



SCHOOL OF ECONOMICS
AND MANAGEMENT
Lund University

Department of Informatics

Magnus Wärja

VT08 **Grafiska gränssnitt**

Programkonstruktion

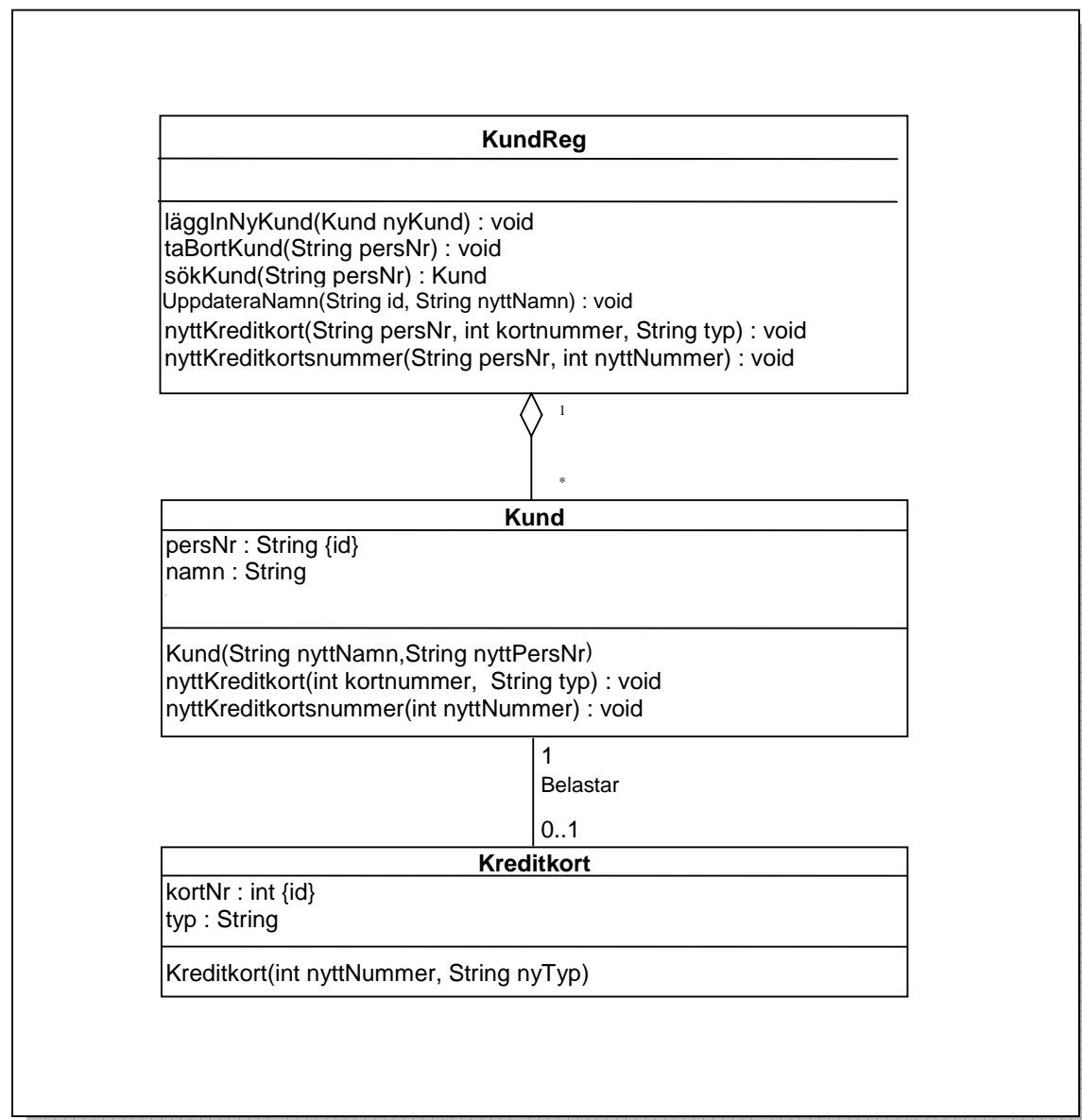


Skapa problemområdeskomponenten

Skapa en applikation och ett projekt i JDeveloper. Förslagsvis döper du båda dessa till UiLab.

