בס״ד

19.7.2020

Names: Ori Ashkenazi.

ClubApp:

NightClubMgmtApp class:

```
/**
 * The class NightClubMgmtApp allows the user to work with the infrastructure.
 * The purpose of the class is to create a user experience in the infrastructure of abstract
class ClubAbstractEntity.
 * @author Nir Sananes & Ori Ashkenazi
 */

        /*Imports*/
        import java.util.*;
        import java.awt.*;
        import java.awt.event.*;
        import javax.swing.*;
        import java.io.*;

        /*NightClubMgmtApp class*/
        public class NightClubMgmtApp
        {
        //Night-Club Regular Customers Repository
        private ArrayList<ClubAbstractEntity> clubbers;
        private JFrame frame;

        /*Default constructor*/
         /**
         * NightClubMgmtApp parameterless constructor, initializes a new
NightClubMgmtApp.
         * Creates a user experience in the infrastructure of the abstract class
ClubAbstractEntity.
         * Classes use: {@link java.awt.event.EventListener}, {@link
javax.swing.JFrame}, {@link javax.swing.JLabel},
   *{@link javax.swing.JButton}, {@link javax.swing.JPanel}, {@link
javax.swing.ImageIcon},{@link javax.swing.Icon},{@link java.util.ArrayList<E>}
   *{@link java.io.showMessageDialog},{@link java.io.showInputDialog},{@link
java.awt.event.WindowAdapter}.
         */
        public NightClubMgmtApp()
        {
        //main JFrame
```

```
        frame=new JFrame("Ori & nir application");
        frame.setLayout(new BorderLayout());
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setPreferredSize(new Dimension(900, 800));
frame.pack();
frame.setLocationRelativeTo(null);
frame.setVisible(true);
//Initializing clubbers
clubbers=new ArrayList<>();
//read from file when app begins
        loadClubbersDBFromFile();
//create JPanels
JPanel[] Panels = new JPanel[4];
for(int i=0; i<4; i++)
{
        Panels[i]=new JPanel();
}

//Picture add to panel
Icon pic = new ImageIcon(getClass().getResource("picture.jpg"));
JLabel imgLabel = new JLabel(new ImageIcon("picture.jpg"));
Panels[0].add(imgLabel);

//Search button
JButton b1=new JButton("Search");
Panels[3].add(b1);

        //Create clubber button
        JButton b2=new JButton("Create clubber");
Panels[2].add(b2);

        //Welcome label
        JLabel label = new JLabel();
        label.setText("Welcome to Ori & Nir clubbers application! Enjoy :)");
        Panels[1].add(label);

        //Panels colors
        Panels[2].setBackground(Color.BLACK);
        Panels[1].setBackground(Color.CYAN);
Panels[3].setBackground(Color.WHITE);
Panels[0].setBackground(Color.DARK_GRAY);

        //Panels add and set
        frame.add(Panels[1],BorderLayout.NORTH);
        frame.add(Panels[0],BorderLayout.CENTER);
        frame.add(Panels[3],BorderLayout.EAST);
frame.add(Panels[2],BorderLayout.WEST);
```

```
for(int i=0; i<4; i++)
{
	Panels[i].revalidate();
	Panels[i].validate();
		Panels[i].repaint();
}

	//Search button action listener
		b1.addActionListener(new ActionListener() {

			@Override
			public void actionPerformed(ActionEvent e) {
				manipulateDB();

			}
	    });


	//Create button action listener
		b2.addActionListener(new ActionListener() {

			@Override
			public void actionPerformed(ActionEvent e) {
				//switch case for the user choose
				String
clubber_choose=JOptionPane.showInputDialog(null,"What kind of clubber are you?\nFor
Person - type 1\nFor Soldier - type 2\nFor Student - type 3\nBack to main menu - type 0");
				if(clubber_choose != null)
				{
				switch(clubber_choose) {
				case "0":
					break;
				case "1":
					clubbers.add(new Person());
					break;
				case "2":
					clubbers.add(new Soldier());
					break;
				case "3":
					clubbers.add(new Student());
					break;
				default:

	JOptionPane.showMessageDialog(null,"Error, you chose
wrong","Error!",JOptionPane.ERROR_MESSAGE);
					break;
```

```
                                        }

                                        }//end of switch case
                                    }//end of action performed
                    });//end of action listener



  //Exit the user window
     frame.addWindowListener(new WindowAdapter() {
   @Override
   public void windowClosing(java.awt.event.WindowEvent windowEvent) {
       if (JOptionPane.showConfirmDialog(frame,"Are you sure you want to close this
window?", "Close Window?",
           JOptionPane.YES_NO_OPTION,JOptionPane.QUESTION_MESSAGE) ==
JOptionPane.YES_OPTION){
                           writeClubbersDBtoFile();
           System.exit(0);
       }
   }
   });
            }//end of parameterless (default) constructor


            /*manipulateDB method*/
             /**
             * manipulateDB method asks the user for a key and check if the key exists in the
club.
             * Classes use: {@link java.io.showMessageDialog},{@link
java.io.showInputDialog}.
             */
            private void manipulateDB()
            {
            boolean found = false;
            while(true)
            {
            String input=JOptionPane.showInputDialog(null,"Please Enter The Clubber's
Key");
            if(input==null)
            {
             JOptionPane.showMessageDialog(null, "Come Back find your Friend soon
:)..\n","Wait!", JOptionPane.INFORMATION_MESSAGE);
             break;
            }
            for(ClubAbstractEntity clubber : clubbers)
            if(clubber.match(input))
            {
```

```
            found = true;
            clubber.setVisible(true);
            break;
            }
            if(!found)
   JOptionPane.showMessageDialog(null, "Clubber with this key does not
exist\n","Wait!", JOptionPane.INFORMATION_MESSAGE);
            else found = !found;
            }
            }//end of method manipulateDB

            /*loadClubbersDBFromFile method*/
            /**
             * loadClubbersDBFromFile method reads from a binary file and fills in the list
of existing people in the club.
             * Classes use: {@link java.io.FileInputStream}, {@link
java.io.ObjectInputStream}.
             * Exception: {@link java.io.EOFException}, {@link
java.io.FileNotFoundException}, {@link java.io.ClassNotFoundException}, {@link
java.io.IOException}.
             */
            private void loadClubbersDBFromFile()
            {
   try{
    FileInputStream fis = new FileInputStream("BKCustomers.dat");
    ObjectInputStream ois = new ObjectInputStream(fis);
    clubbers = (ArrayList<ClubAbstractEntity>) ois.readObject();
    fis.close();
    ois.close();
   }
   catch(EOFException e)
   {
            //EOF
   }
   catch(FileNotFoundException e)
   {
            //Not Exisst
   }
    catch(IOException e)
            {
                     e.printStackTrace();
            }
             catch(ClassNotFoundException e)
            {
                     e.printStackTrace();
            }
            }
```

```
        /*writeClubbersDBtoFile method*/
        /**
         * writeClubbersDBtoFile method writes to a binary file and fills it from the list
of existing people in the club.
         * Classes use: {@link java.io.ObjectOutputStream}, {@link
java.io.FileOutputStream},{@link java.io.showMessageDialog},{@link
javax.swing.JOptionPane}.
         * Exception: {@link java.io.EOFException}, {@link
java.io.FileNotFoundException}, {@link java.io.IOException}.
         */
        private void writeClubbersDBtoFile()
        {
                try
                {
                FileOutputStream fos = new FileOutputStream("BKCustomers.dat");
                ObjectOutputStream oos = new ObjectOutputStream(fos);
                oos.writeObject(clubbers);
                fos.flush();
                oos.close();
    JOptionPane.showMessageDialog(null, "Successfully wrote to the
file.\n","Succsess!", JOptionPane.INFORMATION_MESSAGE);
    }
    catch(EOFException e)
    {
        //EOF
    }
    catch(FileNotFoundException e)
    {
        //file doesnt exist - not happend.
    }
     catch(IOException e)
    {
        e.printStackTrace();
    }
        }

        /*writeClubbersDBtoFile method*/
        /**
         * Main- call NightClubMgmtApp constructor.
         */
        public static void main(String[] args)
        {
        NightClubMgmtApp appliction = new NightClubMgmtApp();
        }
        }//end of class NightClubMgmtApp
```

