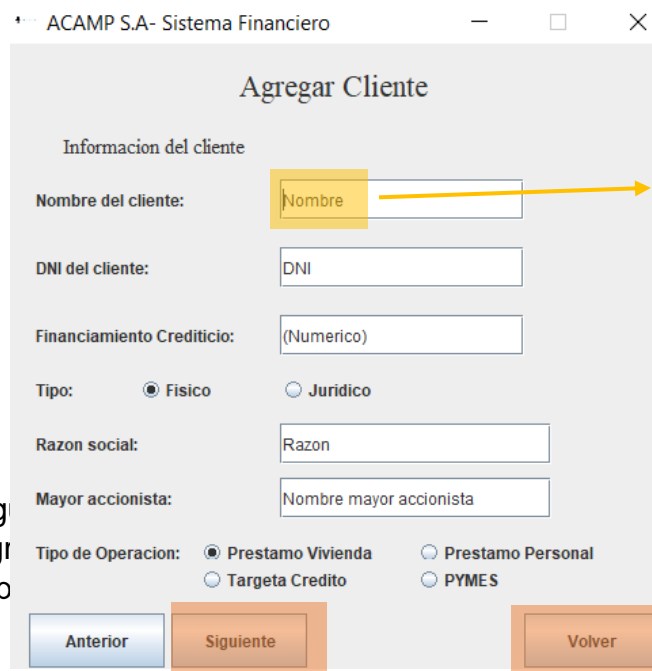
6. Corrida del programa con datos suministrados

1-Una vez iniciado el sistema le aparece un menú como este dispone de cinco

opciones:



2- Si selecciona la primera opción, dispondrá de un campo para Nombre, DNI, Financiamiento crediticio, tipo de identificación, el accionista mayor y se pregunta por el tipo de operación a realizar. En los campos se especifica que deberá rellena, a la hora de rellenar los campos deberá borrar el texto de ejemplo que aparece en el campo correspondiente. Al finalizar se presiona el botón de siguiente. El botón “Volver” hace que el sistema vuelva al menú principal.



Campo de ejemplo que debe ser sustituido con la información correspondiente.

3- Al presionar sig

el empleado
nombre e
ea, podrá regresar

al apartado anterior presionando al botón de “Anterior” pudiendo volver al ingreso del cliente inicial si se ha equivocado con algún dato.

ACAMP S.A- Sistema Financiero

Agregar Cliente

Informacion del Empleado

Nombre del empleado:

DNI del empleado:

Seccion al que pertenece:

☒ Servicio al cliente ☐ Plataforma ☐ Cajero

4- Si en el punto número dos, usted selecciono en el apartado de Tipo de operación, Préstamo de vivienda, se le desplegara los campos correspondientes para la solicitud de un préstamo de vivienda. Si en el campo del tipo de hipoteca, selecciona “Registral” o “Fiduciaria”, se le desplegara la información correspondiente a la selección.

ACAMP S.A- Sistema Financiero

Agregar Cliente

Prestamo Vivienda

Número de operacion:

Monto de crédito:

Tipo de Hipoteca:

☐ Fiduciaria ☒ Registral

Monto valor en colones:

Ubicacion:

Tamaño (M Cuadrados):

Tipo de Hipoteca:

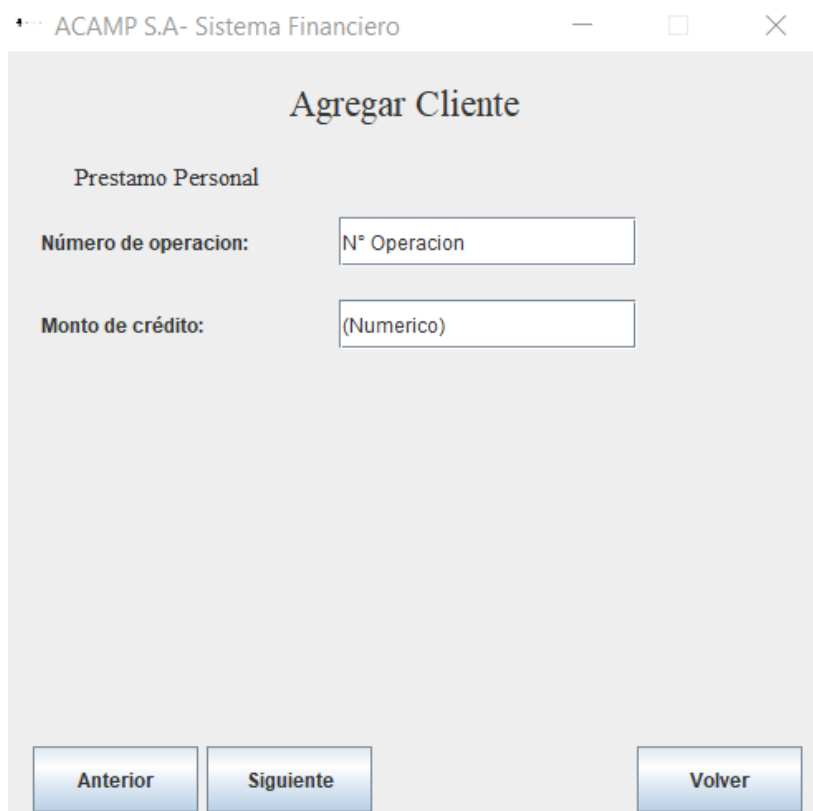
☒ Fiduciaria ☐ Registral

Nombre del fiduciario:

DNI del fiduciario:

Salario del fiduciario:

5- Si se seleccionó “Préstamo personal” en el punto número dos en la sección de Tipo de operación, se desplegará los campos de llenado para el numero de operación a realizar y el monto del crédito a solicitar.



ACAMP S.A- Sistema Financiero

Agregar Cliente

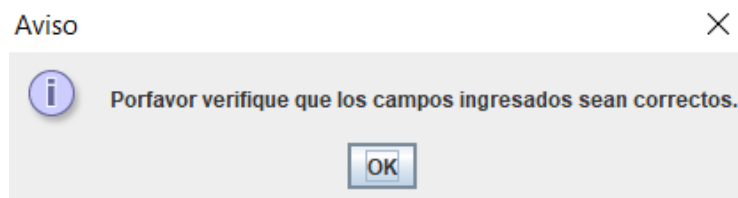
Prestamo Personal

Número de operacion:

Monto de crédito:

Anterior Siguiete Volver

6- Es importante aclar que, si el usuario que manipule el sistema, no ingresa los datos correctamente, el sistema lo alertara.



Aviso

i Porfavor verifique que los campos ingresados sean correctos.

OK

7- Si se seleccionó “Tarjeta de crédito” en el punto número dos en la sección de Tipo de operación, se desplegará los campos de llenado para el monto gastado por el cliente, el monto limite que se dispone en la tarjeta y el número de tarjeta.

ACAMP S.A- Sistema Financiero

Agregar Cliente

Targeta de credito

Monto Gastado: (Numerico)

Monto Limite: (Numerico)

Número de targeta: N° Targeta

Anterior Siguiente Volver

8- Por último, si se seleccionó “PYMES” en el punto número dos en la sección de Tipo de operación, se desplegará los campos de llenado para el número de operación y el monto de crédito a solicitar.

ACAMP S.A- Sistema Financiero

Agregar Cliente

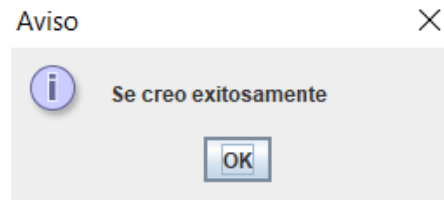
PYMES

Número de operacion: N° Operacion

Monto de crédito: (Numerico)

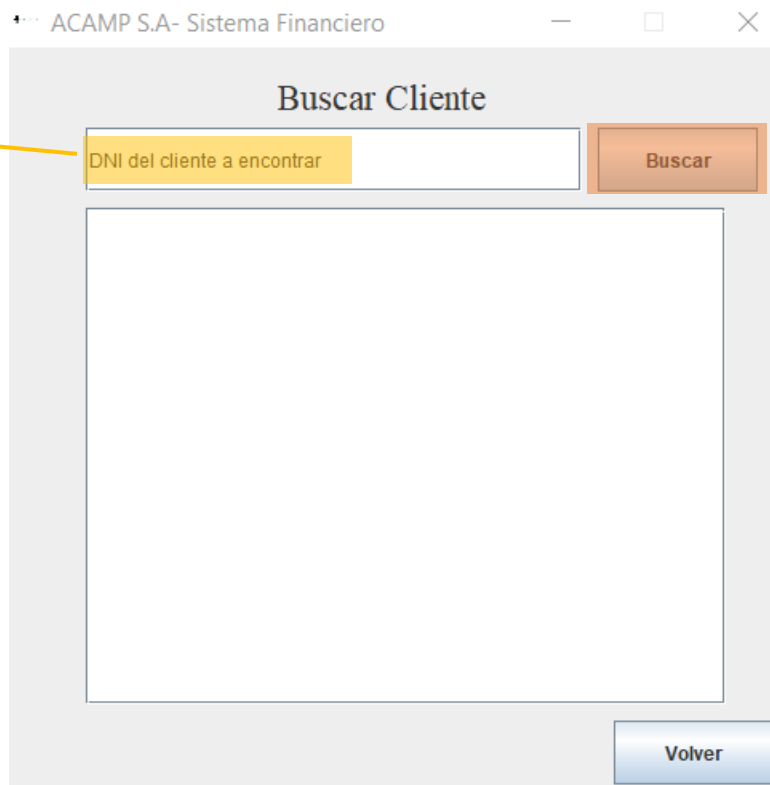
Anterior Siguiente Volver

9- Si todos los datos son correctos, se le informara al usuario.