Implementera nedanstående klassmodell (för enkelhetens skull, lägg ej dina filer i ett paket), koden finns på nedanstående sidor.

Värt att notera är att användandet av åtkomstmetoder (set- och getmetoder) ses som självklart och visualiseras inte i ett klassdiagram. Samma sak gäller för de instansvariabler som används för att skapa en association eller ett aggregat.



Implementering av klassen KundReg

```
import java.util.*;
public class KundReg{
    private ArrayList <Kund> kundReg;

    public KundReg(){
        kundReg = new ArrayList <Kund>();
    }

    //Lägger in en kund i listan
    public void läggInNyKund(Kund nyKund){
        kundReg.add(nyKund);
    }

    //Raderar en kund ur listan
    public void taBortKund(String id){    //id = personnummer
        Kund tmpKund;
        int i = 0; //
        boolean funnen = false;

        while (i < kundReg.size() && funnen == false){
            tmpKund = kundReg.get(i);
            if (tmpKund.avläsPersNr().equals(id)){
                kundReg.remove(i);
                funnen = true;
            }
            i++;
        }
    }

    //Söker och returnerar en kund ur listan
    public Kund sökKund(String id){
        Kund tmpKund = null;
        int i = 0;
        boolean funnen = false;

        while (i < kundReg.size() && funnen == false){
            tmpKund = kundReg.get(i);
            if (tmpKund.avläsPersNr().equals(id)){
                funnen = true;
            }
            i++;
        }
        if (funnen == false){
            tmpKund = null;
        }

        return tmpKund;
    }

    //Uppdaterar en kunds namn
    public void uppdateraKundsNamn(String id, String nyttNamn){
        Kund kund = sökKund(id);

        if(kund != null){
            kund.sättNamn(nyttNamn);
        }
    }

    //Uppdaterar en kunds kreditkortsnummer
    public void nyttKreditkortsnummer(String id, int nyttNummer){
        Kund tmpKund;
        tmpKund = sökKund(id);

        if (tmpKund != null){
            tmpKund.nyttKreditkortsnummer(nyttNummer);
        }
    }
}
```

Implementering av klassen Kund

```
public class Kund{
    private String persNr;
    private String namn;
    private Kreditkort belastar;

    public Kund(String nyttNamn,String nyttPersNr){
        sättPersNr(nyttPersNr);
        sättNamn(nyttNamn);
    }

    public void sättPersNr(String nyttPersNr){
        persNr = nyttPersNr;
    }
    public String avläsPersNr(){
        return persNr;
    }
    public void sättNamn(String nyttNamn){
        namn = nyttNamn;
    }
    public String avläsNamn(){
        return namn;
    }
    public void sättBelastar(Kreditkort nyttKreditkort){
        belastar = nyttKreditkort;
    }
    public Kreditkort avläsBelastar(){
        return belastar;
    }

    public void nyttKreditkortsnummer(int nyttNummer){
        avläsBelastar().sättKortNr(nyttNummer);
    }
}
```

Implementering av klassen Kreditkort

```
public class Kreditkort{
    private int kortNr;
    private String typ;
    private Kund belastar;

    public Kreditkort(int nyttNummer, String nyTyp){
        sättKortNr(nyttNummer);
        sättTyp(nyTyp);
    }

    public void sättKortNr(int nyttNummer){
        kortNr = nyttNummer;
    }
    public int avläsKortNr(){
        return kortNr;
    }
    public void sättTyp(String nyTyp){
        typ = nyTyp;
    }
    public String avläsTyp(){
        return typ;
    }
    public void sättBelastar(Kund nyKund){
        belastar = nyKund;
    }
    public Kund avläsBelastar(){
        return belastar;
    }
}
```