ClubAbstractEntity class:

```
/**
 * This is abstract class ClubAbstractEntity extends from JFrame
 * Builds the frame and the center panel (with buttons and labels) that will be ready for construction by each of the users: Person, Soldier, Student.
 * @author Nir Sananes & Ori Ashkenazi
 */
        /*Imports*/
        import java.awt.*;
        import java.awt.event.*;
        import javax.swing.*;
        import java.io.Serializable;

public  abstract class ClubAbstractEntity extends JFrame
{
        /*Instance Variables*/
        private JPanel centerPanel;
        private JButton okButton;
        private JButton cancelButton;
        private ButtonHandler handler;

        /*Parameterless Constructor*/

         /**
         * ClubAbstractEntity parameterless constructor, initializes a new JFrame , JButton , JPanel , ButtonHandler.
         * Classes use: {@link javax.swing.JLabel},{@link javax.swing.JFrame}
    *{@link javax.swing.JButton}, {@link javax.swing.JPanel},{@link javax.swing.JTextField}
         */
        public ClubAbstractEntity()
        {
                //Create main JFrame
                super("Welcome to the club");

                //Main JFrame settings
                BorderLayout layout1=new BorderLayout();
      setLayout(layout1);
                getContentPane().setBackground(Color.gray);
                setLocationRelativeTo(null);
                setResizable(false);
                setDefaultCloseOperation(0);
                setVisible(true);

                //Create panels, labels, buttons and handlers
                centerPanel= new JPanel();
                JPanel[] WindowPanels= new JPanel[2];
```

```
                    for(int i=0;i<2;i++)
                    WindowPanels[i]=new JPanel();
                    okButton=new JButton("ok");
                    cancelButton=new JButton("cancel");
                    handler=new ButtonHandler();

                    //Gui settings
                    FlowLayout layout2=new FlowLayout(FlowLayout.RIGHT,5,10);
                    centerPanel.setLayout(layout2);
                    WindowPanels[0].setPreferredSize(new Dimension(40, 40));
                    WindowPanels[1].setPreferredSize(new Dimension(50, 50));

        //Buttons ActionListeners
        okButton.addActionListener(handler);
        cancelButton.addActionListener(handler);

        //Gui adding
        WindowPanels[0].add(okButton);
            WindowPanels[0].add(cancelButton);
            add(WindowPanels[0], BorderLayout.SOUTH);
            add(centerPanel, BorderLayout.CENTER);
            add(WindowPanels[1], BorderLayout.WEST);

            }//end of ClubAbAstractEntity parameterless (default) constructor

            /*Methods*/
            /**
             * addToCenter method adds GUI Components to the centerPanel.
             * @param guiComponent Id number of person.
             */
            protected void addToCenter(Component guiComponent)
            {
                    centerPanel.add(guiComponent);
            }//end of addToCenter

            /**
             * disable_cancel_button method disables the ability and fuctional of the
cancelButton click.
             */
            protected void disable_cancel_button()
            {
                    cancelButton.setEnabled(false);
            }//end of disable_cancel_button

            /*Abstract methods*/
                    /**
```

```
	 * disable_cancel_button method disables the ability and fuctional of the
cancelButton click.
	 * Match abstract method - search if person exist in the club by key number.
	 * @param key ID number for search.
	 * @return boolean true or false (the match result).
	 */
	public abstract boolean match(String key);
	/**
	 * validateData abstract method , this function checks whether details meet the
standards of club registration.
	 * @return boolean true or false.
	 * Classes use: {@link javax.swing.JTextField},{@link
javax.swing.JLabel},{@link java.lang.String}.
	 */
	protected abstract boolean validateData();
	/**
	 * commit abstract method ,A function that puts details in the "belly" from text-
fields.
	 * Classes use: {@link javax.swing.JTextField},{@link java.lang.String}.
	 */
	protected abstract void commit();
	/**
	 * rollBack abstract method ,A function that returns details from the "belly" to
the text-fields.
	 * Classes use: {@link javax.swing.JTextField}.
	 */
	protected abstract void rollBack();

	/*Inner Class*/
	 /**
	 * ButtonHandler Private inner class for event handling-implements from
ActionListener - Activates the functionality of the Buttons.
	 * Classes use:  {@link java.awt.event.EventListener}, {@link
javax.swing.JButton}.
	 */
	private class ButtonHandler implements ActionListener,Serializable
	{
		@Override
		public void actionPerformed(ActionEvent event)
		{
			if(event.getActionCommand()=="ok")
			{

	if(validateData()==true){commit(); cancelButton.setEnabled(true); }
			}

			else if(event.getActionCommand()=="cancel")rollBack();
```

```
                }
        }//end of ButtonHandler inner class

}//end of ClubAbstractEntity class
```