10- Al finalizar, el sistema regresara al menú principal como se muestra en el punto número uno. Si se selecciona la opción dos, el cual corresponde a buscar un cliente por medio el DNI, se le desplegará un campo el cual usted deberá sustituir el texto de ejemplo por el DNI del cliente a buscar, luego de ingresar el DNI se Debra presionar el botón “Buscar”.

Campo donde se debe sustituir el texto de ejemplo por el DNI del cliente.



11- Si ingreso el DNI correctamente, se le desplegara la información correspondiente del cliente y de la operación realizada. Desplegada la información, el usuario podrá volver al menú principal.

ACAMP S.A.- Sistema Financiero

Buscar Cliente

29

Accionista mayor:Juan Carlos Rojas
Razon Social:Inversión
Informacion sobre operacion:
Empleado:
Nombre:Erick Murillo
DNI:15204
Seccion:Plataforma
Tipo de operacion:Prestamo Vivienda
Numero de operacion:2
Monto de credito:10000.0
Tasa de interes:9.0
Plazo en meses:84
Interes fijo PYMES:5%
Fiudiciaria:
Nombre:Kenneth S
DNI:34
Salario neto:1200000.0
Cuota a pagar:189.28571428571428
Saldo a pagar:9810.714285714286

12- Si selecciono la opción tres, la cual corresponde a “Ver todos los registros”, se desplegarán todos los registros de los clientes ingresados. Se cuenta con un botón de “Anterior” y “Siguiente” para visualizar todos los clientes. Si no se encuentra un registro anterior o un cliente siguiente, el sistema se lo comunicara.

ACAMP S.A.- Sistema Financiero

Ver Clientes

N° 3

Informacion del cliente:
Nombre:Cris Alvarez
DNI:29
Tipo:Fisico
Monto Financiamiento Credificio:100000.0
Accionista mayor:Juan Carlos Rojas
Razon Social:Inversión
Informacion sobre operacion:
Empleado:
Nombre:Erick Murillo
DNI:15204
Seccion:Plataforma
Tipo de operacion:Prestamo Vivienda
Numero de operacion:2
Monto de credito:10000.0
Tasa de interes:9.0
Plazo en meses:84
Interes fijo PYMES:5%
Fiudiciaria:
Nombre:Kenneth S
DNI:34

Aviso

No hay registros siguientes

Aviso

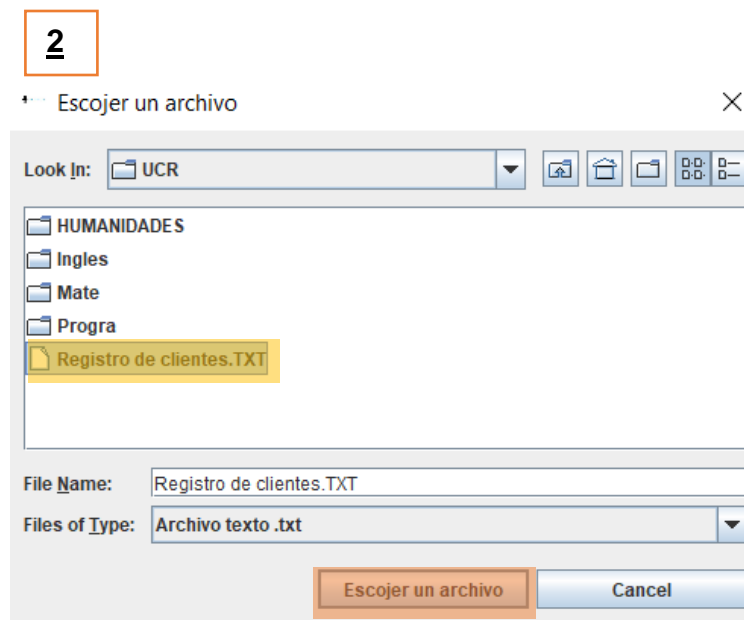
No hay registros anteriores

13- Si en el menú principal, se seleccionó la opción “Guardar lista” se desplegará un botón de “Buscar” el cual al presionar se mostrará los directorios de archivos de la computadora en el que este el sistema, se debe seleccionar un archivos.txt en el que será guardada la información. Seleccionado el directorio, se deberá presionar el botón “Exportar lista”.

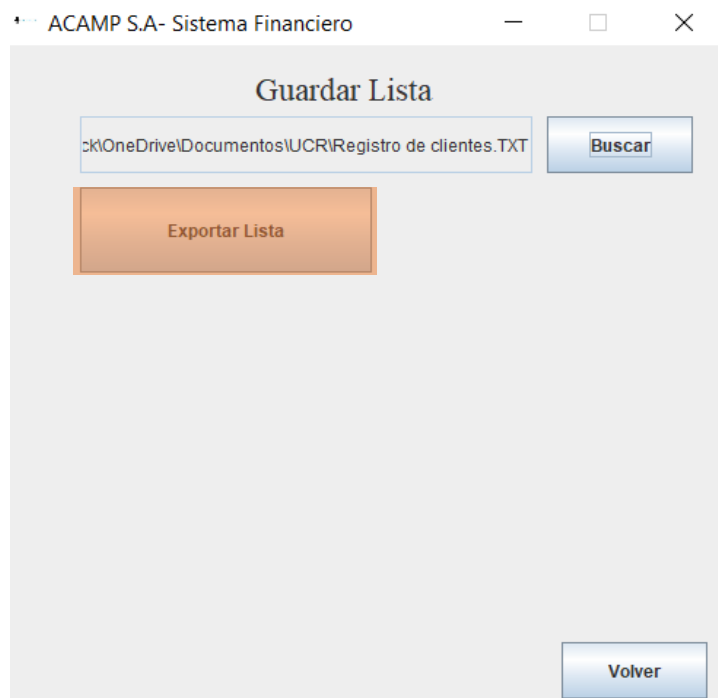
1



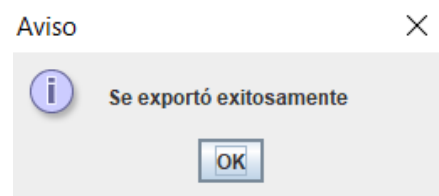
2



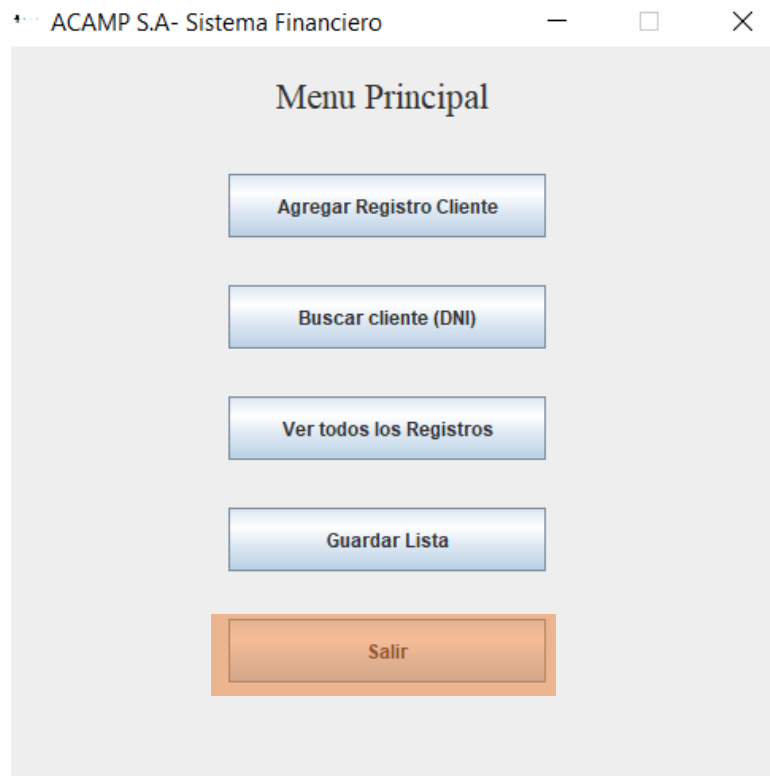
3



4



14- Por ultimo se cuenta con el botón de salir, si se presiona, el sistema finaliza.