Testa din modell

Skapa en klass med en main-metod och testa din modell.

```
public class Test {
    public static void main(String[] args) {
        Kreditkort kreditkort1 = new Kreditkort(123, "visa");
        Kreditkort kreditkort2 = new Kreditkort(234, "Master card");

        Kund kund1 = new Kund("Anna Andersson", "111111-1111");
        Kund kund2 = new Kund("Beata Bengtsson", "222222-2222");

        KundReg kundReg = new KundReg();

        //Koppla kort till kund
        kreditkort1.sättBelastar(kund1);
        kreditkort2.sättBelastar(kund2);

        //Koppla kund till kort
        kund1.sättBelastar(kreditkort1);
        kund2.sättBelastar(kreditkort2);

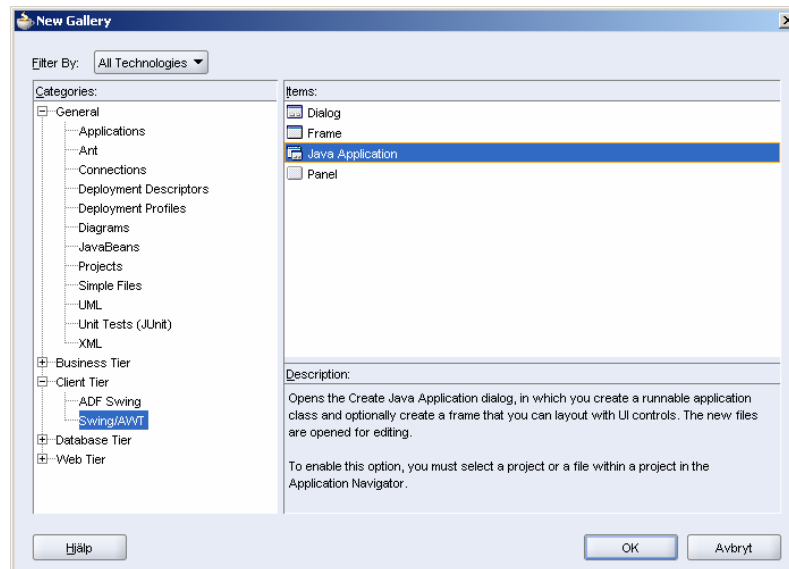
        //Lägger in kunderna i listan
        kundReg.läggInNyKund(kund1);
        kundReg.läggInNyKund(kund2);

        //Testar några av objektens metoder
        System.out.println(kreditkort1.avläsBelastar().avläsNamn());
        System.out.println(kund2.avläsBelastar().avläsKortNr());

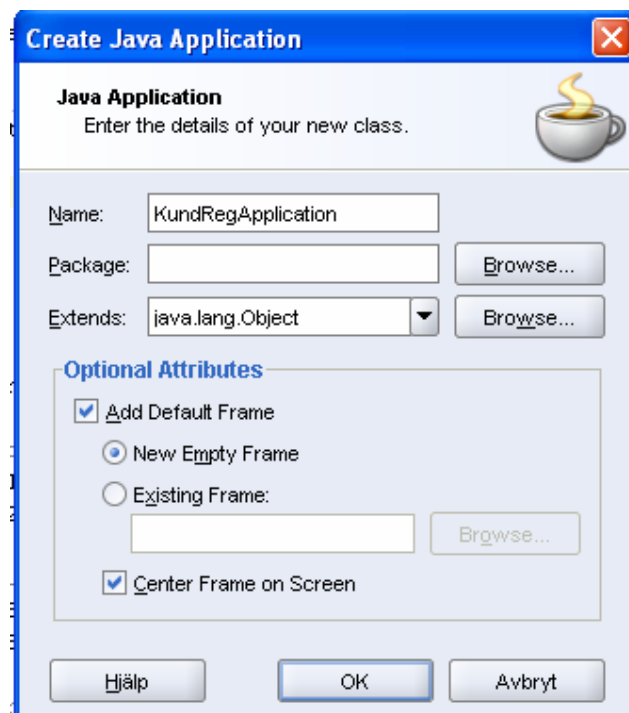
        System.out.println(kundReg.sökKund("111111-1111").avläsNamn());
        kundReg.taBortKund("111111-1111");
        Kund tmpKund = kundReg.sökKund("111111-1111");
        if(tmpKund == null){
            System.out.println("Funkar");
        }
    }
}
```