Person class:

```
/**
 * Person Class extends the state and behaviour of the abstract Class - ClubAbstractEntity
& JFrame
 * @author Nir Sananes & Ori Ashkenazi
 */

        /*Imports*/
        import java.awt.*;
        import java.awt.event.*;
        import javax.swing.*;
        import java.util.*;
        import java.io.Serializable;

public class Person extends ClubAbstractEntity implements Serializable
{
        /*Instance Variables*/
        private String Id;
        private String Name;
        private String Surname;
        private String Tel;
   private JTextField[] fields;
   private JLabel[] redot;

        /*Default Constructor*/
         /**
         * Person parameterless constructor, initializes a new Person.
         * Classes use: {@link javax.swing.JLabel},{@link javax.swing.JFrame}
    *{@link javax.swing.JButton}, {@link javax.swing.JPanel},{@link
javax.swing.JTextField}
         */
        public Person()
        {
                fields=new JTextField[4];
        redot=new JLabel[4];
        String[] Labels_names= {"ID","Name","Surname","Tel"};
        for(int i=0; i<4; i++)//redots and fields loop
        {
                JLabel Label=new JLabel(Labels_names[i]);
                redot[i]=new JLabel("*");
                redot[i].setForeground(Color.red);
```

```
                redot[i].setVisible(false);
                Label.setPreferredSize(new Dimension(60, 20));
                fields[i]=new JTextField(25);
                super.addToCenter(Label);
                super.addToCenter(fields[i]);
                   super.addToCenter(redot[i]);
            }//end of loop
            setTitle("Person Clubber's Data");
            setSize(450,220);
            super.disable_cancel_button();
            this.Id=new String();
                this.Name=new String();
                this.Surname=new String();
                this.Tel=new String();
            }//end of parameterless (default) constructor

            /*Parameters Constructor*/
            /**
             * Person arguments constructor, initializes a new Person with arguments (Id,
Name, Surename, Tel).
             * @param Id ID number of the person.
             * @param Name Name of the person.
             * @param Surname Surname of the person.
             * @param Tel Telephone number of the person.
             * Classes use: {@link javax.swing.JLabel},{@link javax.swing.JFrame}
    *{@link javax.swing.JButton}, {@link javax.swing.JPanel},{@link
javax.swing.JTextField}
             */
   public Person(String Id,String Name,String Surname,String Tel)
   {
            fields=new JTextField[4];
            redot=new JLabel[4];
            String[] Labels_names= {"ID","Name","Surname","Tel"};
            for(int i=0; i<4; i++)//redots and fields loop
            {
                JLabel Label=new JLabel(Labels_names[i]);
                redot[i]=new JLabel("*");
                redot[i].setForeground(Color.red);
                redot[i].setVisible(false);
                Label.setPreferredSize(new Dimension(60, 20));
                fields[i]=new JTextField(25);
                super.addToCenter(Label);
                super.addToCenter(fields[i]);
                   super.addToCenter(redot[i]);
            }//end of loop
            setTitle("Person Clubber's Data");
            setSize(450,220);
```

```
		this.Id=new String(Id);
		this.Name=new String(Name);
		this.Surname=new String(Surname);
		this.Tel=new String(Tel);
	}//end of parameters constructor

		/*Abstract methods*/
		/**
		 * Match abstract method from abstract Class ClubAbstractEntity - search if person exist in the club by key number.
		 * @param key ID number for search.
		 * @return boolean true or false (the match result).
		 * Classes use: {@link javax.swing.JTextField},{@link java.lang.String}.
		 */

		public boolean match(String key)
		{
			return Id.equals(key);
		}//end of match method

		/**
		 * validateData abstract method from abstract Class ClubAbstractEntity, this function checks whether the person's details meet the standards of club registration.
		 * @return boolean true or false.
		 * Classes use: {@link javax.swing.JTextField},{@link javax.swing.JLabel},{@link java.lang.String}.
		 */
		protected boolean validateData()
		{
			//validateData pre-settings
			boolean flag=true;
			boolean check_1=true;
			boolean check_2=true;
			int a=0;
			int counter1=0;
			int counter2=0;
			int cap_let=0;
			int characters=0;
			int first_nums=0;
			int sec_nums=0;
			int third_nums=0;
			char check=0;


			//ID field RE

		if(fields[0].getText().trim().isEmpty()){redot[0].setVisible(true); flag=false; }
```

```
                    else{
                    for(int i=2; i<(fields[0].getText()).length()-2;i++)
                    {

          if((fields[0].getText()).charAt(i)>='0'&&(fields[0].getText()).charAt(i)<='9')counter
1++;
                    }
                    if(!(((fields[0].getText()).charAt(0)>='0'&&
(fields[0].getText()).charAt(0)<='9')&&((fields[0].getText()).charAt(1)=='-
')&&((fields[0].getText()).charAt(9)=='/')&&((fields[0].getText()).charAt(10)>='1')&&((fields[0
].getText()).charAt(10)<='9')&&(counter1==7)))
                              {redot[0].setVisible(true); flag=false; }
                    else {redot[0].setVisible(false); }
                    }//end of ID check

                    //Name field RE

          if(fields[1].getText().trim().isEmpty()){redot[1].setVisible(true); flag=false; }
                    else{
                    for(int i=1; i<(fields[1].getText()).length(); i++)
                    {

          if((fields[1].getText()).charAt(i)>='a'&&(fields[1].getText()).charAt(i)<='z')counter
2++;
                    }

          if(!(((fields[1].getText()).charAt(0)>='A'&&(fields[1].getText()).charAt(0)<='Z')&&(
counter2!=0)))
                              {redot[1].setVisible(true); flag=false; }
                    else {redot[1].setVisible(false); }
                    }//end of Name check

          //Surname field RE

          if(fields[2].getText().trim().isEmpty()){redot[2].setVisible(true); flag=false; check_
1=false; }//if jtextfield empty

          else{check=(fields[2].getText()).charAt(0); a=(fields[2].getText()).length(); }
                    if(a==1){redot[2].setVisible(true); flag=false; check_1=false; }//if
jtextfield contain 1 letter/character
                    else
if(!(check>='A'&&check<='Z')){redot[2].setVisible(true); flag=false; check_1=false; }//if
first letter not capital
                    if(check_1==true){
                    for(int i=0; i<(fields[2].getText()).length(); i++)
                    {
                              check=(fields[2].getText()).charAt(i);
```

```
                    if(!((check>='A'&&check<='Z')||check==(char)39||check=='-
'||(check>='a'&&check<='z')))check_2=false;
                    if(check==(char)39||check=='-')characters++;
                }//check if all the letters or characters are as needed and counting
capital letters+characters
                if(check_2==false){redot[2].setVisible(true); flag=false; }
                else if(characters>2){redot[2].setVisible(true); flag=false; }
                else {redot[2].setVisible(false); }
                }//end of Surename check*/

                //Tel field RE

        if(fields[3].getText().trim().isEmpty()){redot[3].setVisible(true); flag=false; }
                else{
                int i=0;
                int j=0;
                for(; (fields[3].getText()).charAt(i)!=')'; i++)
                {

        if((fields[3].getText()).charAt(i)>='0'&&(fields[3].getText()).charAt(i)<='9')first_nu
ms++;
                }

                for(i=first_nums+3; (fields[3].getText()).charAt(i)!='-'; i++)
                {

        if((fields[3].getText()).charAt(i)>='0'&&(fields[3].getText()).charAt(i)<='9')sec_nu
ms++;
                }

                for(j=i+1; j<(fields[3].getText()).length(); j++)
                {

        if((fields[3].getText()).charAt(j)>='0'&&(fields[3].getText()).charAt(j)<='9')third_n
ums++;
                }

        if(!((fields[3].getText()).charAt(0)=='+')&&((fields[3].getText()).charAt(1)=='(')&&((
fields[3].getText()).charAt(2)!='0')&&((fields[3].getText()).charAt(first_nums+2)==')')&&((fie
lds[3].getText()).charAt(first_nums+3)>='1'&&(fields[3].getText()).charAt(first_nums+3)<='
9')&&(first_nums<=3)&&(sec_nums<=3)&&((fields[3].getText()).charAt(i+1)!='0')&&(third_
nums==7))
                {redot[3].setVisible(true); flag=false; }
                else {redot[3].setVisible(false); }
                }//end of Tel check

                return flag; //return flag!
```

```
        }//end of validateData method

        /**
         * commit abstract method from abstract Class ClubAbstractEntity ,A function
that puts the person's details in the "belly" from text-fields.
         * Classes use: {@link javax.swing.JTextField},{@link java.lang.String}.
         */
        protected void commit()
        {
                Id=new String(fields[0].getText());
                Name=new String(fields[1].getText());
                Surname=new String(fields[2].getText());
                Tel=new String(fields[3].getText());
        };//end of commit()

        /**
         * rollBack abstract method from abstract Class ClubAbstractEntity ,A function
that returns the person's details from the "belly" to the text-fields.
         * Classes use: {@link javax.swing.JTextField}.
         */
        protected void rollBack()
        {
                fields[0].setText(Id);
                fields[1].setText(Name);
                fields[2].setText(Surname);
                fields[3].setText(Tel);
        }//end of rollBack()
}//end of Person class
```