7. Código fuente

7.1 Clase Persona.

```
package manage;

//Clase empleada en las clases "Cliente" "Empleado" "Fiduciaria"
public class Persona {

    //Atributos
    private String nombre;
    private String DNI;

    public Persona() {

    }

    //Metodos SETTERS & GETTERS
    public String getNombre() {
        return nombre;
    }

    public void setNombre(String nombre) {
        this.nombre = nombre;
    }

    public String getDNI() {
        return DNI;
    }

    public void setDNI(String dNI) {
        DNI = dNI;
    }

    //Se brinda toda la informacion de la persona
    public String personaInformacion() {
        return "\nNombre:"+getNombre()+"\nDNI:"+getDNI();
    }
}
```

7.2 Clase Cliente.

```
package manage;

//Clase cliente
public class Cliente extends Persona{

    //Atributos
    private String tipo;
    private String accionistaMayor;
    private String razonSocial;
    private double montoFinanciamiento;
    private Operaciones Operacion;//Agregacion de la clase abstracta
    Operaciones
```

```

public Cliente() {

}

//Metodos Setters & getters
public String getTipo() {
    return tipo;
}

public void setTipo(String tipo) {
    this.tipo = tipo;
}

public String getAccionistaMayor() {
    return accionistaMayor;
}

public void setAccionistaMayor(String accionistaMayor) {
    this.accionistaMayor = accionistaMayor;
}

public String getRazonSocial() {
    return razonSocial;
}

public void setRazonSocial(String razonSocial) {
    this.razonSocial = razonSocial;
}

public double getMontoFinanciamiento() {
    return montoFinanciamiento;
}

public void setMontoFinanciamiento(double montoFinanciamiento) {
    this.montoFinanciamiento = montoFinanciamiento;
}

public Operaciones getOperacion() {
    return Operacion;
}

public void setTipoOperacion(Operaciones tipoOperacion) {
    this.Operacion = tipoOperacion;
}

//Se da la informacion tanto del cliente, como tambien la referente
a la operacion escogida
public String brindarInformacion() {
    return "\tInformacion del
cliente:"+personaInformacion()+"\nTipo:"+getTipo()
        +"\nMonto Financiamiento
Credificio:"+getMontoFinanciamiento()+"\nAccionista
mayor:"+getAccionistaMayor()
        +"\nRazon Social:"+getRazonSocial()+"\n\tInformacion sobre
operacion:"+getOperacion().desplegarInformacion();
}

```

```
}
```

7.3 Clase Empleado.

```
package manage;

//Agregacion de la clase Operaciones
public class Empleado extends Persona{

    //Atributos
    private String seccion;

    public Empleado() {

    }

    //Metodos Setters & getters
    public String getSeccion() {
        return seccion;
    }

    public void setSeccion(String seccion) {
        this.seccion = seccion;
    }

    //Se brindan todos los atributos de esta clase
    public String empleadoInformacion() {
        return personaInformacion()+"\nSeccion:"+getSeccion();
    }

}
```

7.4 Clase Operaciones.

```
package manage;

//SuperClase abstracta Operaciones
public abstract class Operaciones {

    //Atributos
    Empleado empleado;//Agregacion de empleado

    public Operaciones() {
        empleado=new Empleado();//Se crea un empleado
    }

    //Metodos Setters & getters
    public Empleado getEmpleado() {
        return empleado;
    }

    public void setEmpleado(Empleado empleado) {
        this.empleado=empleado;
    }

    //Metodos abstractos
```

```

        public abstract double calcularSaldo();//Metodo que sera
sobrescrito en las clases hijas el cual contendra el saldo a pagar
        public abstract double calcularCuota();//Metodo que sera
sobrescrito en las clases hijas el cual contendra la cuota a pagar

        //Este metodo sera sobrescrito en las clases hijas las cuales
añadiran los atributos a imprimir
        public String desplegarInformacion() {
            return "\nEmpleado:"+empleado.empleadoInformacion();
        }

    }

```

7.5 Clase Prestamos.

```

package manage;

//Clase abstracta empleada en las clases "PYMES" "PERSONAL" "VIVIENDA"
public abstract class Prestamo extends Operaciones {

    //Atributos
    private final double tasaInteres=0.09;
    private final int plazoMeses=84;
    private double montoCredito,montoCredificio;
    private int operacionNumero;
    private String operacion;

    public Prestamo() {

    }

    //Metodos SETTERS & GETTERS
    public double getMontoCredificio() {
        return montoCredificio;
    }

    public void setMontoCredificio(double montoCredificio) {
        this.montoCredificio = montoCredificio;
    }

    public String getOperacion() {
        return operacion;
    }

    public void setOperacion(String operacion) {

```

```

        this.operacion = operacion;
    }

    public double getMontoCredito() {
        return montoCredito;
    }

    public void setMontoCredito(double montoCredito) {
        this.montoCredito = montoCredito;
    }

    public int getOperacionNumero() {
        return operacionNumero;
    }

    public void setOperacionNumero(int operacionNumero) {
        this.operacionNumero = operacionNumero;
    }

    public double getTasaInteres() {
        return tasaInteres;
    }

    public int getPlazoMeses() {
        return plazoMeses;
    }

    /*Metodo sobrescrito "desplegarInformacion()" se añaden los
    atributos de esta clase
    * */
    @Override
    public String desplegarInformacion() {
        return super.desplegarInformacion()+"\nTipo de
operacion:"+getOperacion()+"\nNumero de operacion:"+getOperacionNumero()
        +"\nMonto de credito:"+getMontoCredito()+"\nTasa de
interes:"+getTasaInteres()*100+"\nPlazo en meses:"+getPlazoMeses();
    }

}

```

7.6 Clase Tarjeta.

```

package manage;

//Clase de operacion targeta
public class Tarjeta extends Operaciones {

    //Atributos
    private final double tasaInteres=0.55;
    private final int plazoCreditoFijo=72;
    private double montoGastado,montoLimite;
    private String targetaNumero;

    public Tarjeta() {

```

```

    }

    //Metodos SETTERS & GETTERS
    public double getMontoGastado() {
        return montoGastado;
    }

    public void setMontoGastado(double montoGastado) {
        this.montoGastado = montoGastado;
    }

    public double getMontoLimite() {
        return montoLimite;
    }

    public void setMontoLimite(double montoLimite) {
        this.montoLimite = montoLimite;
    }

    public String getTargetNumero() {
        return targetNumero;
    }

    public void setTargetNumero(String targetNumero) {
        this.targetNumero = targetNumero;
    }

    public double getTasaInteres() {
        return tasaInteres;
    }

    public int getPlazoCreditoFijo() {
        return plazoCreditoFijo;
    }

    /*Metodo sobrescrito "desplegarInformacion()" se añaden los
    atributos de esta clase asi como
    * tambien los valores generados en los metodos sobrescritos
    "calcularCuota" y "calcularSaldo"
    * */
    @Override
    public String desplegarInformacion() {
        return super.desplegarInformacion()+"\nTipo de
operacion:Operacion de Tarjeta de Credito"+"
\nTasa
interes:"+getTargetNumero()+"\nMonto limite:"+getMontoLimite()+
"\nTasa de Interes:55%\nPlazo Credito
Fijo:"+getPlazoCreditoFijo()+"\nMonto Gastado:"+getMontoGastado()+
"\nCuota a pagar:"+calcularCuota()+"\nSaldo a
pagar:"+calcularSaldo();
    }

    /*Se sobrescribe el metodo "calcularCuota" de la super clase
    "Operaciones" con los siguientes criterios

```

```

        * (Monto gastado+(Monto gastado*Tasa de interés))/Plazo de crédito
fijo
    * */
@Override
public double calcularCuota() {
    double total=0;

    total+=getMontoGastado()+(getMontoGastado()*getTasaInteres());
    total=total/getPlazoCreditoFijo();
    return total;
}

/*Se sobrescribe el metodo "calcularSaldo" de la super clase
"Operaciones" con los siguientes criterios
    * Monto gastado - Cuota a pagar
    * */
@Override
public double calcularSaldo() {
    return getMontoGastado()-calcularCuota();
}

}

```

7.7 Clase PYMES.

```

package manage;

//Clase Operacion PYMES
public class PYMES extends Prestamo{

    //Atributos
    private final double interesFijo=0.05;

    public PYMES() {
        //Se especifica la operacion, este valor se guarda en un
        atributo perteneciente a la clase "Prestamo"
        super.setOperacion("PYMES");
    }

    //Metodos SETTER & GETTER
    public double getInteresFijo() {
        return interesFijo;
    }

    /*Metodo sobrescrito "desplegarInformacion()" se añaden los
    atributos de esta clase así como
        * tambien los valores generados en los metodos sobrescritos
    "calcularCuota" y "calcularSaldo"
    * */
    @Override
    public String desplegarInformacion() {
        if(super.getOperacion().equalsIgnoreCase("PYMES")) {
            return super.desplegarInformacion()+"\nInteres fijo
PYMES:5%\nCuota a pagar:"+calcularCuota()+"\nSaldo a
pagar:"+calcularSaldo();

```



```

        }else {
            return super.desplegarInformacion()+"\nInteres fijo
PYMES:5%";
        }
    }

    /*Se sobrescribe el metodo "calcularCuota" de la super clase
    "Operaciones" con los siguientes criterios
    * (Monto de crédito + (Monto de financiamiento crediticio*interés
    fijo) + (Monto de crédito*tasa de interés))/plazo en meses
    * */
    @Override
    public double calcularCuota() {
        double total=0;
        total+=super.getMontoCredito();
        total+=super.getMontoCredificio()*getInteresFijo();
        total+=super.getMontoCredito()*super.getTasaInteres();
        total=total/super.getPlazoMeses();

        return total;
    }

    /*Se sobrescribe el metodo "calcularSaldo" de la super clase
    "Operaciones" con los siguientes criterios
    * Monto de crédito - Cuota a pagar
    * */
    @Override
    public double calcularSaldo() {
        double total=0;
        total=super.getMontoCredito()-calcularCuota();
        return total;
    }

}

