Labbrapport

Laboration 2: Restaurangbokning

Anders Ullström
940803
24 april 2022

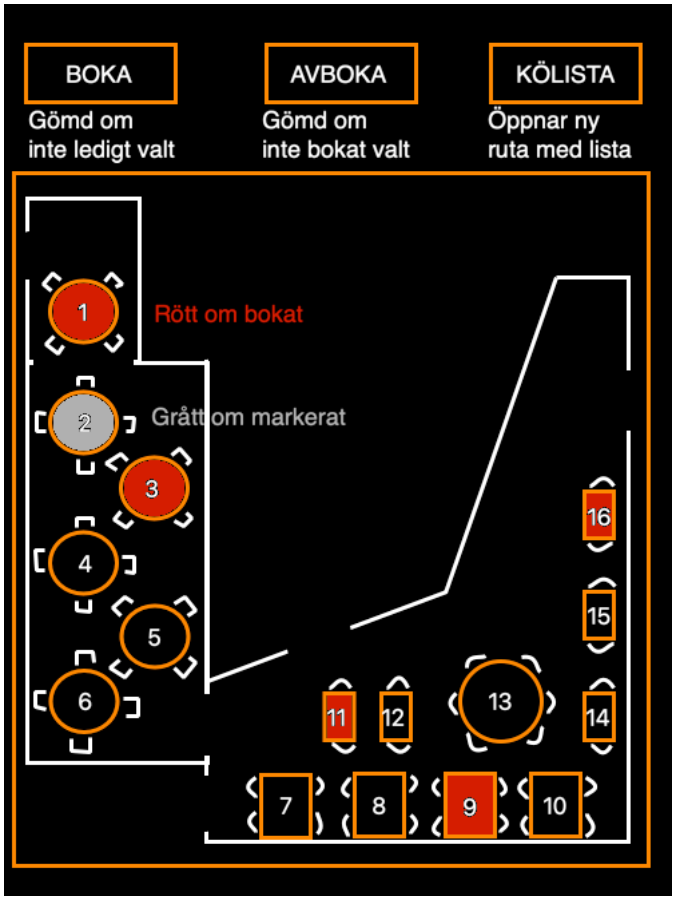
Introduktion

I denna laboration var syftet att skapa ett program som hanterar bordsbokningen hos en restaurang. Detta har inneburit att skapa ett grafiskt användargränssnitt. Denna laboration innehöll följande kravspecifikation:

- Personalen ska kunna boka ett bord.
- Personalen ska kunna avboka ett bord.
- Något kösystem för personal.
- Programmet ska vara implementerat med Backend-Frontend eller Model-View-Controller (med eller utan observer pattern).
- Det ska gå att högerklicka på skärmen.
- Det ska vara anpassat för färgblinda.

Grafisk design

Den grafiska designen bygger i grund och botten på egendesignade PNG-filer som ritats med hjälp av olika verktyg såsom applikationen Paint S. Projektet inleddes med att göra en skiss över hur gränssnittet skulle se ut. Detta gjordes för att snabbt kunna avveckla dåliga idéer utan att behöva skriva om mängdvis med kod. Tillslut så blev en slutgiltig skiss klar som kan ses i Figur 1. Tanken var att användaren skulle kunna klicka på ett valfritt bord och beroende på vilket stadium bordet hade så skulle olika knappar bli synliga. Om bordet var rött så skulle en knapp med texten ”AVBOKA” bli