Soldier class:

```
/**
 * This Class extends the state and behaviour of Person & abstract Class
ClubAbstractEntity & JFrame
 * @author Nir Sananes & Ori Ashkenazi
 */
            /*Imports*/
            import java.awt.*;
            import java.awt.event.*;
            import javax.swing.*;
            import java.util.*;
            import java.io.Serializable;

public class Soldier extends Person implements Serializable
{
            /*Instance Variables*/
             private String personalNum;
             private JTextField field;
             private JLabel redot1;

             /*Default Constructor*/
             /**
              * Soldier constructor, initializes a new Soldier.
              * Classes use: {@link javax.swing.JLabel},{@link javax.swing.JFrame}
    *{@link javax.swing.JButton}, {@link javax.swing.JPanel},{@link
javax.swing.JTextField}
             */
            public Soldier()
            {
                        super();
            redot1=new JLabel("*");
            redot1.setForeground(Color.red);
            redot1.setVisible(false);
            setTitle("Soldier Clubber's Data");
            setSize(450,250);
            JLabel js=new JLabel("Personal No.");
            this.personalNum=new String();
            field=new JTextField(25);
            js.setPreferredSize(new Dimension(80, 20));
            super.addToCenter(js);
            //addToCenter method - add soldier centerpanel to the frame
            super.addToCenter(this.field);
            super.addToCenter(redot1);
            super.disable_cancel_button();
            }//end of parameterless (default) constructor

            /*Parameters Constructor*/
```

```
	/**
	 * Soldier arguments constructor, initializes a new Soldier with arguments (Id,
Name, Surename, Tel, new_personalNum).
	 * @param Id ID number of the soldier.
	 * @param Name Name of the soldier.
	 * @param Surname Surname of the soldier.
	 * @param Tel Telephone number of the soldier.
	 * @param new_personalNum personal number of the soldier.
	 * Classes use: {@link javax.swing.JLabel},{@link javax.swing.JFrame}
   *{@link javax.swing.JButton}, {@link javax.swing.JPanel},{@link
javax.swing.JTextField}
	 */
  public Soldier(String Id,String Name,String Surname,String Tel,String
new_personalNum)
  {
		super(Id,Name,Surname,Tel);
		redot1=new JLabel("*");
		redot1.setForeground(Color.red);
		redot1.setVisible(false);
		setTitle("Soldier Clubber's Data");
		setSize(450,250);
		JLabel js=new JLabel("Personal No.");
		this.personalNum=new String(new_personalNum);
		field=new JTextField(25);
		js.setPreferredSize(new Dimension(80, 20));
		super.addToCenter(js);
		//addToCenter method - add soldier centerpanel to the frame
		super.addToCenter(this.field);
		super.addToCenter(redot1);
		}//end of parameters constructor

		/*Methods*/
		/**
		 * Match abstract method from abstract Class ClubAbstractEntity, searches if
Soldier exist in the club (by key number).
		 * @param key Id number for search.
		 * @return boolean true or false (by match result).
		 * Classes use: {@link javax.swing.JTextField},{@link java.lang.String}.
		 */
		public boolean match(String key)
		{
			if(super.match(key) == true && personalNum.equals(key))
			return true;
			return false;
		};//end of match()

		/**
```

```
	 * validateData abstract method from abstract Class ClubAbstractEntity, the function checks whether the soldier's details meet the standards of club registration.
	 * @return boolean true or false.
	 * Classes use: {@link javax.swing.JTextField},{@link javax.swing.JLabel},{@link java.lang.String}.
	 */
	protected boolean validateData()
	{
		boolean valid=true;
		if(field.getText().trim().isEmpty()){redot1.setVisible(true); valid=false; }

	if(!((field.getText()).charAt(0)=='R'||(field.getText()).charAt(0)=='O'||(field.getText()).charAt(0)=='C')){redot1.setVisible(true); valid=false; }

	if(!((field.getText()).charAt(1)=='/')){redot1.setVisible(true); valid=false; }
		if(valid){
		int counter=0;
		for(int i=2; i<(field.getText()).length(); i++)
		{
		 if((field.getText()).charAt(i)>='0' && (field.getText()).charAt(i)<='9')counter++;
		}
		if(counter!=7){redot1.setVisible(true); valid=false; }
		}
		if(valid){redot1.setVisible(false); }
		return (super.validateData()&&valid);
	}//end of validateData()

	 /**
	 * Commit abstract method from abstract Class ClubAbstractEntity, the function puts the soldier's details in the "belly" from text-fields.
	 * Classes use: {@link javax.swing.JTextField},{@link java.lang.String}.
	 */
	protected void commit()
	{
		super.commit();
		this.personalNum=new String(field.getText());
	};//end of commit()

	/**
	 * rollBack abstract method from abstract Class ClubAbstractEntity, the function returns the soldier's details from the "belly" to the text-fields.
	 * Classes use: {@link javax.swing.JTextField}.
	 */
	protected void rollBack()
	{
		super.rollBack();
```

```
                    field.setText(personalNum);
            }; //end of rollBack()
}
```

Student class:

```
**/
 * This Class extends the state and behaviour of Person & abstract Class
ClubAbstractEntity & JFrame
@ * author Nir Sananes & Ori Ashkenazi
/*
        */Imports/*
        import java.awt;*.
        import java.awt.event;*.
        import javax.swing;*.
        import java.util;*.
        import java.io.Serializable;


public class Student extends Person implements Serializable
}
        */Instance Variables/*
        private String Studentid;
   private JTextField field;
   private JLabel redot2;


        */Default Constructor/*
        **/
         * Student parameterless (default) constructor, initializes a new Student.
         * Classes use: {@link javax.swing.JLabel},{@link javax.swing.JFrame}
*    {@link javax.swing.JButton}, {@link javax.swing.JPanel},{@link
javax.swing.JTextField}
        /*
        public Student()
        }
                super;()
                redot2=new JLabel;("*")
                redot2.setForeground(Color.red);
```

```
                redot2.setVisible(false);
        setTitle("Student Clubber's Data");
        setSize;(450,250)
        JLabel js=new JLabel("Student ID");
        this.Studentid=new String;()
        field=new JTextField;(25)
        js.setPreferredSize(new Dimension(60, 20));
        super.addToCenter(js);
        //addToCenter method - add student centerpanel to the frame
      super.addToCenter(this.field);
      super.addToCenter(redot2);
      super.disable_cancel_button;()
        //{end of paramterless (default) constructor


        */Parameters Constructor/*
        **/
         * Student argument constructor, initializes a new Student with arguments (Id,
Name, Surename, Tel, new_Studentid) .
        @ * param Id ID number of the student.
        @ * param Name Name of the student.
        @ * param Surname Surname of the student.
        @ * param Tel Telephone number of the student.
        @ * param new_Studentid StudentID number.
         * Classes use: {@link javax.swing.JLabel},{@link javax.swing.JFrame}
*    {@link javax.swing.JButton}, {@link javax.swing.JPanel},{@link
javax.swing.JTextField}
        /*
  public Student(String Id,String Name,String Surname,String Tel,String new_Studentid)
}
        super(Id,Name,Surname,Tel);
                redot2=new JLabel;("*")
                redot2.setForeground(Color.red);
```

```
            redot2.setVisible(false);
        setTitle("Student Clubber's Data");
        setSize;(450,250)
        JLabel js=new JLabel("Student ID");
        this.Studentid=new String(new_Studentid);
        field=new JTextField;(25)
        js.setPreferredSize(new Dimension(60, 20));
        super.addToCenter(js);
        //addToCenter method - add student centerpanel to the frame
      super.addToCenter(this.field);
      super.addToCenter(redot2);
//{    end of paramters constructor


        */Methods/*
        **/
         * Match abstract method from abstract Class ClubAbstractEntity, searches if
Student exist in the club (by key numer).
        @ * param key Id number for search.
        @ * return boolean true or false.
         *  Classes use: {@link javax.swing.JTextField},{@link java.lang.String}.
        /*
        public boolean match(String key)
        }
                boolean flag=true;
                int counter3=0;
                int j=0;
                for(int i=3; i<Studentid.length(); )
                }
                        if(Studentid.charAt(i)==key.charAt(j))counter3; ++
                        i++; j; ++
                {
                if(counter3!=5)flag=false;
```

```
        if(super.match(key)||flag)return true;
        return flag;
    //;{end of match()

    **/
```

* validateData abstract method from abstract Class ClubAbstractEntity, the function checks whether the student's details meet the standards of club registration.

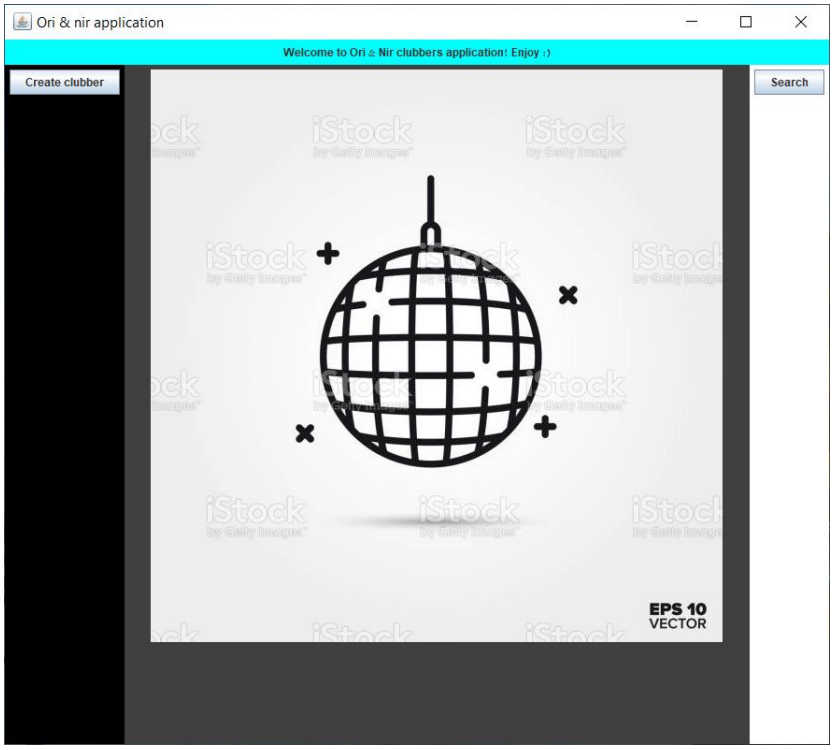
@ * return boolean true or false.