Skapa ett grafiskt gränssnitt

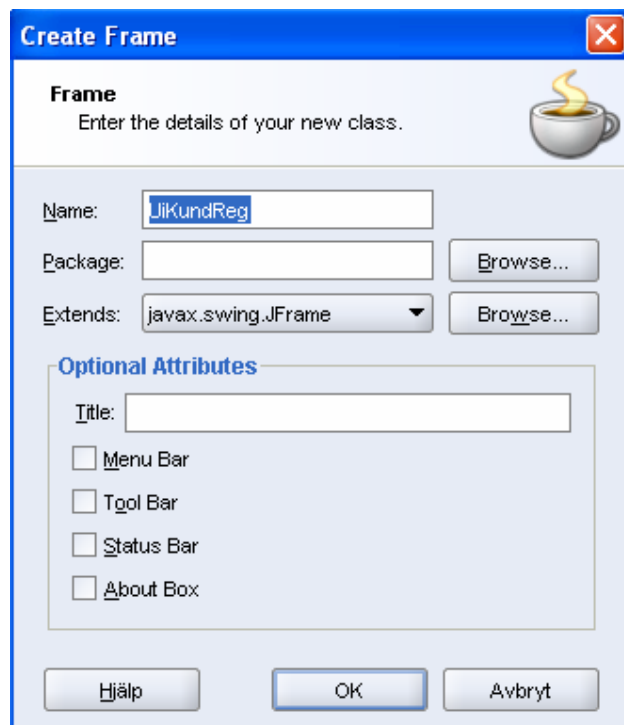
1. Markera i *System Navigator* fönstret ditt projekt och högerklicka.
2. Ur menyn välj *New...*
3. Väl ur *New-fönstret*, *Client Tier*, *Swing/AWT* och sedan *Java Application*



4. Ange ett namn på din applikation under *Name* (förslagsvis *KundRegApplikation*), klicka i *Add Default Frame* samt *New Empty Frame*.



5. Skriv in namn på ditt fönster (förslagsvis UiKundReg):

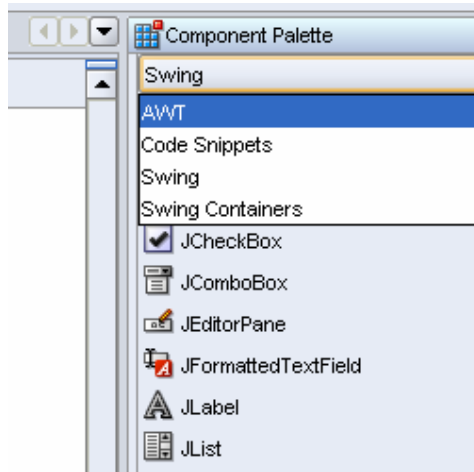






The image shows a 'Create Frame' dialog box with a blue title bar and a close button. The main area is light blue and contains the following elements:

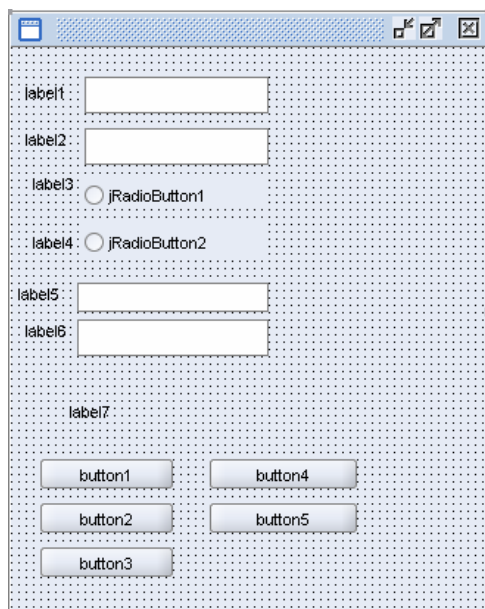
- Frame** section: A text box for 'Name' containing 'UiKundReg'. Below it are 'Package' and 'Extends' text boxes. The 'Extends' box has a dropdown menu showing 'javax.swing.JFrame'. To the right of these are two 'Browse...' buttons. Above the 'Package' and 'Extends' boxes is a small icon of a steaming cup of coffee.
- Optional Attributes** section: A group box containing a 'Title' text box and four checkboxes: 'Menu Bar', 'Tool Bar', 'Status Bar', and 'About Box'. All checkboxes are currently unchecked.
- Buttons**: At the bottom are three buttons: 'Hjälp', 'OK', and 'Avbryt'.