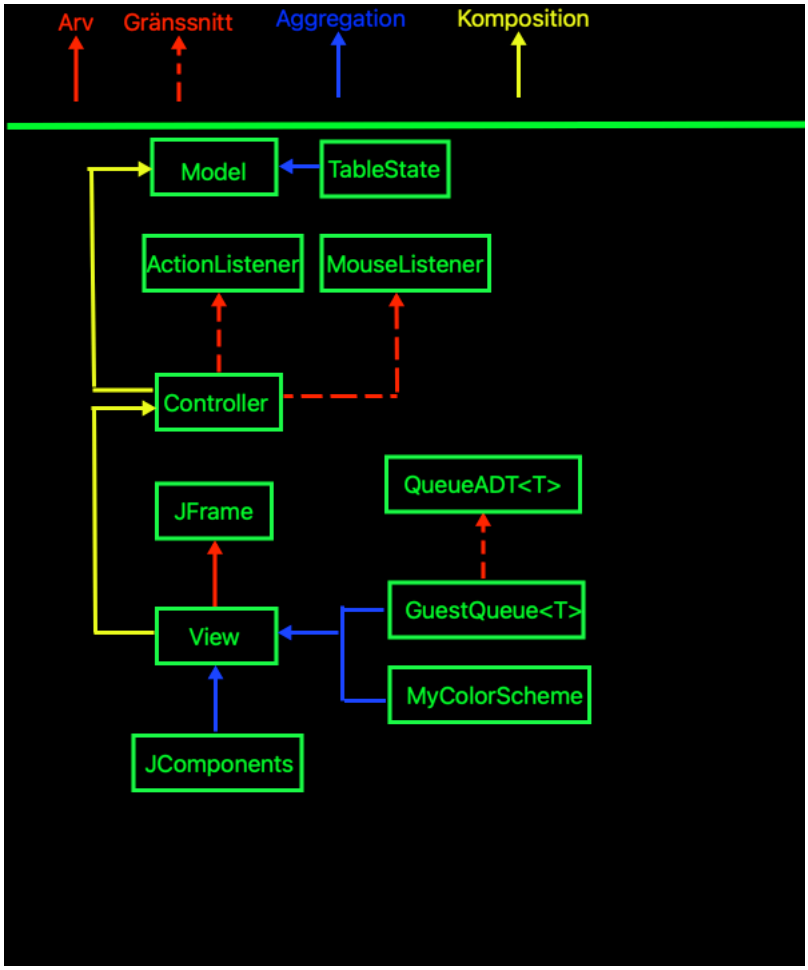
Figur 1: Skiss över gränssnittet.

synlig och om bordet var grönt så skulle en knapp med texten "BOKA" bli synlig. Tanken var även att det skulle finnas en knapp med texten "KÖLISTA" som aktiverar en gömd panel med möjlighet för användaren att skriva in gästernas namn när restaurangen är full eller av någon annan anledning. Denna design skrotades eftersom det ansågs att kölistan alltid kunde vara synlig för användaren.

Under den grafiska designen så har åtanke tagits kring vad användaren ska lägga märke till när den ser användargränssnittet. Det skulle vara uppenbart hur programmet skulle användas även för en förstagångsanvändare.

Det har lagts mycket energi på att minska den motoriska belastningen. Det var viktigt att knapparna var rätt utplacerade och med rätt avstånd samt storlek.

Under utvecklingens gång så ändrades färgschemat så att det skulle vara anpassat för färgblinda. Borden fick då bli blåa om de var lediga, röda om de var bokade och gula om de var markerade. Detta uppnåddes med hjälp av att göra forskning kring hur färgblinda upplever färger.



Figur 3: UML-diagram.

Den mentala belastningen minskades genom att se till att användaren med så få steg som möjligt kunde uppnå sitt ändamål, men vissa steg fick lov att vara lite längre för att bibehålla

pedagogiken i programmet. Den visuella belastningen minskades genom att välja färger och design som passar i en mörk restaurangmiljö. Det blev ett mörkt färgtema i grunden och sedan färglades komponenterna med tillfredsställande färger för ögonen så att det skulle bli gemytligt för användaren och inte överansträngande för ögonen.

Programdesign

Programdesignen följer MVC (Model-View-Controller) och består i grund och botten av tre klasser med samma namn: Model, View och Controller. Detta innebär att programmet är uppdelat i olika sektioner som fyller olika syften. Model agerar som en slags lagring för variabler som ändrar sig under programmets gång. Controller agerar som själva kontrollen för programmet och passerar data till Model och View för att utföra uppgifter som hör hemma i dessa klasser. View hanterar den grafiska representationen av programmet och är det användaren ser när den använder programmet. Se Figur 3 för en representation av klassernas korrelation till varandra i form av ett UML-diagram.