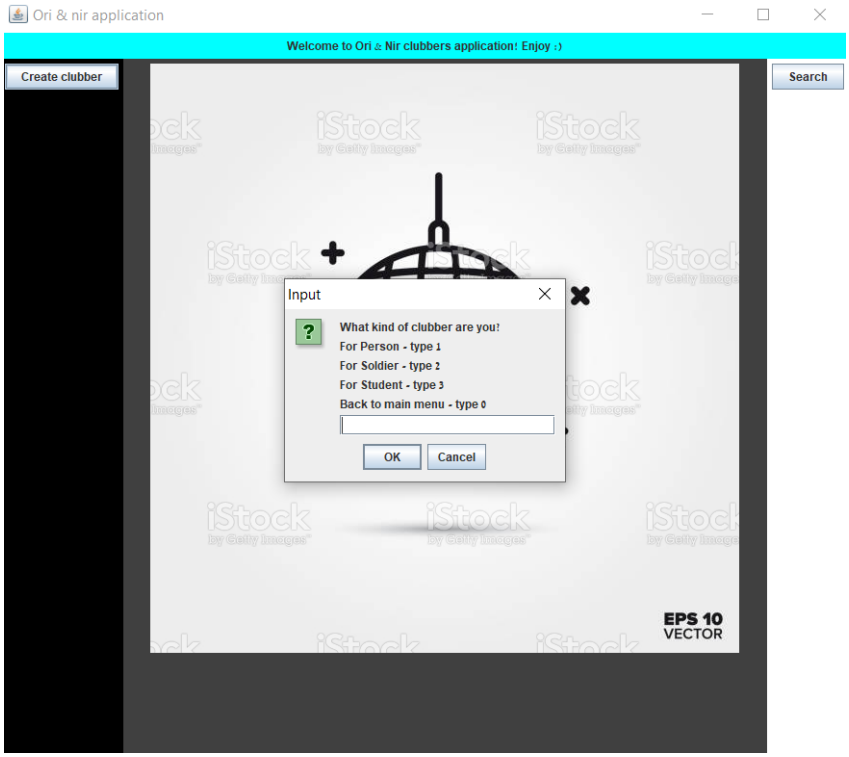
* Classes use: {@link javax.swing.JTextField},{@link javax.swing.JLabel},{@link java.lang.String}.

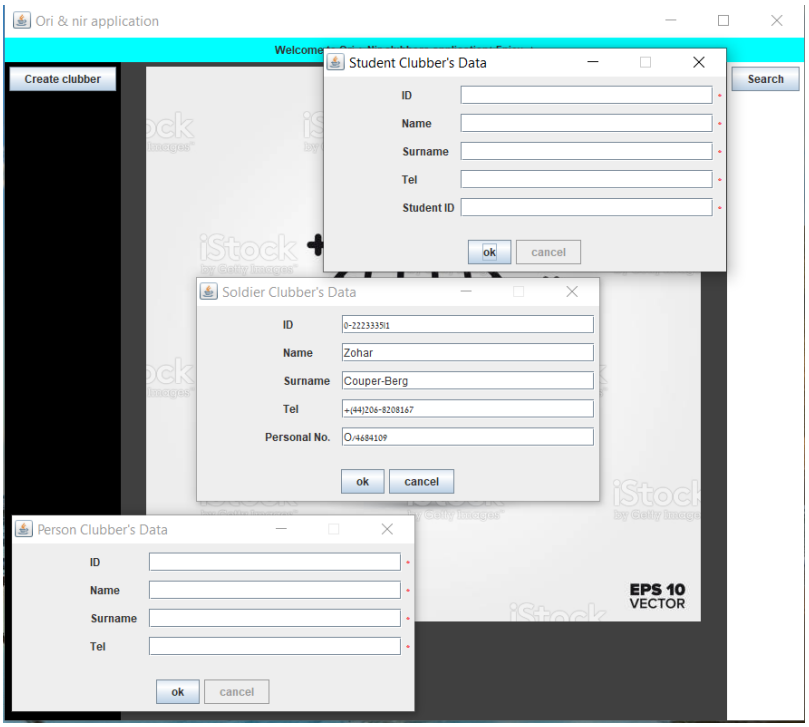
```
    /*
    protected boolean validateData()
    }
        boolean valid=true;
        if(field.getText().trim().isEmpty()){redot2.setVisible(true); valid=false; }
        int a=field.getText().length; ()
        if(a!=8){redot2.setVisible(true); valid=false; }
        if(valid)}
        int counter1=0;
        int counter2=0;
        char check=0;
        for(int i=0; i<3; i++)
        }
            check=(field.getText()).charAt(i);
            if(check>='A'&&check<='Z')counter1; ++
        {
        if(counter1!=3){redot2.setVisible(true); valid=false; }
        if((field.getText()).charAt(3)=='0'){redot2.setVisible(true); valid=false; }
        for(int i=4; i<8; i++)
        }
            check=(field.getText()).charAt(i);
            if(check>='0'&&check<='9')counter2; ++
```

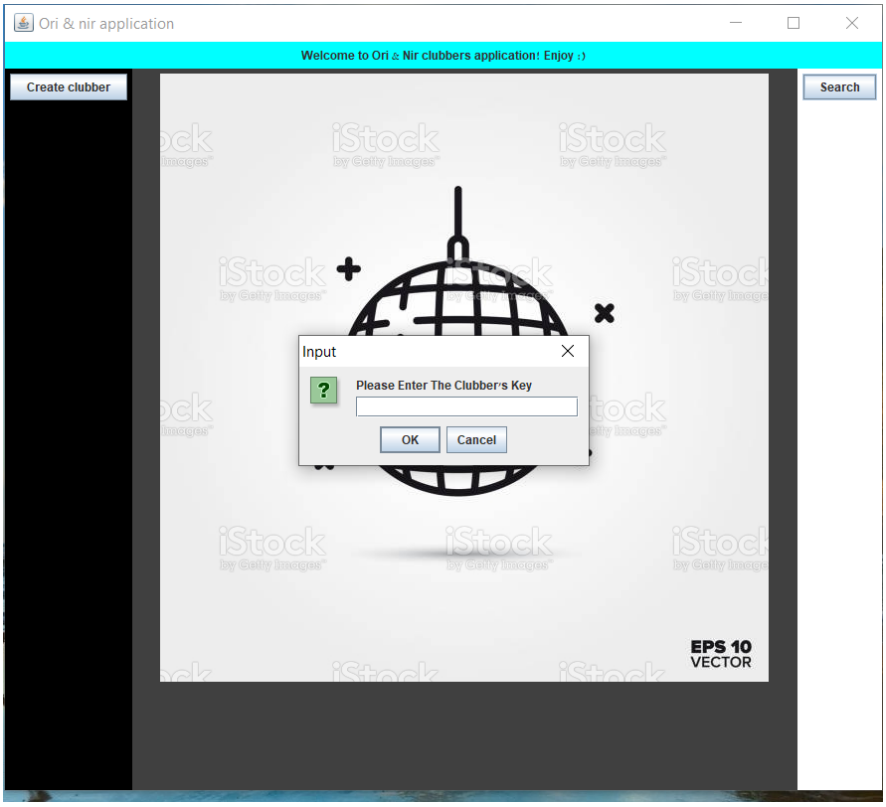
```
                {
                if(counter2!=4){redot2.setVisible(true); valid=false; }
                {
                if(valid){redot2.setVisible(false); }
                return (super.validateData()&&valid);
        //{end of validateData()


**/
```
 * commit abstract method from abstract Class ClubAbstractEntity ,A function that puts the student's details in the "belly" from text-fields.
```
        * Classes use: {@link javax.swing.JTextField},{@link java.lang.String}.
        /*
        protected void commit()
        }
                super.commit;()
                this.Studentid=new String(field.getText());
        //;{end of commit()


        **/
```
 * rollBack abstract method from abstract Class ClubAbstractEntity ,A function that returns the student's details from the "belly" to the text-fields.
```
        * Classes use: {@link javax.swing.JTextField}.
        /*
        protected void rollBack()
        }
                super.rollBack;()
                field.setText(Studentid);
        //;{end of rollBack()
//{end of Student class
```