```

7.8 Clase Vivienda.

```

package manage;

//Clase Operacion vivienda
public class Vivienda extends PYMES{

    //Atributos
    private String tipoHipoteca;
    Fiduciaria fiduciaria;
    Registral registral;

    //Se determina el tipo de hipoteca
    public Vivienda(int type) {
        if(type==1) {
            fiduciaria=new Fiduciaria();
            setTipoHipoteca("Fiduciaria");
        }else {
            registral=new Registral();
            setTipoHipoteca("Registral");
        }
    }
}

```

```

    }

}

//Metodos SETTERS & GETTERS
public Fiduciaria getFiduciaria() {
    return fiduciaria;
}

public Registral getRegistral() {
    return registral;
}

public String getTipoHipoteca() {
    return tipoHipoteca;
}

public void setTipoHipoteca(String tipoHipoteca) {
    this.tipoHipoteca = tipoHipoteca;
}

/*Metodo sobrescrito "desplegarInformacion()" se añaden los
atributos de esta clase asi como
* tambien los valores generados en los metodos sobrescritos
"calcularCuota" y "calcularSaldo"
* pertenecientes a la clase Padre "PYMES"
* */
@Override
public String desplegarInformacion() {
    String toPresent="";
    super.setOperacion("Prestamo Vivienda");

    if(getTipoHipoteca().equalsIgnoreCase("Fiduciaria")) {
        toPresent=fiduciaria.informacionFiduciaria();
    }else {
        toPresent=registral.informacionRegistral();
    }

    return super.desplegarInformacion()+toPresent+"\nCuota a
pagar:"+super.calcularCuota()+"\nSaldo a pagar:"+super.calcularSaldo();
}

}

```

7.9 Clase Personal.

```

package manage;

//Clase Operacion Prestamo Personal
public class Personal extends Prestamo{

    public Personal() {

    }

}

```

```

        /*Metodo sobrescrito "desplegarInformacion()" se añaden los
        atributos de esta clase asi como
        * tambien los valores generados en los metodos sobrescritos
        "calcularCuota" y "calcularSaldo"
        * */
        @Override
        public String desplegarInformacion() {
            super.setOperacion("Prestamo Personal");
            return super.desplegarInformacion()+"\nCuota a
            pagar:"+calcularCuota()+"\nSaldo a pagar:"+calcularSaldo();
        }

        /*Se sobrescribe el metodo "calcularCuota" de la super clase
        "Operaciones" con los siguientes criterios
        * (Monto de crédito + (Monto de crédito*tasa de interés))/plazo en
        meses
        * */
        @Override
        public double calcularCuota() {
            double total=0;
            total+=super.getMontoCredito();
            total+=super.getMontoCredito()*super.getTasaInteres();
            total=total/super.getPlazoMeses();
            return total;
        }

        /*Se sobrescribe el metodo "calcularSaldo" de la super clase
        "Operaciones" con los siguientes criterios
        * Monto de crédito - Cuota a pagar
        * */
        @Override
        public double calcularSaldo() {
            double total=0;
            total=super.getMontoCredito()-calcularCuota();
            return total;
        }
    }
}

```

7.10 Clase Fiduciaria.

```

package manage;

//Tipo de Hipoteca "Fiduciaria"
public class Fiduciaria extends Persona{
    //Atributos
    private double salarioNeto;

    public Fiduciaria() {

    }

    //Metodos SETTERS & GETTERS
    public double getSalarioNeto() {
        return salarioNeto;
    }
}

```

```

    public void setSalarioNeto(double salarioNeto) {
        this.salarioNeto=salarioNeto;
    }

    //Se brindan todos los atributos de esta clase
    public String informacionFiduciaria() {
        return "\nFiudiciaria:"+personaInformacion()+"\nSalario
neto:"+getSalarioNeto();
    }

}

```

7.11 Clase Registral.

```

package manage;

//Tipo de Hipoteca Registral
public class Registral {

    //Atributos
    private double valorRegistral,sizeMCuadrados;
    private String ubicacion;

    public Registral() {

    }

    //Metodos SETTERS & GETTERS
    public double getValorRegistral() {
        return valorRegistral;
    }

    public double getSizeMCuadrados() {
        return sizeMCuadrados;
    }

    public void setSizeMCuadrados(double sizeMCuadrados) {
        this.sizeMCuadrados = sizeMCuadrados;
    }

    public void setValorRegistral(double valorRegistral) {
        this.valorRegistral = valorRegistral;
    }
}

```

```

    public String getUbicacion() {
        return ubicacion;
    }

    public void setUbicacion(String ubicacion) {
        this.ubicacion = ubicacion;
    }

    //Brinda todos los atributos de Registral
    public String informacionRegistral() {
        return "\nRegistral:\nUbicacion:"+getUbicacion()+"\nValor
registral (colones):"+getValorRegistral()+
        "\nTamaño:"+getSizeMCuadrados();
    }
}

```

7.12 Clase Interfaz.

```

package pp2_gr1_cris_taisigue_erick_vasquez;

import javax.swing.*;
import javax.swing.filechooser.FileNameExtensionFilter;

import java.awt.Font;
import java.awt.event.*;
import java.io.*;
import java.util.ArrayList;

public class Interfaz extends JFrame{

    final int ANCHO=500,ALTO=500;
    final String
    directorio="src/pp2_gr1_cris_taisigue_erick_vasquez/Acamp19.dat";
    int siguiente=0;
    JPanel panel=new JPanel();
    JLabel subTitle=new JLabel();
    JButton back=new JButton("Volver");
    MenuPrincipal mainMenu;
    VerTodos ver;
    BuscarDNI buscar;
    Guardar guardar;
    Crear crear;
    Font commonFont=new Font("TimesRoman",Font.PLAIN,22);
    Cliente registro[];
    ManipularDatos data;

    public Interfaz() {
        data=new ManipularDatos();
    }
}

```

```

siguiente=data.maximoRango();
subTitle.setBounds(170,10,200,40);
subTitle.setFont(commonFont);
back.setVisible(false);
back.setBounds(380,420,100,40);
back.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        changer(5);
    }
});

this.setTitle("ACAMP S.A- Sistema Financiero");
this.setLocationRelativeTo(null);
this.setResizable(false);
this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
this.setSize(ANCHO,ALTO);
panel.setLayout(null);
panel.add(subTitle);
panel.add(back);
this.add(panel);
this.setVisible(true);
buscar=new BuscarDNI();
mainMenu=new MenuPrincipal();
ver=new VerTodos();
guardar=new Guardar();
crear=new Crear();
inicio();
}

//Metodo para realizar notificaciones al usuario
public void mensajeDialogo(String text,int type) {
    if(type==1) {

        JOptionPane.showMessageDialog(null,text,"Aviso",JOptionPane.INFORMA
TION_MESSAGE);
    }else {

        JOptionPane.showMessageDialog(null,text,"Aviso",JOptionPane.WARNING
_MESSAGE);
    }
}

//Metodo iniciado en la construccion de la clase principal Interfaz
public void inicio() {
    mainMenu.toggleComponents(true);
}

//Muestra los distintos subMenus en el sistema
public void changer(int ax) {
    switch(ax) {

        case 1:{
            mainMenu.toggleComponents(false);
            crear.crearToggle(true,1);
            back.setVisible(true);
        }break;

```

```

        case 2:{
            mainMenu.toggleComponents(false);
            buscar.toggleComponents(true);
            back.setVisible(true);
        }break;

        case 3:{
            mainMenu.toggleComponents(false);
            ver.toggleComponents(true);
            back.setVisible(true);
        }break;

        case 4:{
            mainMenu.toggleComponents(false);
            guardar.toggleComponents(true);
            back.setVisible(true);
        }break;

        case 5:{
            ver.toggleComponents(false);
            mainMenu.toggleComponents(true);
            crear.crearToggle(false,92);
            buscar.toggleComponents(false);
            guardar.toggleComponents(false);
            back.setVisible(false);
            crear.resetCrear();
            buscar.resetSearch();
            ver.resetVer();
            guardar.resetExportar();
        }break;
    }
}

//Crear SubMenu
class Crear{

    JLabel subMenu=new JLabel();
    ClienteMenu menuCliente;
    JButton next=new JButton("Siguiente");
    JButton anterior=new JButton("Anterior");
    EmpleadoInfo empleado;
    PrestamoInfo prest;
    TargetaOperacion targeta;

    //Clase principal para la agregacion de registros
    public Crear() {
        subMenu.setFont(new Font("TimesRoman",Font.PLAIN,15));
        subMenu.setBounds(40,60,200,30);
        next.setBounds(120,420,100,40);
        anterior.setBounds(15,420,100,40);
        menuCliente=new ClienteMenu();
        empleado=new EmpleadoInfo();
        prest=new PrestamoInfo();
        targeta=new TargetaOperacion();
        crearToggle(false,92);
    }
}