Utforma gränssnittet

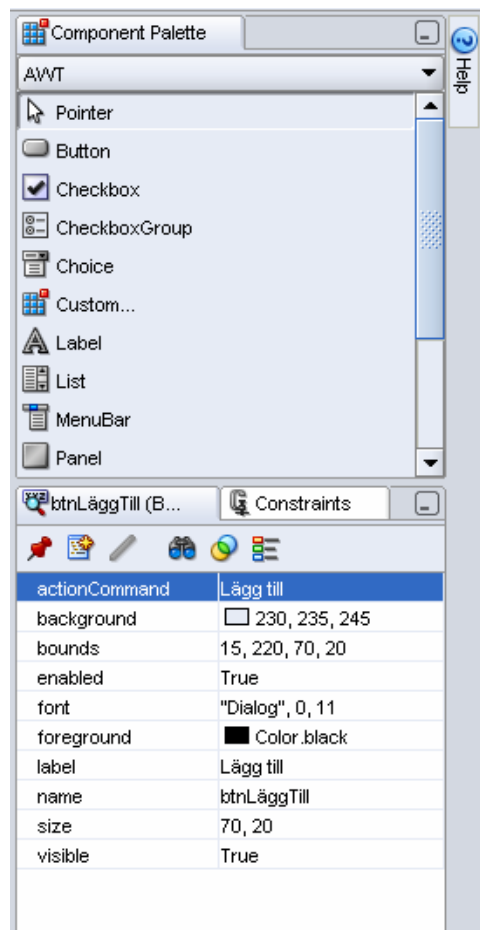
1. Om Design-läget inte öppnades automatiskt dubbelklicka på *UiKundReg.java* och välj fliken *Design* i nedre kanten av huvudfönstret.
2. Byt palette från Swing till AWT. enligt bilden nedan.



3. Markera och lägg ut fem knappar.  Button
4. Markera och lägg ut fyra textfält.  TextField
5. Markera och lägg ut sju etiketter.  Label
6. Byt palett från AWT till Swing (som i steg två) och lägg ut två radioknappar.  JRadioButton



7. Använd *Property Inspector* fönstret för att modifiera komponenternas egenskaper.



Komponent	Label	Name
button1	Lägg till	btnLäggTill
button2	Sök	btnSök
button3	Radera	btnRadera
button4	Nytt namn	btnNyttNamn
button5	Nytt kreditkortsnummer	btnNyttKortnr
label1	Namn:	lblNamn
label2	Personnummer:	lblPersonnr
label3	Har Kort	lblHarKort
label4	Har inte kort	lblHarInteKort
label5	Korttyp:	lblKorttyp
label6	Kortnummer:	lblKortnr
Label7	Respons	lblRespons
textField1		txtNamn
textField2		txtPersonnr
textField3		txtKorttyp
textField4		txtKortnummer
jRadioButton1		radioKortJa
jRadioButton2		radioKortNej

När du är klar skall din gränssnittskomponent ha nedanstående utseende. Du kan provköra genom att starta (Run) *KundRegApplication.java*.



The screenshot shows a Java Swing window with a title bar containing a Java logo and standard window controls (minimize, maximize, close). The window has a light gray background and contains the following elements:

- Input fields for "Namn:" and "Personnummer:".
- Radio buttons for "Har kort:" and "Har inte kort:".
- Input fields for "Korttyp:" and "Kortnummer:".
- A section header "Respons" in bold.
- A group of buttons: "Lägg till", "Sök", "Radera", "Nytt namn", and "Nytt kreditkortsnummer".

Koppla ihop ditt grafiska gränssnitt med problemområdet(modellen)

Vi har nu implementerat vårt grafiska gränssnitt och vårt problemområde, nu saknas bara en klass som knyter ihop det hela. Vi skulle kunna koppla oss direkt från gränssnittet till vårt problemområde. Detta vill vi dock inte av flera anledningar, främst på grund av framtida problem avseende underhåll, modifiering mm. Vårt mål är att skapa en arkitektur där gränssnitt och problemområde hålls skilda åt och inte är direkt beroende av varandra.