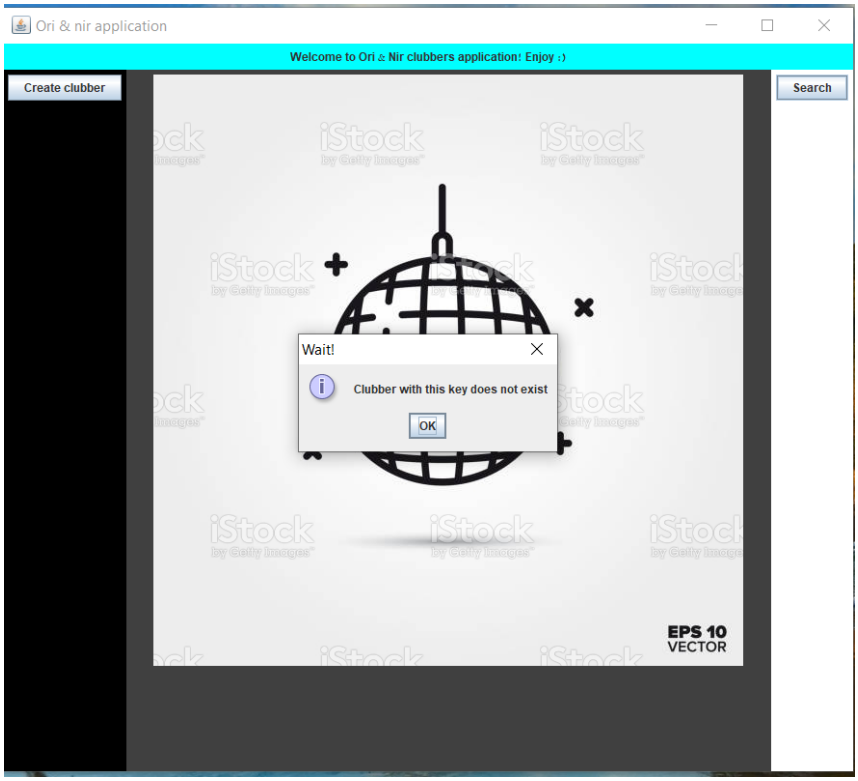
Print screen:

