Laboration B - Trådar

Syftet med laborationen är att du ska träna på att använda trådar.

Grundläggande uppgifter

Uppgift 1 – Textvisning med Timer

Klasserna ***Text*** och ***TextViewer*** är givna. Klassen *Text* kapslar in information om en text – själva texten och hur texten ska visas i ett fönster (textfärg, bakgrundsfärg och font). Klassen *TextViewer* visar objekt av typen Text.

Placera filerna *Text.java* och *TextViewer.java* i paketet *labB*.
Skapa en klass med namnet ***Uppgift1***. Klassen *Uppgift1* ska implementera interfacet *ActionListener*.

```
public class Uppgift1 implements ActionListener {
```

Interfacet *ActionListener* finns i paketet java.awt.event och måste importeras.

Klassen *Uppgift1* ska innehålla följande instansvariabler:

```
private ArrayList<Text> texter = new ArrayList<Text>(); // importera java.util
private TextViewer viewer = new TextViewer(); // labB, ingen import
private Timer timer = new Timer(3000, this); // importera javax.swing
int index = 0;
```

Nu är det dags att skriva konstruktorn. I konstruktorn ska du lägga till Text-objekt i listan texter (det är dessa objekt som ska visas i TextViewer-objektet) och därefter starta timern.

```
public Uppgift1() {
   String s1 = "Att lära utan att tänka är meningslöst, att tänka utan att lära är farligt";
   String s2 = "Den, som tror sig veta allt, vet ingenting ";
   String s3 = "Hellre olärd och klok, än lärd och dum ";
   String s4 = "Sanna ord är inte alltid vackra, vackra ord är inte alltid sanna ";
   texter.add(new Text(s1, Color.blue, Color.yellow, new Font("Serif", Font.BOLD, 14)));
   texter.add(new Text(s2, Color.white, Color.darkGray, new Font("SansSerif", Font.PLAIN, 24)));
   texter.add(new Text(s3, Color.yellow, Color.darkGray, new Font("Monospaced", Font.ITALIC, 28)));
   texter.add(new Text(s4,, Color.red, Color.blue, new Font("Dialog", Font.BOLD + Font.ITALIC, 20)));
   timer.start();
}
```

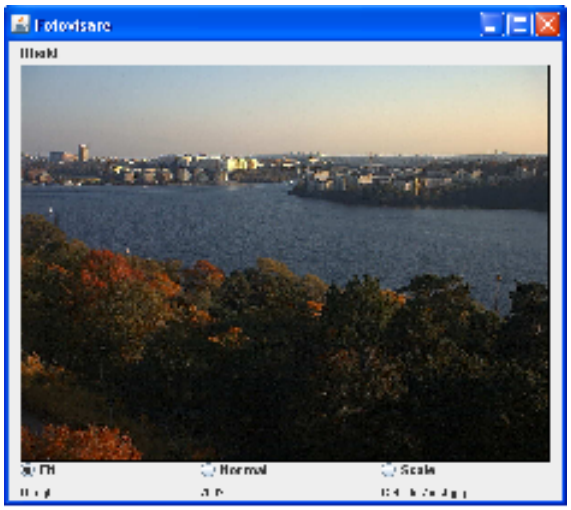
Nu återstår endast att skriva metoden *actionPerformed*. *actionPerformed* anropas av *Timer*-objektet (indirekt via händelsetråden) och ska se till att en ny text visas.

```
public void actionPerformed(ActionEvent e) {
    if( index < texter.size() ) {
        viewer.setText(texter.get(index)); // Text i position index visas
        index++; // Position ökas med 1 (ny Text visas vid nästa anrop)
    } else {
        timer.stop(); // Inga fler texter att visa – stoppa timern
    }
}
```

Testkör *Uppgift1* genom att skapa ett objekt av klassen:

```
Uppgift1 u1 = new Uppgift1();
```

Uppgift 2 – Bildspel1 med Thread



Klasserna ***Fotovisare*** och ***ScaleIcon*** är givna och är till för att visa ett Foto-objekt (se labA). Klassen *Fotovisare* är genererad med hjälp av ett *GUI-verktyg*. Därför kan koden vara lite svårläst. Viktiga metoder i *Fotovisare* är *setFoto* och *setState*.