Implementation

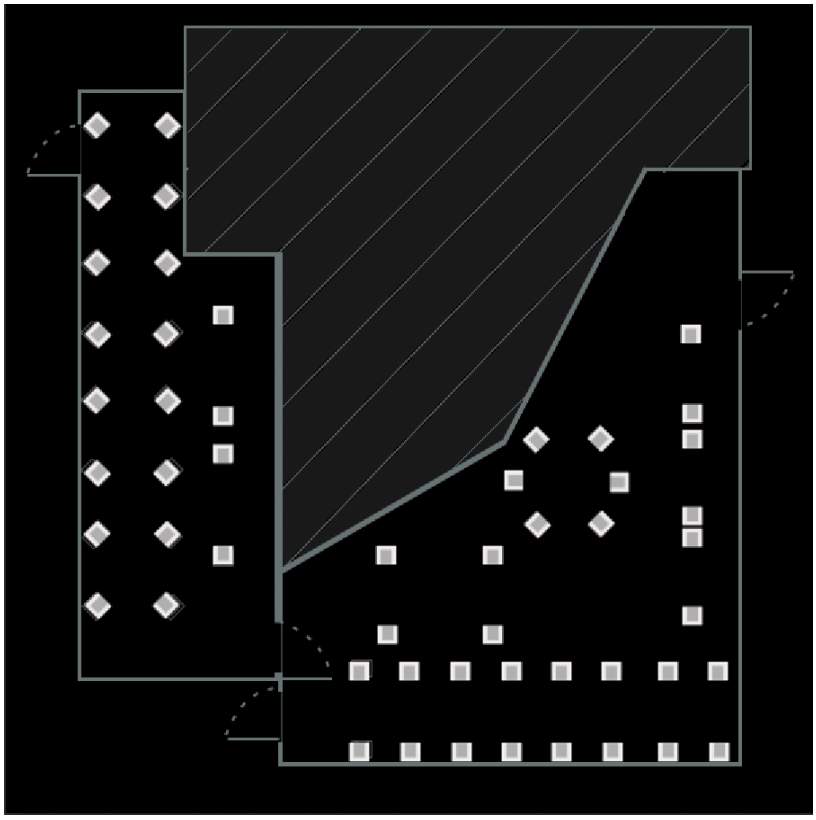
Utifrån dem redan nämnda klasserna ovan i UML-diagrammet så innehåller dessa även en mängd olika objekt och metoder.

Klassen View:

Denna klass är det användaren ser med ögonen när den använder programmet. Den består av en mängd olika barn till JComponent. Först och främst så ärver denna klass ifrån JFrame för att kunna rita upp en skärm som sedan programmet ritas i. Det mesta i denna klass sker redan i konstruktorn vid programmets start. Då instansieras komponenter såsom JList, JPanel, JLayeredPane, JButton, ImageIcon, JLabel, JTextArea, GuestQueue, JScrollPane, JCalendar.

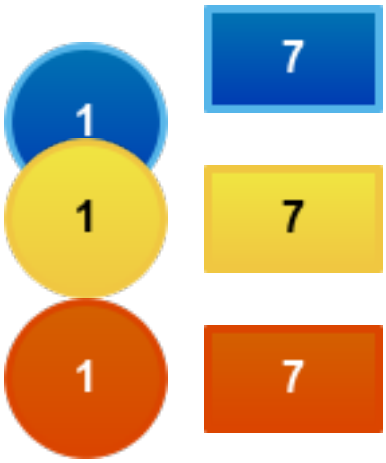
Instansen av JList lagrar data i form av String och hämtar denna data ifrån klassen GuestQueue som implementerar gränssnittet GuestADT som har fyra metoder. Dessa metoder heter enqueue() som lägger till gäst sist i kön, dequeue() som tar bort gäst ifrån kön beroende på vilken gäst som är markerad i listan, isEmpty() som returnerar true om kön är tom och getModel() som returnerar instansen av DefaultListModel som lagrat alla gäster i som olika strängar. Denna JList som har fått sin data ifrån kön läggs sedan till i en JScrollPane. Denna JScrollPane med en tillhörande JButtons med namn ”Sök bord” och ”Ta bort” läggs i sin tur in i en egen JPanel. Instansen av JTextArea och en instans av JButton med namn ”Lägg till” läggs sedan till i en annan JPanel. Dessa två paneler läggs slutligen in i en panel med BorderLayout.EAST. Detta för att skapa ett fint upplägg med knappar och rutor (se Figur 8).