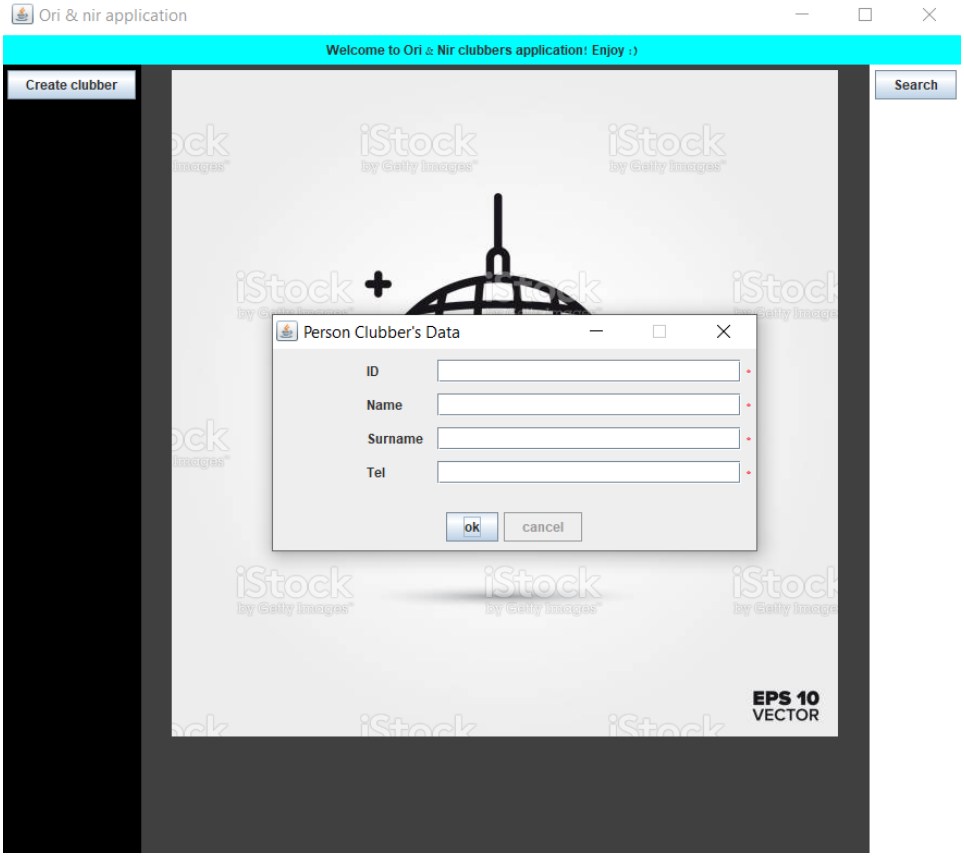


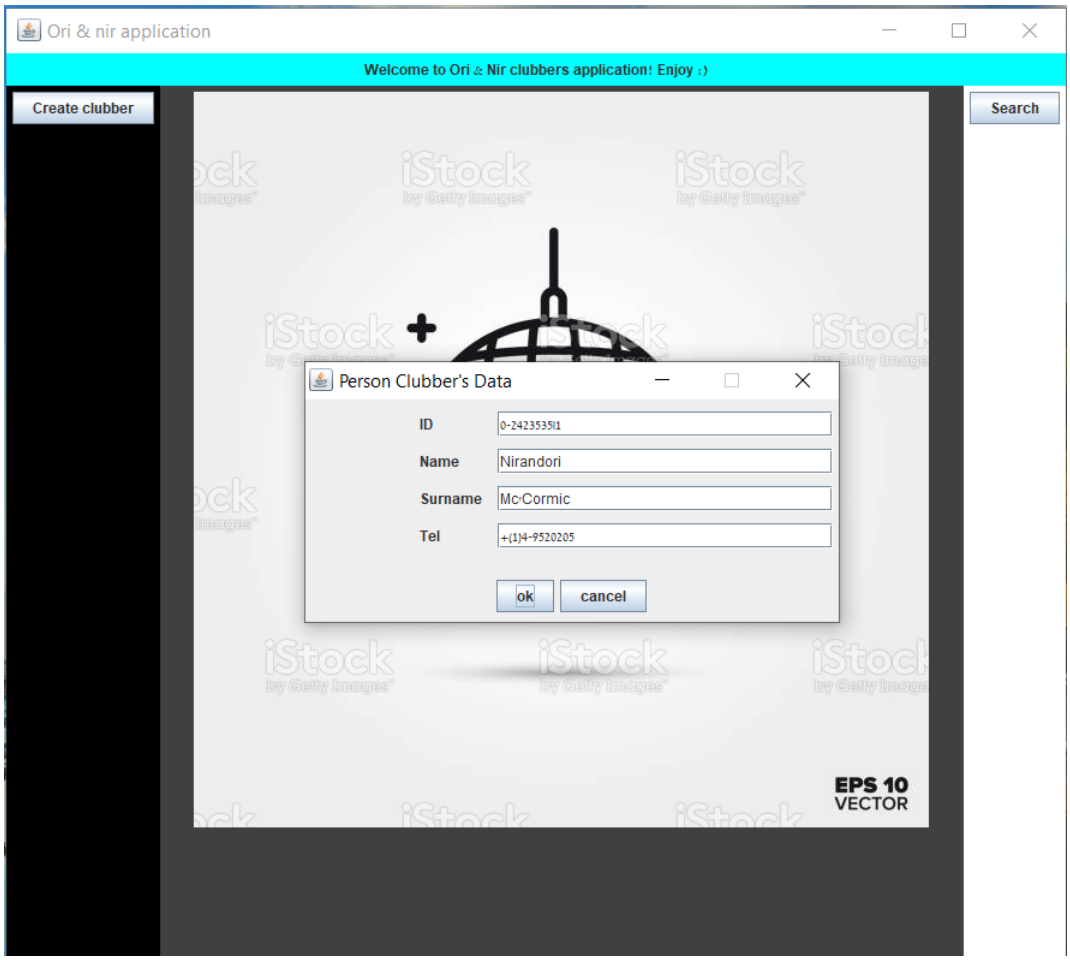


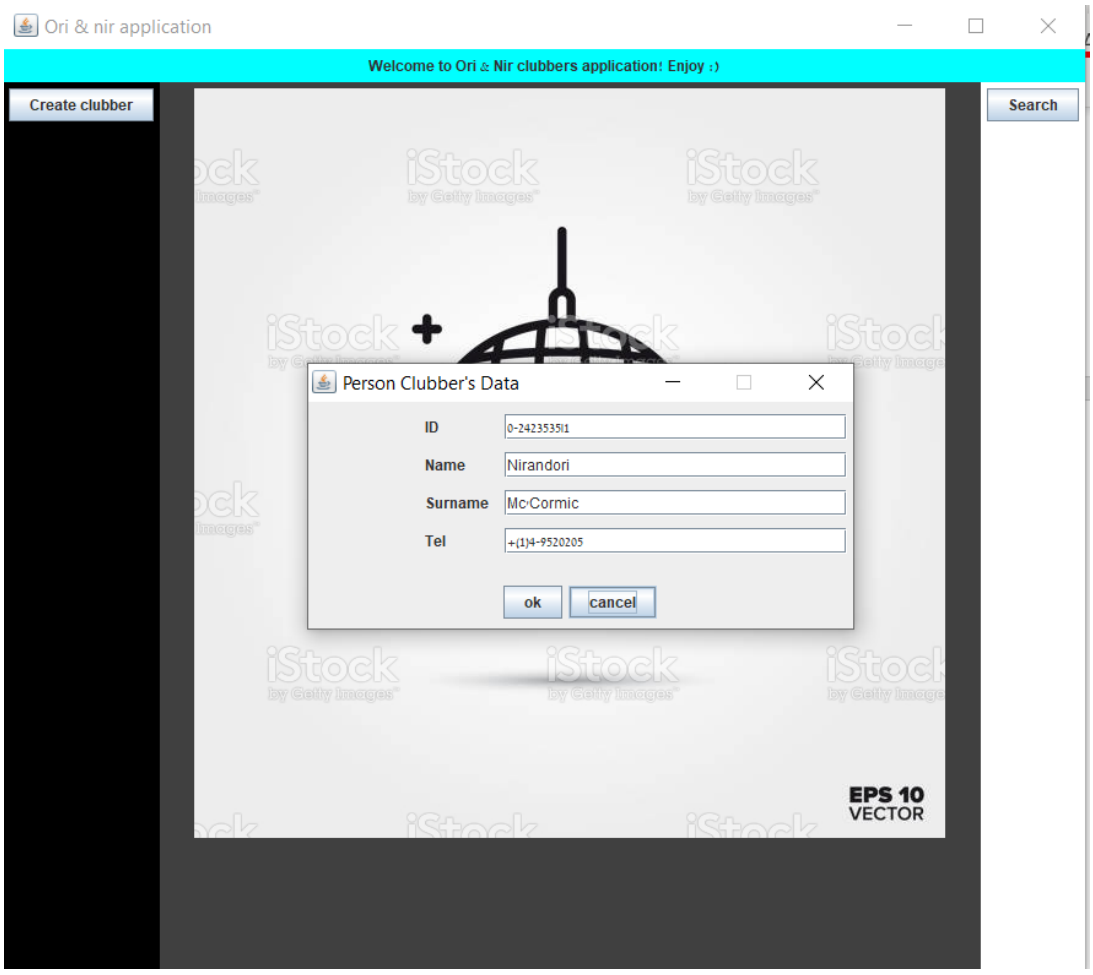

Ori & nir application
Create clubber
Search
Student Clubber's Data
ID
Name
Surname
Tel
Student ID
ok
cancel
Soldier Clubber's Data
ID
0-22233591
Name
Zohar
Surname
Couper-Berg
Tel
+(44)206-8208167
Personal No.
O/4684109
ok
cancel
Person Clubber's Data
ID
Name
Surname
Tel
ok
cancel
EPS 10
VECTOR

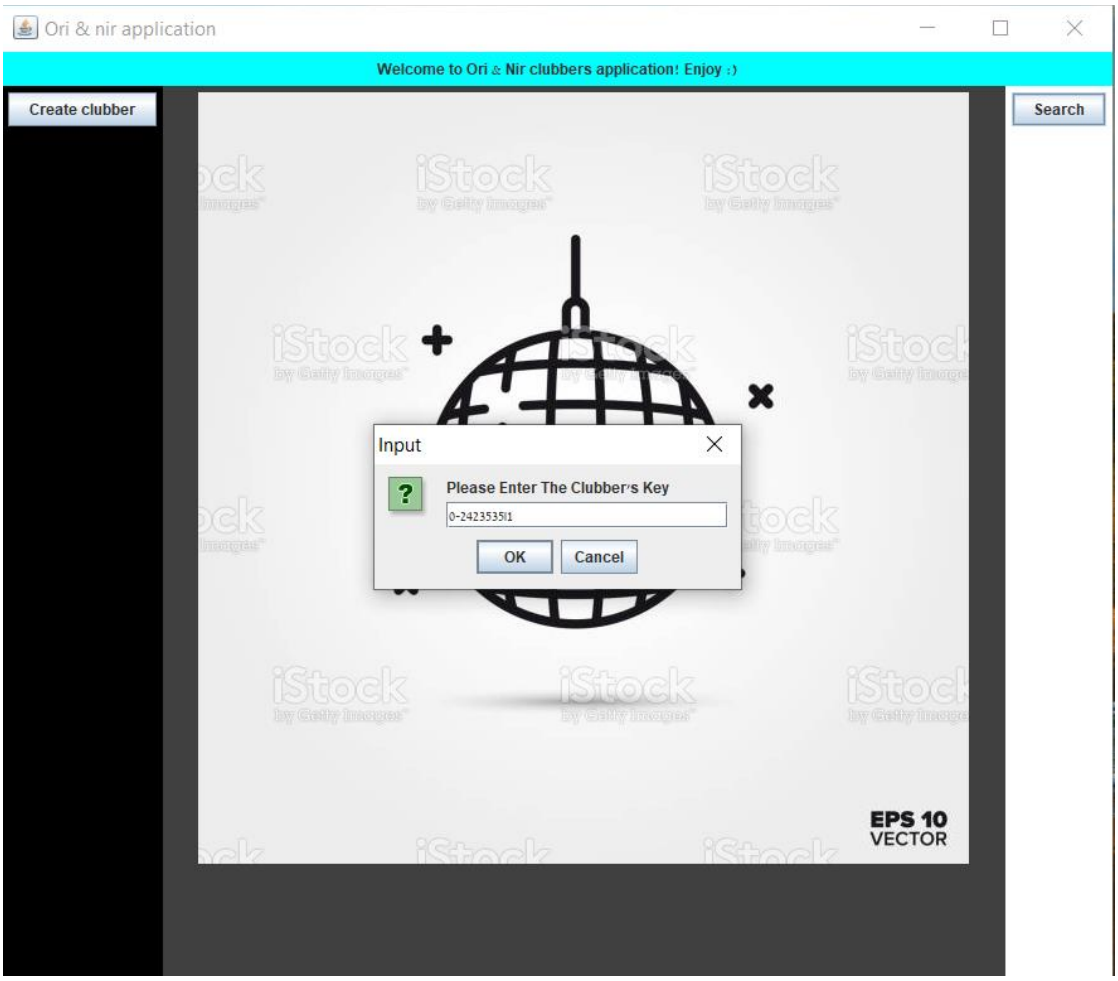

Ori & nir application
Welcome to Ori & Nir clubbers application! Enjoy :)
Create clubber
Search
Input
Please Enter The Clubber's Key
OK
Cancel
EPS 10
VECTOR