Placera *Fotovisare.java, ScaleIcon.java, TestScaleIcon.java* och *DemoFotovisare.java* i paketet *labB*. Du måste ha filerna Foto.java och Album.java i paketet labA. Om du inte gjort hela labA finner du filerna i DA111TLAHT08.pdf.

Programmen *TestScaleIcon.java* och *DemoFotovisare.java* demonstrerar de givna filerna.

Testkör programmet DemoFotovisare. För att bilder ska visas krävs att du ändrar bildfilerna i programmet till bildfiler (jpg/gif/png) som finns på din dator.

När du skapar ett Fotovisare-objekt så ska du ange hur bilder ska visas från början. Alternativ är:

- ScaleIcon.FIT Bilden fyller hela bildytan i komponenten
- ScaleIcon.NORMAL Bilden visas i naturlig storlek
- ScaleIcon.SCALE Bilden skalas om proportionerligt och visas så stor som möjligt.

Fotovisaren ska användas för att skapa ett bildspelsprogram, ett program som automatiskt skiftar mellan bilder som lagras i ett Album-objekt.

Första versionen av bildspelet, ***Bildspel1***, tar emot ett Album-objekt och visningstid (hur lång tid varje bild ska visas) vid konstruktion, skapar en Fotovisare och sätter slutligen igång en tråd vilken skiftar bilder med angivet mellanrum.

Instruktioner

Skapa klassen ***Bildspel1***. Följande instansvariabler ska finnas i klassen:

```
private Album album;            // Innehåller Foto-objekt som ska visas
private Thread thread;          // Skifta bilder
private Fotovisare visare;      // Visa bilder
private long delay;             // Visningstid av bild
private int bildNr = -1;        // Bild som ska visas.
```

Sedan är det dags att skriva konstruktorn. Varje rad är kommenterad:

```
public Bildspel1( Album inAlbum, long inDelay ) {
    album = inAlbum;                    // Lagra Album-objektet i variabeln album
    delay = inDelay;                    // Lagra visningstid av bild i delay
    visare = new Fotovisare( ScaleIcon.SCALE );         // Skapa Fotovisare
    // Hämta Foto-objekt ur album (foto nr 0) och visa det i Fotovisaren
    visare.setFoto( album.hämtaFoto( bildNr ) );
    visare.setVisible( true );          // göra Fotovisaren synlig
}
```

Nu kan du testa din klass med nedanstående program. Du måste ändra bildnamnen så att de refererar till bilder som finns på din dator.

Testprogram

```
Album album = new Album();
Foto f1 = new Foto("På badstranden", 2003, Foto.FAMILJ,"C:/bilder/IMGP3366.JPG");
Foto f2 = new Foto("Utsikt", 2002, Foto.ÖVRIGT, "C:/bilder/host.jpg");
Foto f3 = new Foto("Greklandskarta", 2005, Foto.RESOR, "C:/bilder/grekland.gif");
Foto f4 = new Foto("Sommarfötter", 2002, Foto.BARNBARN, "C:/bilder/fötter.jpg");
Foto f5 = new Foto("Glassdags", 2003, Foto.FAMILJ, "C:/bilder/glass.jpg");
album.läggTill( f1 );
album.läggTill( f2 );
album.läggTill( f3 );
album.läggTill( f4 );
album.läggTill( f5 );
Bildspel1 prog = new Bildspel1( album, 3000 );
```

Som du ser visas första bilden som lagras i Album-objektet.