```

```

        panel.add(subMenu);
        panel.add(next);
        panel.add(anterior);
        next.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                String text=subMenu.getText();
                if(text.equalsIgnoreCase("Informacion del
cliente")) {
                    crearSwitcher(2);
                }else if(text.equalsIgnoreCase("Informacion
del Empleado")) {
                    crearSwitcher(3);
                }else {
                    crearSwitcher(4);
                }
            }
        });

        anterior.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                String text=darOperacion();
                if(text.equalsIgnoreCase("Informacion del
Empleado")) {
                    crearSwitcher(1);
                }else if(text.equalsIgnoreCase("Targeta de
credito")) {
                    crearSwitcher(2);
                }else if(text.equalsIgnoreCase("Prestamo
Personal")) {
                    crearSwitcher(2);
                }else if(text.equalsIgnoreCase("PYMES")) {
                    crearSwitcher(2);
                }else if(text.equalsIgnoreCase("Prestamo
Vivienda")) {
                    crearSwitcher(2);
                }
            }
        });
    }

    //Resetea los campos subMenus en la clase Crear
    public void resetCrear() {
        prest.prestReset();
        menuCliente.resetClient();
        empleado.resetEm();
        targeta.resetT();
    }

    //Dar los distintos subMenus en la clase Crear
    public String darOperacion() {
        return subMenu.getText();
    }

    public ClienteMenu giveClientM() {
        return menuCliente;
    }

```



```

public EmpleadoInfo giveEmpleado() {
    return empleado;
}

public PrestamoInfo givePrestamo() {
    return prest;
}

public TargetaOperacion giveTargeta() {
    return targeta;
}

//Metodo para la manipulacion de la visibilidad del menu
Agregar registro del cliente
public void crearToggle(boolean cond,int ox) {
    if(cond) {
        subTitle.setText("Agregar Cliente");
    }

    subMenu.setVisible(cond);
    next.setVisible(cond);
    anterior.setVisible(cond);
    crearSwitcher(ox);
}

//Metodo para la manipulacion de los distintos subMenus de la
clase Crear
public void crearSwitcher(int op) {
    switch(op) {

        //Muestra los campos correspondientes al cliente
        case 1:{
            menuCliente.toggle(true);
            empleado.toggle(false);
            prest.tooglePrest(false);
            targeta.toggle(false);
        }break;

        //Desplegar el subMenu de Empleado
        case 2:{
            menuCliente.toggle(false);
            empleado.toggle(true);
            prest.tooglePrest(false);
            targeta.toggle(false);
        }break;

        //Desplegar los campos de Prestamos o Targeta
        case 3:{
            menuCliente.toggle(false);
            empleado.toggle(false);

            if(menuCliente.radioCheck() !=4) {
                prest.tooglePrest(true);
            }

            prest.changerP(menuCliente.radioCheck());
        }else {

```

```

        targeta.toggle(true);
    }

    }break;

    case 4:{
        //Intentar añadir un nuevo registro
        if(data.agregar(crear)) {
            mensajeDialogo("Se creo
exitosamente",1);

            prest.tooglePrest(false);
            targeta.toggle(false);
            subMenu.setVisible(false);
            next.setVisible(false);
            back.setVisible(false);
            anterior.setVisible(false);
            resetCrear();
            mainMenu.toggleComponents(true);
        }else {
            mensajeDialogo("Porfavor verifique que
los campos ingresados sean correctos.",1);
        }

    }break;

    //Esconde todos los subMenu de la clase Crear
    case 92:{

        menuCliente.toggle(false);
        empleado.toggle(false);
        prest.tooglePrest(false);
        targeta.toggle(false);
    }break;
    }

    //SubMenu(Crear) ClienteMenu
    class ClienteMenu{
        JTextField nombreC=new JTextField("Nombre");
        JTextField dniC=new JTextField("DNI");
        JTextField montoFinanciamiento=new
JTextField("(Numerico)");
        JTextField razonC=new JTextField("Razon");
        JTextField majorC=new JTextField("Nombre mayor
accionista");

        JLabel nLabel=new JLabel("Nombre del cliente:");
        JLabel dLabel=new JLabel("DNI del cliente:");
        JLabel montoLabel=new JLabel("Financiamiento
Crediticio:");

        JLabel tLabel=new JLabel("Tipo:");
        JLabel major=new JLabel("Mayor accionista:");
        JLabel razon=new JLabel("Razon social:");
        ButtonGroup grupoRadio=new ButtonGroup();
        JRadioButton fisico=new JRadioButton("Fisico",true);

```

```

JRadioButton juridico=new JRadioButton("Juridico");

ButtonGroup operacion=new ButtonGroup();
JLabel toLabel=new JLabel("Tipo de Operacion:");
JRadioButton vivienda=new JRadioButton("Prestamo
Vivienda",true);
JRadioButton personal=new JRadioButton("Prestamo
Personal");
JRadioButton PYMES=new JRadioButton("PYMES");
JRadioButton targeta=new JRadioButton("Targeta
Credito");

public ClienteMenu() {
    nLabel.setBounds(20,100,200,30);
    nombreC.setBounds(200,100,180,30);

    dLabel.setBounds(20,150,200,30);
    dniC.setBounds(200,150,180,30);

    montoLabel.setBounds(20,200,200,30);
    montoFinanciamiento.setBounds(200,200,180,30);

    tLabel.setBounds(20,240,70,30);
    fisico.setBounds(95, 240, 100, 30);
    juridico.setBounds(200, 240, 100, 30);

    razon.setBounds(20,280,100,30);
    razonC.setBounds(200,280,200,30);
    major.setBounds(20,320,100,30);
    majorC.setBounds(200,320,200,30);

    toLabel.setBounds(20,360,130,30);
    vivienda.setBounds(140,360,150,30);
    personal.setBounds(300,360,150,30);
    targeta.setBounds(140,380,150,30);
    PYMES.setBounds(300,380,150,30);

    toggle(false);
    panel.add(toLabel);
    panel.add(nombreC);
    panel.add(dniC);
    panel.add(montoFinanciamiento);
    panel.add(dLabel);
    panel.add(nLabel);
    panel.add(montoLabel);
    panel.add(tLabel);
    panel.add(razon);
    panel.add(razonC);
    panel.add(major);
    panel.add(majorC);

    panel.add(PYMES);

```

```

        panel.add(vivienda);
        panel.add(personal);
        panel.add(targeta);

        grupoRadio.add(fisico);
        grupoRadio.add(juridico);

        operacion.add(PYMES);
        operacion.add(personal);
        operacion.add(vivienda);
        operacion.add(targeta);

        panel.add(fisico);
        panel.add(juridico);
    }

    //Cambiar visibilidad de los componentes de esta clase
    public void toggle(boolean cond) {
        if(cond) {
            subMenu.setText("Informacion del cliente");
        }
        toLabel.setVisible(cond);
        nombreC.setVisible(cond);
        dniC.setVisible(cond);
        montoFinanciamiento.setVisible(cond);
        nLabel.setVisible(cond);
        dLabel.setVisible(cond);
        montoLabel.setVisible(cond);
        tLabel.setVisible(cond);
        razon.setVisible(cond);
        razonC.setVisible(cond);
        major.setVisible(cond);
        majorC.setVisible(cond);
        fisico.setVisible(cond);
        juridico.setVisible(cond);
        vivienda.setVisible(cond);
        targeta.setVisible(cond);
        personal.setVisible(cond);
        PYMES.setVisible(cond);
    }

    //Resetear los campos de la clase ClienteMenu
    public void resetClient() {
        nombreC.setText("Nombre");
        dniC.setText("DNI");
        montoFinanciamiento.setText("(Numerico)");
        razonC.setText("Razon");
        majorC.setText("Nombre mayor accionista");
    }

    //Dar tipo de cliente
    public String clienteCheck() {
        if(juridico.isSelected()) {
            return "Juridico";
        }else {
            return "Fisico";
        }
    }

```

```

    }

    //Dar tipo de operacion
    public int radioCheck() {
        int check=0;

        if(PYMES.isSelected()) {
            check=1;
        }else if(personal.isSelected()) {
            check=2;
        }else if(vivienda.isSelected()) {
            check=3;
        }else if(targeta.isSelected()) {
            check=4;
        }

        return check;
    }

    //Dar campos de texto de la clase ClienteMenu
    public String darNombreCliente() {
        return nombreC.getText();
    }

    public String darDniCliente() {
        return dniC.getText();
    }

    public String darMontoF() {
        return montoFinanciamiento.getText();
    }

    public String darRazon() {
        return razonC.getText();
    }

    public String darMayorA() {
        return majorC.getText();
    }

}

//subMenu(Crear) clase EmpleadoInfo
class EmpleadoInfo{
    JLabel nLabel=new JLabel("Nombre del empleado:");
    JLabel dLabel=new JLabel("DNI del empleado:");
    JTextField nombreE=new JTextField("Nombre");
    JTextField dniE=new JTextField("DNI");
    ButtonGroup seccion=new ButtonGroup();
    JRadioButton platOne=new JRadioButton("Servicio al
cliente",true);
    JRadioButton platTwo=new JRadioButton("Plataforma");
    JRadioButton plaThree=new JRadioButton("Cajero");
    JLabel tipe=new JLabel("Seccion al que pertenece:");

    public EmpleadoInfo() {