Den självaste restaurangdelen av gränssnittet som uppvisar alla bord består av en instans av JLayeredPane med BorderLayout.CENTER bara för att möjliggöra olika lager med komponenter ovanpå varandra. I denna instans så läggs alla PNG-filer in i form av bakgrundsbild (se Figur 4), blåa bord (se Figur 5 och 6). Alla blåa bord läggs in i två arrayer med plats för restaurangens alla runda och fyrkantiga bord. Det användes två arrayer för att skapa separation mellan de fyrkantiga och runda borden. Alla index i arrayerna instansieras



Figur 4: Bakgrundsbild.

sedan som ImageIcons av PNG-filer med klassen Controller's implementerade MouseListener på vardera ikon. Alla instanser av JButton får även Controller's ActionListener med egna ActionCommands. Det finns ett undantag med knapparna man bokar och avbokar med. Dessa knappar får inget ActionCommand förrän användaren har klickat på ett bord, ledigt eller icke-ledigt. Detta är för att programmet ska kunna veta vilken typ av bord som har blivit klickat, om det är runt eller fyrkantigt, bara så att dess PNG-fil skall kunna bytas ut mot en likvärdig PNG-fil av annan kulör. Slutligen så läggs dessa ImageIcons in i två nya arrayer av klassen JLabel med samma storlek och samma separation som föregående arrayer. Detta gjordes för att kunna placera bilderna direkt i instansen av JLayeredPane.



Figur 5.

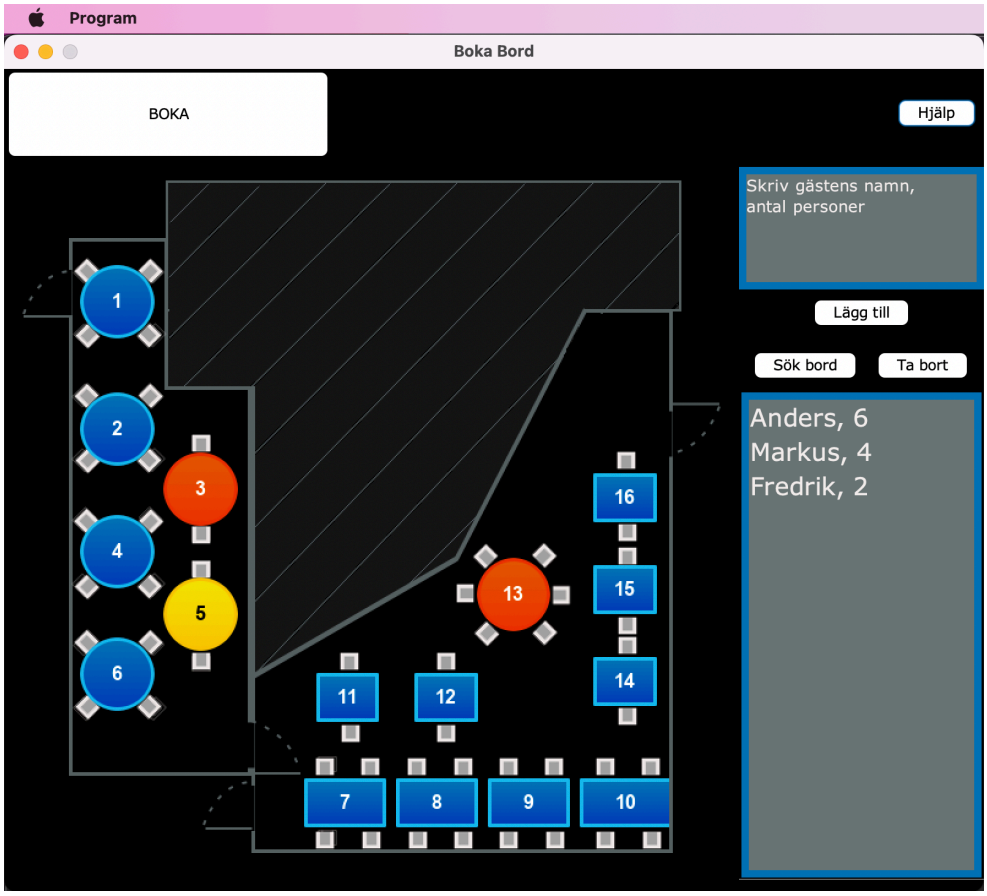
Figur 6.

En sista instans av en JPanel skapas med BorderLayout.NORTH där tre JButtons läggs till. En knapp är för att boka och denna förblir dold tills användaren klickat på ett blått bord. En annan knapp är för att avboka och även denna förblir dold tills användaren klickat på en rött bord. Den tredje och sista knappen är för att visa/dölja en JPanel med namn "Hjälp". Denna panel visar ett guidefönster vad alla olika färger på borden betyder (se Figur 7).