Tråd för bildspel

Nu är det dags att skapa en tråd som skiftar bilden som visas. Två saker ska du göra

1. Klassen *Bildspel1* ska implementera interfacet *Runnable*. Detta innebär att du ska lägga till *implements Runnable* i klassens deklaration:

   ```
   public class Bildspel1 implements Runnable {
   ```

   Dessutom måste du skriva run-metoden som kommer att utgöra själva tråden:

   ```
   public void run() {
       while( true ) {                  // exekvera tills programmet avbryts
           try {
               Thread.sleep( delay );                        // Visningstid för bild
               bildNr = ( bildNr + 1 ) % album.antalFoto();  // Nr på nästa bild
               visare.setFoto( album.hämtaFoto( bildNr ) );  // Hämta + visa bild
           } catch( InterruptedException e ) {}
       }
   }
   ```

2. Du måste skapa ett Thread-objekt som exekverar run-metoden i Bildspel1. Tråden måste också startas.

   Lägg till följande två rader sist i konstruktorn:

   ```
   thread = new Thread( this ); // Skapa Thread-objekt, använd run-metoden i denna klassen
   thread.start(); // Starta tråden
   ```

Nu kan du köra testprogrammet ovan på nytt. Nu bör bildspelet fungera.

Uppgift 3 – Bildspel2 med Timer

Du ska skriva programmet ***Bildspel2***, vilket ska fungera på samma sätt som ***Bildspel1***. Skillnaden är att det nya programmet ska använda sig av ett objekt av typen *Timer* (javax.swing.Timer) i stället för ett *Thread*-objekt.

Börja med att skapa klassen *Bildspel2*. Kopiera sedan allt från *Bildspel1* till *Bildspel2*. Ganska mycket av koden kommer att återanvändas. Se till att ändra alla *Bildspel1* till *Bildspel2* efter kopieringen.

Det är inte så stora ändringar som behövs:

- *Runnable* ska ändras till *ActionListener* (import krävs)
- *run*-metoden ska ändras till *actionPerformed*
- Klassen ska ha en *Timer* som instansvariabel (import krävs)
- *Timer*-objektet ska skapas och sedan startas i konstruktorn
- *actionPerformed*-metoden ska endast ändra *index* och visa ny bild
- Ta bort instansvariablerna *thread* och *delay* och en del instruktioner (rödmarkerade)

Uppgift 4 – Textvisning med Thread

Du ska skriva ett program, ***Uppgift4***, som fungerar på samma sätt som programmet *Uppgift1*. Skillnaden är att *Uppgift4* ska använda sig av ett *Thread*-objekt i stället för ett *Timer*-objekt.

Börja med att skapa klassen *Uppgift4* och kopiera sedan innehållet i *Uppgift1* till *Uppgift4*. Glöm inte ändra alla Uppgift1 till Uppgift4.

Sedan är det dags att göra vissa ändringar i klassen:

- *ActionListener* ändras till *Runnable*
- *actionPerformed* ändras till *run*
- i *run*-metoden måste det vara en loop som itererar genom listan texter. Det måste också vara en paus på 3 sekunder (*Thread.sleep*) mellan anropen till *setText*-metoden.
- Instansvariabeln av typen *Timer* ska ersättas med en av typen *Thread*. Även tråden måste startas i konstruktorn.
- Ta bort onödiga importer

Testkör programmet genom att skapa ett objekt av typen *Uppgift4*.

```
Uppgift4 u4 = new Uppgift4();
```

Fördjupande uppgifter

Uppgift 5 – Bildspel3

Det vore bra att kunna starta och stoppa bildspelet. Du ska i denna uppgift lägga till ett fönster med två knappar, ***Bildspel3Input***, och skriva klassen ***Bildspel3*** vilken innehåller funktionalitet för att starta och stoppa bildspelet.

Bildspel3Input, del 1

Börja med att konstruera ett fönster med knappar för att starta / stoppa bildspelet. Du kan kalla klassen för *Bildspel3Input*.

Programmet ska avslutas om fönstret stängs. Fönstret ska bli synligt genom anrop till *setVisible* i konstruktorn.

Från början ska Starta bildspel-knappen vara aktiv och Stoppa bildspel-knappen vara dimmad.