```

```

        nLabel.setBounds(20,100,200,30);
        nombreE.setBounds(200,100,180,30);
        dLabel.setBounds(20,150,200,30);
        dniE.setBounds(200,150,180,30);
        tipe.setBounds(20,220,180,30);
        platOne.setBounds(50,280,150,30);
        platTwo.setBounds(210,280,150,30);
        plaThree.setBounds(370,280,80,30);
        toggle(false);
        panel.add(nLabel);
        panel.add(nombreE);
        panel.add(dLabel);
        panel.add(dniE);
        panel.add(tipe);
        panel.add(plaThree);
        panel.add(platOne);
        panel.add(platTwo);
        seccion.add(plaThree);
        seccion.add(platTwo);
        seccion.add(platOne);
    }

    //Resetear los campos pertenecientes a la clase de
Empleado
    public void resetEm() {
        nombreE.setText("Nombre");
        dniE.setText("DNI");
    }

    public void toggle(boolean cond) {
        if(cond) {
            subMenu.setText("Informacion del Empleado");
        }

        nLabel.setVisible(cond);
        nombreE.setVisible(cond);
        dniE.setVisible(cond);
        dLabel.setVisible(cond);
        platOne.setVisible(cond);
        platTwo.setVisible(cond);
        plaThree.setVisible(cond);
        tipe.setVisible(cond);
    }

    //Dar la seccion del empleado
    public String checkEmpleado() {
        if(platOne.isSelected()) {
            return "Servicio al cliente";
        }else if(platTwo.isSelected()) {
            return "Plataforma";
        }else{
            return "Cajero";
        }
    }
}

```

```

//Dar los campos del submenu Empleado
public String darNombreEmpleado() {
    return nombreE.getText();
}

public String darDniEmpleado() {
    return dniE.getText();
}

}

//submenu(crear) operacion clase TargetaOperacion
class TargetaOperacion {

    JLabel moLabel=new JLabel("Monto Gastado:");
    JLabel mlLabel=new JLabel("Monto Limite:");
    JLabel nuLabel=new JLabel("Número de targeta:");

    JTextField montoG=new JTextField("(Numerico)");
    JTextField montoL=new JTextField("(Numerico)");

    JTextField targetNum=new JTextField("N° Targeta");

    public TargetaOperacion() {

        moLabel.setBounds(20,100,200,30);
        montoG.setBounds(200,100,180,30);
        mlLabel.setBounds(20,150,200,30);
        montoL.setBounds(200,150,180,30);
        nuLabel.setBounds(20,250,200,30);
        targetNum.setBounds(200,250,180,30);
        toggle(false);
        panel.add(mlLabel);
        panel.add(moLabel);
        panel.add(montoG);
        panel.add(nuLabel);
        panel.add(targetNum);
        panel.add(montoL);
    }

}

//Cambiar la visibilidad de los componentes en esta
clase
public void toggle(boolean cond) {
    if(cond) {
        subMenu.setText("Targeta de credito");
    }

    moLabel.setVisible(cond);
    montoG.setVisible(cond);
    mlLabel.setVisible(cond);
    montoL.setVisible(cond);
}

```

```

        nuLabel.setVisible(cond);
        targetNum.setVisible(cond);
    }

    //Resetar los campos pertencientes a la clase
TargetaOperacion
    public void resetT() {
        montoG.setText("(Numerico)");
        montoL.setText("(Numerico)");
        targetNum.setText("N° Targeta");
    }

    //Dar los campos del subMenu Targeta
    public String darMontoG() {
        return montoG.getText();
    }

    public String darMontoL() {
        return montoL.getText();
    }

    public String darTargetNum() {
        return targetNum.getText();
    }
}

//subMenu(crear) Operaciones:PYMES,Prestamo Personal,
Prestamo Vivienda
class PrestamoInfo {

    JTextField number=new JTextField("N° Operacion");
    JTextField montoCredito=new JTextField("(Numerico)");
    JLabel nLabel=new JLabel("Número de operacion:");
    JLabel dLabel=new JLabel("Monto de crédito:");
    Vivienda vivienda=new Vivienda();

    public PrestamoInfo() {
        nLabel.setBounds(20,100,200,30);
        number.setBounds(200,100,180,30);
        dLabel.setBounds(20,150,200,30);
        montoCredito.setBounds(200,150,180,30);
        tooglePrest(false);
        panel.add(montoCredito);
        panel.add(dLabel);
        panel.add(nLabel);
        panel.add(number);
    }

    //Resetear los campos de esta Clase(Prestamo)
    public void prestReset() {
        number.setText("N° Operacion");
        montoCredito.setText("(Numerico)");
        vivienda.resetV();
    }
}

```



```

    }

    //Mostrar los distintos subMenus de la clase Prestamo
    public void changerP(int aux) {
        switch(aux) {

            case 1:{
                subMenu.setText("PYMES");
            }break;

            case 2:{
                subMenu.setText("Prestamo Personal");
            }break;

            case 3:{
                vivienda.toggle(true);
            }break;

        }
    }

    //Cambiar visibilidad de los campos de esta
    Clase(Prestamo)
    public void togglePrest(boolean cond) {
        nLabel.setVisible(cond);
        dLabel.setVisible(cond);
        montoCredito.setVisible(cond);
        number.setVisible(cond);

        if(cond==false) {
            vivienda.toggle(cond);
        }
    }

    //Dar campos del subMenu prestamo
    public String darNumeroOp() {
        return number.getText();
    }

    public String darMontoCredito() {
        return montoCredito.getText();
    }

    public Vivienda giveVienda() {
        return vivienda;
    }

    //SubMenu(Prestamos) operacion, clase vivienda
    class Vivienda{

        JLabel aLabel=new JLabel("Nombre del
fiduciario:");

        JLabel cLabel=new JLabel("DNI del fiduciario:");
        JLabel sLabel=new JLabel("Salario del
fiduciario:");

        JTextField nombreF=new JTextField("Nombre");

```

```

        JTextField dniF=new JTextField("DNI");
        JTextField Salario=new JTextField("(Numerico)");
        JTextField montoValor=new
JTextField("(Numerico)");
        JTextField ubicacion=new JTextField("Ubicacion");
        JTextField tama=new JTextField("(Numerico)");
        JLabel mLabel=new JLabel("Monto valor en
colones:");

        JLabel uLabel=new JLabel("Ubicacion:");
        JLabel tLabel=new JLabel("Tamaño (M Cuadrados):");
        ButtonGroup tipoH=new ButtonGroup();
        JRadioButton fidu=new JRadioButton("Fiduciaria");
        JRadioButton rent=new
JRadioButton("Registral",true);
        JLabel tiper=new JLabel("Tipo de Hipoteca:");

    public Vivienda() {

        rent.setBounds(200,235,100,30);
        fidu.setBounds(80,235,100,30);
        tiper.setBounds(50,200,100,30);
        aLabel.setBounds(50,270,150,30);
        nombreF.setBounds(200,270,130,30);
        cLabel.setBounds(50,315,150,30);
        dniF.setBounds(200,315,130,30);
        sLabel.setBounds(50,350,150,30);
        Salario.setBounds(200,350,130,30);

        mLabel.setBounds(50,270,150,30);
        montoValor.setBounds(200,270,130,30);
        uLabel.setBounds(50,315,150,30);
        ubicacion.setBounds(200,315,130,30);
        tLabel.setBounds(50,350,150,30);
        tama.setBounds(200,350,130,30);

        toggle(false);
        panel.add(mLabel);
        panel.add(montoValor);
        panel.add(uLabel);
        panel.add(ubicacion);
        panel.add(tLabel);
        panel.add(tama);

        panel.add(aLabel);
        panel.add(cLabel);
        panel.add(nombreF);
        panel.add(dniF);
        panel.add(tiper);
        panel.add(sLabel);
        panel.add(Salario);
        panel.add(fidu);
        panel.add(rent);
        tipoH.add(rent);
        tipoH.add(fidu);

```

```

        rent.addActionListener(new ActionListener()
        {
            public void
            actionPerformed(ActionEvent e) {
                viviendaCheck();
            }
        });

        fidu.addActionListener(new ActionListener()
        {
            public void
            actionPerformed(ActionEvent e) {
                viviendaCheck();
            }
        });
    }

    //Cambiar visibilidad de los componentes
    public void toggle(boolean cond) {
        tiper.setVisible(cond);
        fidu.setVisible(cond);
        rent.setVisible(cond);
        aLabel.setVisible(cond);
        nombreF.setVisible(cond);
        cLabel.setVisible(cond);
        dniF.setVisible(cond);
        Salario.setVisible(cond);
        sLabel.setVisible(cond);

        mLabel.setVisible(cond);
        montoValor.setVisible(cond);
        uLabel.setVisible(cond);
        ubicacion.setVisible(cond);
        tLabel.setVisible(cond);
        tama.setVisible(cond);

        if(cond) {
            subMenu.setText("Prestamo Vivienda");
            viviendaCheck();
        }
    }

    //Resetear componentes de esta clase
    public void resetV() {
        nombreF.setText("Nombre");
        dniF.setText("DNI");
        Salario.setText("(Numerico)");
        montoValor.setText("(Numerico)");
        ubicacion.setText("Ubicacion");
        tama.setText("(Numerico)");
    }

    //Desplegar campos segun Tipo de Hipoteca
    public void viviendaCheck() {
        if(fidu.isSelected()) {
            aLabel.setVisible(true);