Vi skapar därför ytterligare en klass vars syfte är att fungera som en länk mellan det grafiska gränssnittet och vår modell.

Skapa en ny helt vanlig klass vid namn Controller och skriv in (kopiera in) nedanstående kod.

```
public class Controller{
    KundReg kundReg; //Refererar till modellen (registret)
    UiKundReg uiKundReg; //Refererar till det grafiska gränssnittet

    public Controller(KundReg ettKundReg, UiKundReg ettUiKundReg) {
        kundReg = ettKundReg;
        uiKundReg = ettUiKundReg;
    }

    public void läggInNyKund(String namn, String personnr){
        Kund tmpKund = new Kund(namn, personnr);

        kundReg.läggInNyKund(tmpKund);
    }
    public void läggInNyKund(String namn, String personnr, String korttyp, int kortnummer){
        Kund tmpKund = new Kund(namn, personnr);
        Kreditkort kreditkort = new Kreditkort(kortnummer, korttyp);

        tmpKund.sättBelastar(kreditkort);
        kundReg.läggInNyKund(tmpKund);
    }

    public void taBortKund(String id){
        kundReg.taBortKund(id);
    }

    public String[] sökKund(String id){
        Kund tmpKund;
        String[] enKund = null;
        tmpKund = kundReg.sökKund(id);

        if(tmpKund != null && tmpKund.avläsBelastar() != null){
            enKund = new String[4];
            enKund[0] = tmpKund.avläsNamn();
            enKund[1] = tmpKund.avläsPersNr();
            enKund[2] = tmpKund.avläsBelastar().avläsTyp();
            enKund[3] = Integer.toString(tmpKund.avläsBelastar().avläsKortNr());
        }
        else if(tmpKund != null){
            enKund = new String[2];
            enKund[0] = tmpKund.avläsNamn();
            enKund[1] = tmpKund.avläsPersNr();
        }
        return enKund;
    }
    public void uppdateraKundsNamn(String id, String nyttNamn){
        kundReg.uppdateraKundsNamn(id, nyttNamn);
    }
    public void nyttKreditkortsnummer(String id, int nyttNummer){
        kundReg.nyttKreditkortsnummer(id, nyttNummer);
    }
}
```

Koppla gränssnittet

Vi skall vi skapa associationen mellan vårt gränssnitt och controllerklassen, så att vi kan skicka våra händelser (ifrån knapparna, lägg till och sök) till kontrollern. Denna tar händelserna och sänder dem vidare till modellen samt ser till att eventuella returvärden (t ex sök) returneras tillbaka till vårt grafiska gränssnitt. Vi grupperar även våra ”radioknappar” så att de kan fungera ihop.

1. Gå till *Kodeditorn*, genom att gå till *UiKundReg.java* och klicka på fliken *Source* (i nedre kanten på huvudfönstret).
2. Skriv (kopiera) in det **fetstiltat** i texten nedan till klassen *UiKundReg*:

```
Import...;
public class UiKundReg extends JFrame{
    private Label lblNamn = new Label();
    private Label lblPersonnr = new Label();
    private Label lblHarKort = new Label();
    private Label lblHarInteKort = new Label();
    private Label lblKorttyp = new Label();
    private Label lblPhone = new Label();
    private TextField txtKortnr = new TextField();
    private TextField txtKorttyp = new TextField();
    private TextField txtPersonnr = new TextField();
    private TextField txtNamn = new TextField();
    private JRadioButton radioKortJa = new JRadioButton();
    private JRadioButton radioKortNej = new JRadioButton();
    private Label lblRespons = new Label();
    private Label lblRubrik = new Label();
    private JButton jButton2 = new JButton();
    private Button btnLäggTill = new Button();
    private Button btnSök = new Button();
    private Button btnRadera = new Button();
    private Button btnNyttNamn = new Button();
    private Button btnNyttKortnr = new Button();

    private ButtonGroup group = new ButtonGroup(); // Radioknappar måste grupperas!
    private Controller controller; //Koppling till klassen Controller

    public void sättController(Controller controller){
        this.controller = controller;
    }

    public UiKundReg() {
        try {
            jblInit();
        } catch (Exception e) {
            e.printStackTrace();
        }
        group.add(radioKortNej); //Radioknapparna grupperas
        group.add(radioKortJa);
    }
}
```