Klassen ska innehålla instansvariabeln

```
private Bildspel3 controller;
```

Instansvariabeln ska tilldelas värde via konstruktorn. Det innebär att konstruktorn ska se ut så här (förutom din kod som designar fönster, knappar mm):

```
public Bildspel3Input( Bildspel3 controller) {
    this.controller = controller;
    // övrig kod
}
```

Nu återstår endast att något ska hända när man klicka på en av knapparna. Vi återkommer till detta när klassen *Bildspel3* är skriven.

Bildspel3

Börja med att skapa klassen ***Bildspel3***. Du ska kopiera innehållet i *Bildspel1* till *Bildspel3*.

Följande ändringar måste göras:

1. Lägg till instansvariabeln input och skapa ett fönster:
   ```
   private Bildspel3Input input = new Bildspel3Input( this );
   ```

2. Ta bort två rader i konstruktorn:
```
thread = new Thread( this );
thread.start();
```

3. Se till att tråden i run-metoden kan avbrytas. Detta styrs bl.a. av värde i referensen Thread. Så här ska run-metoden se ut (ändringarna / tilläggen är fet-markerade):

```
public void run() {
    while( thread != null ) {
        try {
            Thread.sleep( delay );
            bildNr = ( bildNr + 1 ) % album.antalFoto();
            visare.setFoto( album.hämtaFoto( bildNr ) );
        } catch( InterruptedException e ) {
            thread = null;
        }
    }
}
```

4. Lägga till metod för att starta tråden:
```
public void startaBildspel() {
    if( thread == null ) {    // om thread inte refererar till tråd så
        thread = new Thread( this );      // skapa tråd
        thread.start();       // starta tråd
    }
}
```

5. Lägga till metod för att stoppa tråden:
```
public void stoppaBildspel() {
    if( thread != null ) {    // om tråd refererar till tråd så
        thread.interrupt();   // avbryt tråd (vid sleep -> InterruptedException)
        thread = null;        // sätt thread till null (ej sleep)
    }
}
```

Bildspel3Input, del 2

Nu är det dags att se till att metoderna *startaBildspel* respektive *stoppaBildspel* anropas när användaren klickar på respektive knapp. De ska dessutom fungera så att knappen som man klickar på ska dimmas medan den andra ska aktiveras.

1. Lägg till en händelsehanterare för Starta bildspel-knappen och skriv följande kod i actionPerformed-metoden (i exemplet är knapparnas namn btnStartaBildspel och btnStoppaBildspel:

```
btnStartaBildspel.setEnabled( false );   // dimma starta-knapp
btnStoppaBildspel.setEnabled( true );    // aktivera stoppa-knapp
controller.startaBildspel();  // anrop av starta-metod
```

2. Lägg till en händelsehanterare för Stoppa bildspel-knappen och skriv följande kod i actionPerformed-metoden:

```
btnStoppaBildspel.setEnabled( false );   // dimma stoppa-knapp
btnStartaBildspel.setEnabled( true );    // aktivera starta-knapp
controller.stoppaBildspel();  // anrop av stoppa-metod
```

Nu kan du testa den nya versionen av bildspelsprogram med samma testprogram som för *Bildspel1*. Men glöm inte ändra sista raden till:

```
Bildspel3 prog = new Bildspel3( album, 3000 );
```

Uppgift 6 – Bildspel4

Det bör ju gå att starta och stoppa ett bildspel som det i ***Bildspel2***. Skapa klasserna ***Bildspel4Input*** och ***Bildspel4*** vilka ska ge samma körresultat som Uppgift 5 men där bildspelet sköts av ett *Timer*-objekt.

Extrauppgifter

Uppgift 7 – Animering genom bildväxling

Om man växlar bilder ganska ofta så kan man åstadkomma en animering. I denna uppgift ska du skriva en klass, ***Animation***, vilken *ärver JLabel*. En *JLabel*-komponent är ju bra för att visa en bild och det ska vi utnyttja.