```

```

        nombreF.setVisible(true);
        cLabel.setVisible(true);
        dniF.setVisible(true);
        Salario.setVisible(true);
        sLabel.setVisible(true);

        mLabel.setVisible(false);
        montoValor.setVisible(false);
        uLabel.setVisible(false);
        ubicacion.setVisible(false);
        tLabel.setVisible(false);
        tama.setVisible(false);
    }else if(rent.isSelected()) {
        aLabel.setVisible(false);
        nombreF.setVisible(false);
        cLabel.setVisible(false);
        dniF.setVisible(false);
        Salario.setVisible(false);
        sLabel.setVisible(false);

        mLabel.setVisible(true);
        montoValor.setVisible(true);
        uLabel.setVisible(true);
        ubicacion.setVisible(true);
        tLabel.setVisible(true);
        tama.setVisible(true);
    }
}

//Dar tipo de hipoteca
public int tipoHipoteca() {
    int total=0;
    if(fidu.isSelected()) {
        total=1;
    }else if(rent.isSelected()) {
        total=2;
    }
    return total;
}

//Dar campos del subMenu(Prestamos) Vivienda
//Tipo Fiduciario
public String darNombreF() {
    return nombreF.getText();
}

public String darDniF() {
    return dniF.getText();
}

public String darSalarioF() {
    return Salario.getText();
}

//Tipo Rental
public String darMontoV() {

```

```

        return montoValor.getText();
    }

    public String darUbicacion() {
        return ubicacion.getText();
    }

    public String darSize() {
        return tama.getText();
    }
}

}

}

//Guardar subMenu
class Guardar{
    JFileChooser dir=new JFileChooser();
    JButton abrir=new JButton("Buscar");
    JButton export=new JButton("Exportar Lista");
    JTextField textDir=new JTextField("Directorio (Archivo) a
exportar");
    FileNameExtensionFilter filtro=new
FileNameExtensionFilter("Archivo texto .txt","txt");
    public Guardar() {
        dir.setAcceptAllFileFilterUsed(false);
        dir.addChoosableFileFilter(filtro);
        textDir.setEditable(false);

        textDir.setBounds(50,50,310,40);
        abrir.setBounds(370,50,100,40);
        export.setBounds(50,100,200,60);

        toggleComponents(false);
        panel.add(dir);
        panel.add(abrir);
        panel.add(textDir);
        panel.add(export);

        export.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                if(dir.getSelectedFile()==null) {
                    mensajeDialogo("Seleccione un archivo
al cual exportar los registros",1);
                }else {

                    if(data.exportar(dir.getSelectedFile().toString())) {
                        mensajeDialogo("Se exportó
exitosamente",1);
                    }else {
                        mensajeDialogo("Fallo a la hora
de exportar",2);
                    }
                }
            }
        });
    }
}

```

```

        }
    }
});

abrir.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        dir.showDialog(panel, "Escojer un archivo");
        dir.setApproveButtonText("Seleccionar");
        if(dir.getSelectedFile() != null) {

textDir.setText(dir.getSelectedFile().toString());
        }
    }
});

}

//Cambiar visibilidad de los componentes
public void toggleComponents(boolean cond) {
    if(cond) {
        subTitle.setText("Guardar Lista");
    }
    dir.setVisible(cond);
    abrir.setVisible(cond);
    textDir.setVisible(cond);
    export.setVisible(cond);
}

//Resetear los campos perteneciente a la clase de Guardar
public void resetExportar() {
    textDir.setText("Directorio (Archivo) a exportar");
    dir.setSelectedFile(null);
}

}

//Buscar SubMenu
class BuscarDNI{
    JTextArea clients=new JTextArea();
    JTextField aBuscar=new JTextField("DNI del cliente a
encontrar");
    JButton gosSearch=new JButton("Buscar");
    JScrollPane scroller;

    public BuscarDNI() {
        clients.setEditable(false);
        aBuscar.setBounds(50,50,310,40);
        gosSearch.setBounds(370,50,100,40);
        scroller=new JScrollPane(clients);
        scroller.setBounds(50,100,400,310);
        toggleComponents(false);
        //panel.add(clients);
        panel.add(scroller);
        panel.add(aBuscar);
        panel.add(gosSearch);
        gosSearch.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                String text=aBuscar.getText();

```

```

        int encontrado=-5;
        for(int cont=0;cont<registro.length;cont++)
        {
            if(registro[cont]!=null) {

                if(registro[cont].getDNI().equalsIgnoreCase(text)) {
                    encontrado=cont;
                }
            }

            if(encontrado>=0) {

                clients.setText(registro[encontrado].brindarInformacion());
            }else {
                clients.setText("");
                mensajeDialogo("No se encontro ningun
cliente con el DNI "+text,1);
            }
        }
    });
}

//Cambiar visiblidad de los componentes de esta clase
public void toggleComponents(boolean cond) {
    if(cond) {
        subTitle.setText("Buscar Cliente");
    }
    //clients.setVisible(cond);
    aBuscar.setVisible(cond);
    gosSearch.setVisible(cond);
    scroller.setVisible(cond);
}

//Resetea los campos en de la clase BuscarNis
public void resetSearch() {
    clients.setText("");
    aBuscar.setText("DNI del cliente a encontrar");
}

}

//Ver todos subMenu
class VerTodos{
    JScrollPane scrollerT;
    JTextArea clientsT=new JTextArea();
    JButton nexto=new JButton("Siguiete");
    JButton ante=new JButton("Anterior");
    int currentPos=0;
    int maximoR=0;
    public VerTodos() {
        clientsT.setEditable(false);
        nexto.setBounds(120,420,100,40);
        ante.setBounds(15,420,100,40);
        scrollerT=new JScrollPane(clientsT);
        scrollerT.setBounds(50,60,400,350);
    }
}

```

```

toggleComponents(false);
panel.add(scrollerT);
panel.add(nexto);
panel.add(ante);

ante.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        verRegistro(false);
    }
});

nexto.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        verRegistro(true);
    }
});

}

//Se verifica si existe un anterior o siguiente registro y se
muestra
public void verRegistro(boolean type) {

    if(type) {
        if(currentPos==maximoR) {
            mensajeDialogo("No hay registros
siguientes",1);
        }else {
            currentPos+=1;
            clientsT.setText("N°
" +(currentPos+1)+"\n"+registro[currentPos].brindarInformacion());
        }
    }else {
        if(currentPos==0) {
            mensajeDialogo("No hay registros
anteriores",1);
        }else {
            currentPos-=1;
            clientsT.setText("N°
" +(currentPos+1)+"\n"+registro[currentPos].brindarInformacion());
        }
    }

}

//Contenido a mostrar cuando se despliega la opcion de ver
todos los registros
public void mostrarInicial() {
    maximoR=data.maximoRango()-1;
    clientsT.setText("N°
" +(currentPos+1)+"\n"+registro[currentPos].brindarInformacion());
}

//Cambiar visibilidad de los componentes de esta clase
public void toggleComponents(boolean cond) {
    if(cond) {

```



```

        subTitle.setText("Ver Clientes");
        mostrarInicial();
    }
    scrollerT.setVisible(cond);
    nexto.setVisible(cond);
    ante.setVisible(cond);
}

//Resetear componentes del subMenu ver todos
public void resetVer() {
    clientsT.setText("");
    currentPos=0;
}

}

//MenuPrincipal menu
class MenuPrincipal{
    JButton agregar=new JButton("Agregar Registro Cliente");
    JButton buscar=new JButton("Buscar cliente (DNI)");
    JButton verTodos=new JButton("Ver todos los Registros");
    JButton exportar=new JButton("Guardar Lista");
    JButton salir=new JButton("Salir");

    public MenuPrincipal() {
        agregar.setBounds(140,80,200,40);
        buscar.setBounds(140,150,200,40);
        verTodos.setBounds(140,220,200,40);
        exportar.setBounds(140,290,200,40);
        salir.setBounds(140,360,200,40);
        toggleComponents(false);

        panel.add(salir);
        panel.add(verTodos);
        panel.add(exportar);
        panel.add(buscar);
        panel.add(agregar);

        agregar.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {

                if(siguiente<(registro.length-1)) {
                    changer(1);
                }else {
                    mensajeDialogo("Se ha llegado a la
maxima capacidad del arreglo, proceda a reiniciar el sistema para obtener
más capacidad",1);
                }
            }
        });

        exportar.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                if(registro[0]==null) {

```

```

                                mensajeDialogo("No se han registrado
clientes para exportar",1);
                                }else {
                                    changer(4);
                                }
                            }
                        });

verTodos.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        if(registro[0]==null) {
            mensajeDialogo("No se han registrado
clientes",1);

        }else {
            changer(3);
        }
    }
});

salir.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        System.exit(0);
    }
});

buscar.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        if(registro[0]==null) {
            mensajeDialogo("No se han registrado
clientes",1);

        }else {
            changer(2);
        }
    }
});

}

//Cambiar visibilidad de los components del menuPrincipal
public void toggleComponents(boolean cond) {
    if(cond) {
        subTitle.setText("Menu Principal");
    }
    agregar.setVisible(cond);
    buscar.setVisible(cond);
    verTodos.setVisible(cond);
    exportar.setVisible(cond);
    salir.setVisible(cond);
}

}

//Manipulador de datos Recuperar,Almacenar datos & Exportar
informacion
class ManipularDatos{