Figur 7: Guidefönster.

Klassen View har en rad av olika metoder också. Dessa existerar för att kunna förändra olika komponenters värden ifrån klassen Controller under programmets gång.

Det finns en sista klass som heter MyColorScheme som endast är till för att färglägga olika komponenter med klassens definierade färger från de implementerade metoderna.



Figur 7: Färdig produkt av programmet i MacOS.

Klassen Controller:

Denna klass består till mesta del av metoder som kallas på från instanserna av Model och View. Denna klass agerar som ett bollplank. Eftersom den implementerar gränssnitten ActionListener och MouseListener så måste även dessa gränssnitts metoder implementeras. Logiken är som följande: Användaren klickar på ett bord och då skickas ett event till muslyssnaren som sedan avgör vilken PNG-fil som har blivit klickad på och utför den if-sats som är ekvivalent till detta event. Genom att skriva e.toString().contains(namn) i en if-sats så kunde det avgöras vilken PNG-fil som hade blivit tryckt på grund av att alla JLabels som innehåller PNG-filerna har ett unikt namn (se Figur 8). Det ska

```
// OBS Siffran indikerar antal platser vid bordet.
roundTableLabel[0].setName("Green First 4");
roundTableLabel[1].setName("Green Second 4");
roundTableLabel[2].setName("Green Third 2");
roundTableLabel[3].setName("Green Fourth 4");
roundTableLabel[4].setName("Green Fifth 2");
roundTableLabel[5].setName("Green Sixth 4");
roundTableLabel[6].setName("Green Thirteenth 6");
```

Figur 8: Representation av de 6 runda bordens namn.

tilläggas att muslyssnaren enbart använder en av alla implementerade metoder och denna är mousePressed(). I denna metod så kollas först om vänster- eller högerklick har blivit nedtryckt. Om det är ett vänsterklick så triggas metoden controlTables() och om det är ett högerklick så triggas metoden calendar().
I metoden controlTables() som tar emot ett event så jämförs detta event i en rad olika if-satser som kan delas upp i olika sektioner.

- Den första sektionen av if-satser jämför eventet med de blåa bordens JLabels satta namn. När en av dessa if-satser triggas så ändras instansen av TableState vid namn greenState till TableState.Marked i klassen Model med metoden setGreenState(). Sedan så kallas metoden enableBookButton() för att göra bokningsknappen synlig. Sedan beroende på om det är ett runt blått bord eller fyrkantigt blått bord så kallas metoderna setRoundTableStatus() eller setSquareTableStatus() som ändrar PNG-filen till den ekvivalenta gula typen som representerar att bordet är markerat och sedan kallas metoderna setBookButtonActionCommand(”roundBook”) eller setBookButtonActionCommand(”squareBook”). Borden får även ett nytt namn som blir identiskt som innan fast denna gången börja namnet med ”Red” för att det ska gå att kontrollera om bordet är bokat eller inte.
- Den andra sektionen fungerar snarlikt fast omvänt. Denna gör på samma sätt fast för röda bord. Om ett rött bord blir klickat så syns avbokningsknappen istället och det blir ett eget ActionCommand.
- Den tredje sektionen är till för markerade gula bord. Det är för att det ska gå att avmarkera bordet om det blivit felmarkerat av användaren. Denna sektion byter ut det gula bordet till PNG-filen den var innan, antingen blå eller röd. Boka- eller avbokningsknappen döljs även igen vid detta skede och TableState i Model ändras till TableState.Unmarked. Bordets namn ändras även till det föregående namnet i detta skede.

När detta skede är över och användaren har bestämt sig för vilket bord den vill boka/avboka så kan användaren klicka på boknings- eller avbokningsknappen. Detta är nu möjligt att styra på grund av att knapparna fått ett ActionCommand. När användaren klickar på bokningsknappen så blir bordet rött tills användaren väljer att avboka. När användaren klickar på avbokningsknappen så blir bordet blått tills användaren väljer att boka igen.