- De objekt som ska visas ska lagras i en *Icon*-array (*Icon[]*).
- Bilderna ska bytas med vissa intevall. Dessa intevall ska lagras i en *long*-array(*long[]*). Olika bilder kan alltså visas under olika lång tid. Det behöver dock inte vara lika många värden i denna array som i *Icon*-arrayen. Man ska t.ex. kunna ange en array med endast ett värde och då ska detta värde användas varje gång.
- Klassen ska ha en tråd vilken pausar och skiftar bild upprepade gånger. Händelsetråden ska ju inte pausas.
- Man ska kunna starta animationen och stoppa animationen.
- Som en utbyggnad av lösningen ska vi göra animationen trådsäker. Swing-komponenterna är inte trådsäkra och därför ska de endast modifieras av händelsetråden.

Ett skal till klassen är givet, komplettera med kod:

```
public class Animation extends JLabel implements Runnable {
    private Icon[] iconList;
    private long[] delay;
    private Thread thread;
    private int iconIndex = 0;
    private int delayIndex = 0;

    public Animation( Icon[] iconList, long[] delay ) {
        this.iconList = iconList;
        this.delay = delay;
        setIcon(iconList[iconIndex]);
    }

    public void startAnimation() {
        // starta tråd
    }

    public void stopAnimation() {
        // stoppa tråd
    }

    public void run() {
        // Medan tråden är aktiv
        //    pausa tråden delay[delayIndex] millisekunder
        //    räkna upp iconIndex med 1 (0, 1, 2, …, iconIndex-1, 0, 1, …)
        //    räkna upp delayindex med 1 (0, 1, 2, …, delayIndex-1, 0, 1, …}
        //    visa bild i position iconIndex
    }
}
```

Du kan testa klassen *Animation* med nedanstående program. I programmet används bilderna new1.jpg, new2.jpg,.., new10.jpg. Du finner bilderna i ***LabB.zip***. Placera dem i lämplig mapp och ändra i programmet.

```
JFrame frame = new JFrame("AnimationEx");
Icon[] imageIcons = {new ImageIcon("C:/bilder/new1.jpg"),
                     new ImageIcon("C:/bilder/new2.jpg"),
                     new ImageIcon("C:/bilder/new3.jpg"),
                     new ImageIcon("C:/bilder/new4.jpg"),
                     new ImageIcon("C:/bilder/new5.jpg"),
                     new ImageIcon("C:/bilder/new6.jpg"),
                     new ImageIcon("C:/bilder/new7.jpg"),
                     new ImageIcon("C:/bilder/new8.jpg"),
                     new ImageIcon("C:/bilder/new9.jpg"),
                     new ImageIcon("C:/bilder/new10.jpg")};
long[] delay1 = {100,90,80,70,60,50,60,70,80,90}; // testa även med {50}
Animation anim1 = new Animation(imageIcons,delay1);
Container c = frame.getContentPane();
c.setLayout(null);
frame.setSize(140,170);
frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
anim1.setBounds(10,10,49,128);
c.add(anim1);
frame.setVisible(true);
anim1.startAnimation();
```

Det går naturligtvis bra att ha flera *Animation*-objekt i samma fönster. Om ovanstående program fungerar på ett vettigt sätt så kan du köra nedanstående program.

```
JFrame frame = new JFrame("AnimationEx");
Icon[] imageIcons = {new ImageIcon("C:/bilder/new1.jpg"),
                     new ImageIcon("C:/bilder/new2.jpg"),
                     new ImageIcon("C:/bilder/new3.jpg"),
                     new ImageIcon("C:/bilder/new4.jpg"),
                     new ImageIcon("C:/bilder/new5.jpg"),
                     new ImageIcon("C:/bilder/new6.jpg"),
                     new ImageIcon("C:/bilder/new7.jpg"),
                     new ImageIcon("C:/bilder/new8.jpg"),
                     new ImageIcon("C:/bilder/new9.jpg"),
                     new ImageIcon("C:/bilder/new10.jpg")};
long[] delay1 = {100,90,80,70,60,50,60,70,80,90};
long[] delay2 = {50};
Animation anim1 = new Animation(imageIcons,delay1);
Animation anim2 = new Animation(imageIcons,delay2);
Container c = frame.getContentPane();
c.setLayout(null);
frame.setSize(140,170);
frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
anim1.setBounds(10,10,49,128);
anim2.setBounds(70,10,49,128);
c.add(anim1);
c.add(anim2);
frame.setVisible(true);
anim1.startAnimation();
anim2.startAnimation();
try {
    Thread.sleep(10000);
    anim1.stopAnimation();
    Thread.sleep(5000);
    anim2.stopAnimation();
    Thread.sleep(2000);
    anim1.startAnimation();
    Thread.sleep(2000);
    anim2.startAnimation();
}catch(InterruptedException e) {}
```