```

```

        int max=1;

        public ManipularDatos() {
            if(!Recuperar()) {
                mensajeDialogo("Error a la hora de obtener el
archivo "+directorio+", los datos se guardaran en la memoria RAM ",2);
                registro=new Cliente[10];
            }
        }

        public Boolean Recuperar() {
            boolean bool=false;
            boolean condicion=true;
            try {
                File archivo=new File(directorio);
                //Se virifica si el archivo existe
                if(archivo.exists() && !archivo.isDirectory()) {

                    try {
                        int space=0;
                        Cliente tempArr[];
                        FileInputStream entrada=new
FileInputStream(archivo);
                        ObjectInputStream medio=new
ObjectInputStream(entrada);

                        tempArr=(Cliente[])medio.readObject();
                        for(int
cont=0;cont<tempArr.length;cont++) {
                            if(tempArr[cont]!=null) {
                                space=cont;
                            }
                        }

                        space+=1;

                        registro=new Cliente[space+10];

                        for(int
cont=0;cont<tempArr.length;cont++) {
                            if(tempArr[cont]!=null) {

                                registro[cont]=tempArr[cont];
                            }
                        }

                    }catch(Exception e) {
                        registro=new Cliente[10];
                    }
                }else {
                    //Si no existe el archivo
                    archivo.createNewFile();
                    mensajeDialogo("Se creo el archivo en el
directorio '"+directorio+"' donde se guardaran los datos",1);
                    registro=new Cliente[10];
                }
            }
        }
    }
}

```

```

        }

        bool=true;
    }catch(Exception e){
        bool=false;
    }

    return bool;
}

public boolean Almacenar(Cliente clienteP[]) {
    boolean bool=false;

    try {

        Cliente tempArrar[];
        FileOutputStream archivo;
        ObjectOutputStream medio;

        //True para append
        archivo=new FileOutputStream(directorio);
        medio=new ObjectOutputStream(archivo);

        tempArrar=new Cliente[(maximoRango()+1)];

        for(int cont=0;cont<(maximoRango()+1);cont++) {
            //if(clienteP[cont]!=null) {
                tempArrar[cont]=clienteP[cont];
            //}
        }

        medio.writeObject(tempArrar);
        medio.close();
        bool=true;
    }catch(IOException e){
        bool=false;
        e.printStackTrace();
    }
    return bool;
}

//Dar el maximo rango ocupado en el arreglo de registro
public int maximoRango() {
    int maxim=-5;

    for(int cont=0;cont<registro.length;cont++) {
        if(registro[cont]!=null) {
            maxim=cont;
        }
    }

    if(maxim==-5) {
        return 0;
    }
}

```

```

        }else {
            return maxim+1;
        }
    }

    //Crear un nuevo registro de cliente
    public boolean agregar(Crear crear) {
        boolean ret=false;
        try {
            //Creacion del objeto Cliente
            Cliente clienteTemporal=new Cliente();

            clienteTemporal.setAccionistaMayor(crear.giveClientM().darMayorA());

;

            clienteTemporal.setDNI(crear.giveClientM().darDniCliente());

            clienteTemporal.setMontoFinanciamiento(Double.parseDouble(crear.giveClientM().darMontoF()));

            clienteTemporal.setNombre(crear.giveClientM().darNombreCliente());

            clienteTemporal.setRazonSocial(crear.giveClientM().darRazon());

            clienteTemporal.setTipo(crear.giveClientM().clienteCheck());
            Operaciones
            operacionTemp=operacionTemporal(crear,clienteTemporal.getMontoFinanciamiento());

            operacionTemp.getEmpleado().setDNI(crear.giveEmpleado().darDniEmpleado());

            operacionTemp.getEmpleado().setNombre(crear.giveEmpleado().darNombreEmpleado());

            operacionTemp.getEmpleado().setSeccion(crear.giveEmpleado().checkEmpleado());

            clienteTemporal.setTipoOperacion(operacionTemp);
            registro[siguiente]=clienteTemporal;

            //Añadir objeto al registro Acamp9.dat
            if(!data.Almacenar(registro)) {
                mensajeDialogo("Ocurrio un error al guardarlo en el fichero externo",2);
            }

            siguiente++;
            ret=true;
        }catch(Exception e) {
            ret=false;
        }

        return ret;
    }

    //Se determina el tipo de operacion y se crea un objeto el cual se le agregara a la clase Cliente

```

```

        public Operaciones operacionTemporal(Crear crear, double
monto) {
            if(crear.darOperacion().equalsIgnoreCase("Prestamo
Personal")) {
                Personal temp=new Personal();

                temp.setOperacionNumero(Integer.parseInt(crear.givePrestamo().darNu
meroOp()));
                temp.setMontoCredificio(monto);

                temp.setMontoCredito(Double.parseDouble(crear.givePrestamo().darMon
toCredito()));
                return temp;
            }else
            if(crear.darOperacion().equalsIgnoreCase("Prestamo Vivienda")) {
                Vivienda temp=new
Vivienda(crear.givePrestamo().giveVienda().tipoHipoteca());

                temp.setOperacionNumero(Integer.parseInt(crear.givePrestamo().darNu
meroOp()));
                temp.setMontoCredificio(monto);

                temp.setMontoCredito(Double.parseDouble(crear.givePrestamo().darMon
toCredito()));

                if(crear.givePrestamo().giveVienda().tipoHipoteca()==1) {

                    temp.getFiduciaria().setDNI(crear.givePrestamo().giveVienda().darDn
iF());

                    temp.getFiduciaria().setNombre(crear.givePrestamo().giveVienda().da
rNombreF());

                    temp.getFiduciaria().setSalarioNeto(Double.parseDouble(crear.givePr
estamo().giveVienda().darSalarioF()));
                }else {

                    temp.getRegistral().setSizeMCuadrados(Double.parseDouble(crear.give
Prestamo().giveVienda().darSize()));

                    temp.getRegistral().setUbicacion(crear.givePrestamo().giveVienda().
darUbicacion());

                    temp.getRegistral().setValorRegistral(Double.parseDouble(crear.give
Prestamo().giveVienda().darMontoV()));
                }

                return temp;
            }else if(crear.darOperacion().equalsIgnoreCase("Targeta
de credito")) {
                Targeta temp=new Targeta();

                temp.setMontoGastado(Double.parseDouble(crear.giveTargeta().darMont
oG()));

                temp.setMontoLimite(Double.parseDouble(crear.giveTargeta().darMonto
L()));

```

```

        temp.setTargetNumero(crear.giveTargeta().darTargetNum());
        return temp;
    }else{
        PYMES temp=new PYMES();

        temp.setOperacionNumero(Integer.parseInt(crear.givePrestamo().darNumeroOp()));
        temp.setMontoCredificio(monto);

        temp.setMontoCredito(Double.parseDouble(crear.givePrestamo().darMontoCredito()));
        return temp;
    }
}

//Eportar el arreglo clientes al directorio dado USO de
usuario
public Boolean exportar(String dir) {
    boolean bool=false;
    try {
        FileWriter archivo=new FileWriter(dir);

        for(int cont=0;cont<registro.length;cont++) {
            if(registro[cont]!=null) {
                if(cont==0) {
                    archivo.write("\t\tRegistros de
clientes");

                    archivo.append("\n\nN°"+(cont+1)+" ":"+"\n"+registro[cont].brindarInformacion());
                }else {
                    archivo.append("\n\nN°"+(cont+1)+" ":"+"\n"+registro[cont].brindarInformacion());
                }
            }
        }

        archivo.close();
        bool=true;
    }catch(Exception e) {
        bool=false;
    }
    return bool;
}

//Metodo main para construir la interfaz grafica
public static void main(String[] args) {
    Interfaz interfaz=new Interfaz();
}
}

```

8. Problemas y Limitaciones

8.1 Problema: No se presentaron problemas a la hora de ejecutar el programa.

8.2 Limitaciones:

1. Si el usuario ingresa un dato tipo entero o decimal en uno de los datos en el que se le solicita un carácter, producirá un error.
2. Hay una limitante de 10 usuarios, si se desea ingresar más clientes se debe reiniciar el sistema y se contara con 10 espacios más para clientes además de los que ya se ingresó anteriormente.
3. No es una limitante, pero es importante aclarar que, a la hora de seleccionar un directorio para guardar los datos del cliente, tiene que existir un archivo “.txt” previamente.

9. Conclusión

El trabajo realizado determina que si bien algunos de los funcionamientos del código del programa, tales como la descripción del funcionamiento del programa y la descripción de los datos para un mayor entendimiento. A la vez, se comprende el problema que presentaba la entidad financiera ACAMP S.A, en el departamento de despacho del cliente, el cual era la ineficacia a la hora de los funcionarios poder solicitar la información correspondiente de cada usuario. Se planteo una solución de índole entendible y muy descriptivo con la finalidad de comprender el programa para cualquier funcionario de la identidad. Asimismo, se trató de realizar comentarios claros para que cualquier programador entendiera las funciones que realizaba los algoritmos impuestos en el código, si bien se trató de comentar exclusivamente aquellos atributos, métodos o ciclos que fueran confusos a simple vista.

Al mismo tiempo, tiene una interfaz sencilla y legible para que el funcionario comprende. El programa cuenta con una serie de instrucciones, la cual cumple la necesidad de poder evacuar dudas sencillas hacia el funcionario que esté utilizando el programa. En estas instrucciones se especifica con claridad que elementos deberá ingresar en cada opción.

Con respecto al trabajo, se pudo abarcar cada uno de los puntos de los Objetivos satisfactoriamente, creando un programa ordenado y de forma muy descriptiva para cualquier persona que lo manipule.