Kölistan följer en mängd regler och logik ifrån klassen Controller. Detta innebär att om användaren markerar en gäst i kön och trycker på sök bord så kommer funktionen getTableSpots() kallas på i klassen View. Denna metod jämför antalet gäster i sällskapet med antalet platser på alla bord och kommer att gömma alla bord som är för små till storleken sällskapet är på. Regex har använts för att kunna ta bort alla bokstäver ifrån kölistans markerade gäst och enbart ha kvar siffran som indikerar antalet gäster i sällskapet. Detta underlättade getTableSpots()-metodens funktionalitet.

Om användaren högerklickar på ett blått bord så kommer en kalender upp. Denna kalender är bara för att visa framtidsvisionen för detta program och fyller ingen funktion förutom att det blir estetiskt tilltalande.

Klassen Model:

Denna klass består av variabler, boolean, Strings och Enums. Denna klass lagrar senast markerade bord, hur många bokningar, om ett bord är markerat eller inte och om kölistan är markerad eller inte. Denna klass kontrollerar även bordens namn och ändrar det vid behov. Klassen Controller hämtar och ändrar alla variabler härifrån och använder sig av variablerna när den behöver.

Problem

Det uppstod lite funderingar när det insågs att MouseListener inte hade några inbyggda metoder för att sätta ett ActionCommand som det går med ActionListener. Detta löstes tillslut genom att jämföra eventet på detta vis:

```
if(e.toString().contains(namn) { }
```

Det gick alltså att hitta PNG-filen som ville bytas ut genom att sätta ett namn på varje JLabel, vilket fungerade precis som ett ActionCommand eftersom alla namn är unika i projektet.

Ett annat problem var att bygga olika lager med JComponents. Detta löstes tillslut genom lite googlande och då hittades den inbyggda klassen JLayeredPane som är skapad till detta syfte. Då gick det lätt att placera ut komponenter ovanpå andra och flytta runt dem på skärmen med metoden setBounds() utan att påverka andra komponenters position.

Slutsats

I denna laboration så krävdes det att tänka till lite för att samspelet mellan komponenterna skulle falla på plats. Det var lätt att förvirras ibland och oväntade buggar skedde. Det var dock ingenting som ej var överkomligt och om inte annat så var det lärorikt. Det är underhållande att skapa grafiska användargränssnitt för att det blir så tydligt alla ändringar som görs eftersom det blir visuellt för ögat. Det är kul att se vad som går att skapa med inbyggda komponenter och att det faktiskt går att göra det kosmetiskt tilltalande på egen hand.

Referenser

- www.stackoverflow.com

- https://www.extensis.com/blog/how-to-design-for-color-blindness
- www.google.com *För att skapa ett färgschema anpassat för färgblinda.*
- https://buttonoptimizer.com/ *För att skapa egna knappar som PNG-fil.*
- Karlstads Universitet. (2022). *Föreläsning 1: Introduktion.* https://kau.instructure.com/courses/13875/pages/forelasning-1-introduktion?module_item_id=386737

  Kurs: Grafiska Användargränssnitt
- Karlstads Universitet. (2022). *Föreläsning 2: Grundläggande Java och Objektorientering.* https://kau.instructure.com/courses/13875/pages/forelasning-2-grundlaggande-java-och-objektorientering?module_item_id=390976

  Kurs: Grafiska Användargränssnitt
- Karlstads Universitet. (2022). *Föreläsning 3: Swing Widgets.* https://kau.instructure.com/courses/13875/pages/forelasning-3-swing-widgets?module_item_id=386738

  Kurs: Grafiska Användargränssnitt
- Karlstads Universitet. (2022). *Föreläsning 4: Swing Layouts.* https://kau.instructure.com/courses/13875/pages/forelasning-4-swing-layouts?module_item_id=386739

  Kurs: Grafiska Användargränssnitt
- Karlstads Universitet. (2022). *Föreläsning 5: Rita i Swing.* https://kau.instructure.com/courses/13875/pages/forelasning-5-rita-i-swing?module_item_id=386740

Kurs: Grafiska Användargränssnitt
- Karlstads Universitet. (2022). *Föreläsning 6 MVC och Observer Pattern.* https://kau.instructure.com/courses/13875/pages/forelasning-6-mvc-och-observer-pattern?module_item_id=386741

  Kurs: Grafiska Användargränssnitt
- Karlstads Universitet. (2022). *Föreläsning 7 Interaktionsdesign.* https://kau.instructure.com/courses/13875/pages/forelasning-7-interaktionsdesign*?module_item_id=386742*

  Kurs: Grafiska Användargränssnitt