Nu återstår det att se till att det är händelsetråden som uppdaterar *JLabel*-komponenten, dvs anropar *setIcon*-metoden. (Det är nämligen inte helt lyckat om din tråd och händelsetråden av misstag är inne i *setIcon*-metoden samtidigt. Det kan bli programkrasch). Det innebär att du

ska ersätta anropet till *setIcon* med ett anrop till metoden *SwingUtilities.invokeLater(Runnable)*. *run*-metoden i denna *Runnable*-implementering ska anropa *setIcon*-metoden. Vad som händer är att *invokeLater* placerar *Runnable*-objektet i en buffert och händelsetråden exekverar *run*-metoden i det buffrade objektet så fort det finns tid.

1. Implementera *Runnable* i en *inre klass* som ser ut så här:
```
private class Animate implements Runnable {
    public void run() {
        setIcon(iconList[iconIndex]);
    }
}
```
2. Lägg till en instansvariabel i *Animation*:
```
private Animate animate = new Animate();
```
3. Sedan byter du ut det ursprungliga anropet till setIcon med:
```
SwingUtilities.invokeLater(animate);
```

Testa din nya version av *Animation* med de tidigare programmen.

Lösningar

Uppgift 1

```
package labB;
import java.util.*;
import javax.swing.Timer;
import java.awt.*;
import java.awt.event.*;

public class Uppgift1 implements ActionListener {
    private ArrayList<Text> texter = new ArrayList<Text>();
    private TextViewer viewer = new TextViewer();
    private Timer timer = new Timer(3000, this);
    int index = 0;

    public Uppgift1() {
        String s1 = "Att lära utan att tänka är meningslöst, att tänka utan att lära är farligt";
        String s2 = "Den, som tror sig veta allt, vet ingenting ";
        String s3 = "Hellre olärd och klok, än lärd och dum ";
        String s4 = "Sanna ord är inte alltid vackra, vackra ord är inte alltid sanna ";
        texter.add(new Text(s1, Color.blue, Color.yellow, new Font("Serif", Font.BOLD, 14)));
        texter.add(new Text(s2, Color.white, Color.darkGray, new Font("SansSerif", Font.PLAIN, 24)));
        texter.add(new Text(s3, Color.yellow, Color.darkGray, new Font("Monospaced", Font.ITALIC, 28)));
        texter.add(new Text(s4,, Color.red, Color.blue, new Font("Dialog", Font.BOLD + Font.ITALIC, 20)));
        timer.start();
    }

    public void actionPerformed(ActionEvent e) {
        if( index < texter.size() ) {
            viewer.setText(texter.get(index));
            index++;
        } else {
            timer.stop();
        }
    }

    public static void main(String[] args) {
        new Uppgift1();
    }
}
```

Bildspel1

```
package labB;
import labA.*; // Foto, Album

public class Bildspel1 implements Runnable {
    private Album album;
    private Thread thread;
    private Fotovisare visare;
    private long delay;
    private int bildNr = 0;

    public Bildspel1( Album inAlbum, long inDelay ) {
        album = inAlbum;
        delay = inDelay;
        visare = new Fotovisare( ScaleIcon.SCALE );
        visare.setFoto( album.hämtaFoto( bildNr ) );
        visare.setVisible( true );
        thread = new Thread( this );
        thread.start();
    }

    public void run() {
        while( true ) {
            try {
                Thread.sleep( delay );
                bildNr = ( bildNr + 1 ) % album.antalFoto();
```

```
                visare.setFoto( album.hämtaFoto( bildNr ) );
            } catch( InterruptedException e ) {}
        }
    }
}
```