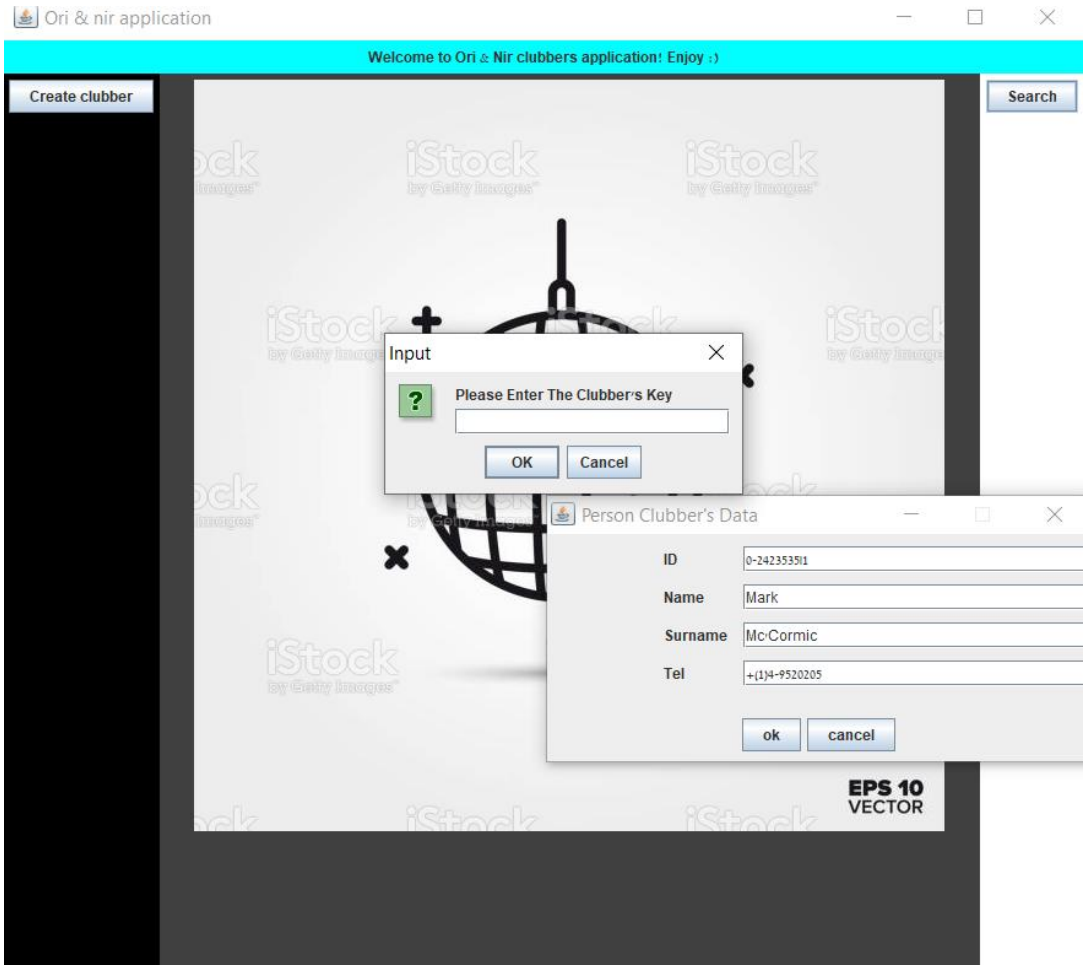

Ori & nir application
Welcome to Ori & Nir clubbers application! Enjoy ;)
Create clubber
Search
Wait!
Clubber with this key does not exist
OK
EPS 10
VECTOR

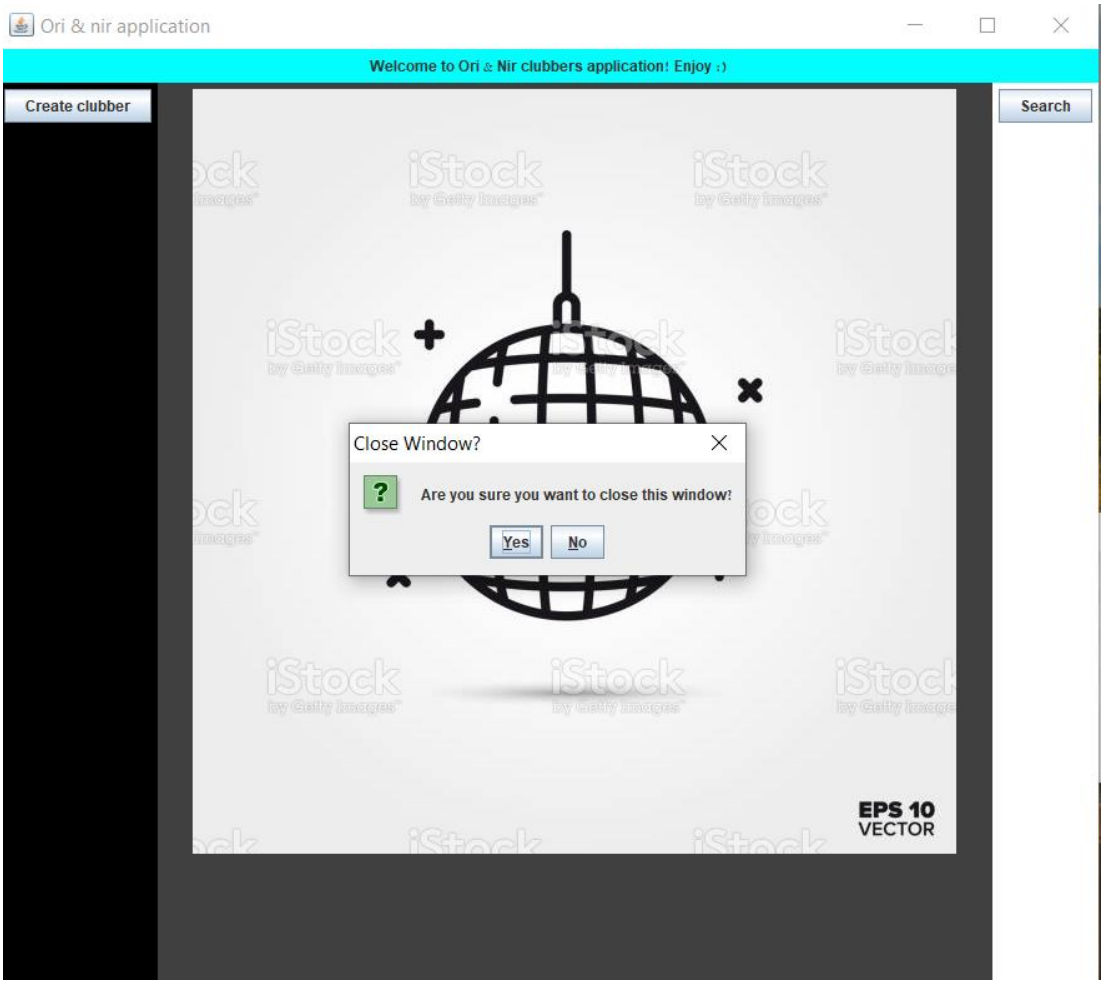

Ori & nir application
Welcome to Ori & Nir clubbers application! Enjoy ;)
Create clubber
Search
Person Clubber's Data
ID
Name
Surname
Tel
ok
cancel
EPS 10
VECTOR


Ori & nir application
Welcome to Ori & Nir clubbers application! Enjoy :)
Create clubber
Search
Person Clubber's Data
ID
0-242353511
Name
Nirandori
Surname
Mc·Cormic
Tel
+(1)4-9520205
ok
cancel
EPS 10
VECTOR


Ori & nir application
Welcome to Ori & Nir clubbers application! Enjoy :)
Create clubber
Search
Person Clubber's Data
ID
0-2423535l1
Name
Nirandori
Surname
McCormic
Tel
+(1)4-9520205
ok
cancel
EPS 10
VECTOR


Ori & nir application
Welcome to Ori & Nir clubbers application! Enjoy :)
Create clubber
Search
Input
Please Enter The Clubber's Key
0-242353511
OK
Cancel
EPS 10
VECTOR


Ori & nir application
Welcome to Ori & Nir clubbers application! Enjoy :)
Create clubber
Search
Input
Please Enter The Clubber's Key
OK
Cancel
Person Clubber's Data
ID
0-24235351
Name
Mark
Surname
McCormic
Tel
+(1)4-9520205
ok
cancel
EPS 10
VECTOR


Ori & nir application
Welcome to Ori & Nir clubbers application! Enjoy :)
Create clubber
Search
Close Window?
Are you sure you want to close this window?
Yes
No
EPS 10
VECTOR