Skapa och ta emot händelser

1. Öppna Designfliken. (i nedre kanten på huvudfönstret).
2. Dubbelklicka på knappen *Lägg Till*.
3. I metoden som genereras, skriv in följande:

```
private void btnLäggTill_actionPerformed(ActionEvent e) {  
    String namn = txtNamn.getText();  
    String personnr = txtPersonnr.getText();  
  
    if (radioKortJa.isSelected() == true) {  
        String korttyp = txtKorttyp.getText();  
        int kortnummer = Integer.parseInt(txtKortnr.getText());  
  
        controller.läggInNyKund(namn, personnr, korttyp, kortnummer);  
    }  
    else {  
        controller.läggInNyKund(namn, personnr);  
    }  
}
```

4. Öppna Designfliken igen. (i nedre kanten på huvudfönstret).
5. Dubbelklicka på knappen *Sök*.
6. I metoden som genereras, skriv in följande:

```
private void btnSök_actionPerformed(ActionEvent e) {  
    String id = txtPersonnr.getText();  
  
    String[] tmpKund = controller.sökKund(id);  
  
    if (tmpKund != null) {  
        if (tmpKund.length == 4) {  
            txtNamn.setText(tmpKund[0]);  
            txtPersonnr.setText(tmpKund[1]);  
            txtKorttyp.setText(tmpKund[2]);  
            txtKortnr.setText(tmpKund[3]);  
            radioKortJa.setSelected(true);  
        }  
        else if (tmpKund.length == 2) {  
            txtNamn.setText(tmpKund[0]);  
            txtPersonnr.setText(tmpKund[1]);  
            radioKortNej.setSelected(true);  
        }  
    }  
    else {  
        lblRespons.setText("Kunden saknas i registret");  
    }  
}
```

Vi kan nu skicka våra händelser vidare till controllerklassen, som har ansvaret att vidarebefordra dem till modellen. Controllerklassen ser även till att returnera eventuella returvärden tillbaka till gränssnittet och allt sker utan att gränssnittet är direkt kopplat till vår modell.

Kör din applikation

Applikationens (KundRegApplication.java) uppgift är nu att skapa ett Kundregister, skapa ett fönster och skapa kontrollern. Dessa skall sedan kopplas ihop med varandra. Applikationen lämnar sedan över till användaren som styr programmet via händelser. Ett undantag finns dock, det är avsluta-händelsen som skickas till applikationen.

1. Öppna KundRegApplication.java
2. Genomför de förändringar som är markerade med fetstil.

```
import java.awt.Dimension;
import java.awt.Toolkit;
import javax.swing.JFrame;
import javax.swing.UIManager;

public class KundRegApplication {
    UiKundReg gränssnittet;

    public KundRegApplication() {
        gränssnittet = new UiKundReg();

        Dimension screenSize = Toolkit.getDefaultToolkit().getScreenSize();
        Dimension frameSize = gränssnittet.getSize();
        if (frameSize.height > screenSize.height) {
            frameSize.height = screenSize.height;
        }
        if (frameSize.width > screenSize.width) {
            frameSize.width = screenSize.width;
        }
        gränssnittet.setLocation( ( screenSize.width - frameSize.width ) / 2, ( screenSize.height -
            frameSize.height ) / 2 );
        gränssnittet.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );
        gränssnittet.setVisible(true);
    }

    public static void main(String[] args) {
        try {
            UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName());
        } catch (Exception e) {
            e.printStackTrace();
        }

        KundRegApplication kundRegApplication = new KundRegApplication();
        kundRegApplication.koppla(); //Metoden finns nedan!
    }

    private void koppla(){
        KundReg kundReg = new KundReg();
        Controller controller = new Controller(kundReg, gränssnittet);
        gränssnittet.sättController(controller);
    }
}
```