Bildspel2

```
package labB;
import labA.*; // Foto, Album
import javax.swing.*; // Ny
import java.awt.event.*;

public class Bildspel2 implements ActionListener {
    private Album album;
    private Timer timer; // Ny
    private Fotovisare visare;
    private int bildNr = 0;

    public Bildspel2( Album inAlbum, int inDelay ) {
        album = inAlbum;
        visare = new Fotovisare( ScaleIcon.SCALE );
        visare.setFoto( album.hämtaFoto( bildNr ) );
        visare.setVisible( true );
        timer = new Timer(inDelay, this); // ny
        timer.start(); // ny
    }

    public void actionPerformed(ActionEvent e) {
        bildNr = ( bildNr + 1 ) % album.antalFoto();
        visare.setFoto( album.hämtaFoto( bildNr ) );
    }
}
```

Uppgift4

```
package labB;
import java.util.*;
import java.awt.*;

public class Uppgift4 implements Runnable {
    private ArrayList<Text> texter = new ArrayList<Text>();
    private TextViewer viewer = new TextViewer();
    private Thread thread = new Thread(this); // Timer bort

    public Uppgift4() {
        String s1 = "Att lära utan att tänka är meningslöst, att tänka utan att lära är farligt";
        String s2 = "Den, som tror sig veta allt, vet ingenting ";
        String s3 = "Hellre olärd och klok, än lärd och dum ";
        String s4 = "Sanna ord är inte alltid vackra, vackra ord är inte alltid sanna ";
        texter.add(new Text(s1, Color.blue, Color.yellow, new Font("Serif", Font.BOLD, 14)));
        texter.add(new Text(s2, Color.white, Color.darkGray, new Font("SansSerif", Font.PLAIN, 24)));
        texter.add(new Text(s3, Color.yellow, Color.darkGray, new Font("Monospaced", Font.ITALIC, 28)));
        texter.add(new Text(s4,, Color.red, Color.blue, new Font("Dialog", Font.BOLD + Font.ITALIC, 20)));
        thread.start();
    }

    public void run() {
        int index = 0;
        while(index < texter.size()) {
            try {
                Thread.sleep(3000);
                viewer.setText(texter.get(index));
                index++;
            }catch(InterruptedException e) {}
        }
    }
```

```
    public static void main(String[] args) {
        new Uppgift4();
    }
}
```

Bildspel3

```
package labB;
import labA.Album;
import labA.Foto;

public class Bildspel3 implements Runnable {
    private Bildspel3Input input = new Bildspel3Input(this);
    private Album album;
    private Thread thread;
    private Fotovisare visare;
    private long delay;
    private int bildNr = 0;

    public Bildspel3( Album inAlbum, long inDelay ) {
        album = inAlbum;
        delay = inDelay;
        visare = new Fotovisare( ScaleIcon.SCALE );
        visare.setFoto( album.hämtaFoto( bildNr ) );
        visare.setVisible( true );
    }

    public void startaBildspel() {
        if( thread == null ) {
            thread = new Thread( this );
            thread.start();
        }
    }

    public void stoppaBildspel() {
        if( thread != null ) {
            thread.interrupt();
            thread = null;
        }
    }

    public void run() {
        while( thread != null ) {
            try {
                Thread.sleep( delay );
                bildNr = ( bildNr + 1 ) % album.antalFoto();
                visare.setFoto( album.hämtaFoto( bildNr ) );
            } catch( InterruptedException e ) {
                thread = null;
            }
        }
    }
}
```

Bildspel3Input

```
package labB;
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

public class Bildspel3Input {
    private Bildspel3 controller;
    private JFrame frame = new JFrame("Bildspel3Input");
    private JButton btnStartaBildspel = new JButton("Starta bildspel");
```

```
    private JButton btnStoppaBildspel = new JButton("Stoppa bildspel");

    public Bildspel3Input(Bildspel3 controller) {
        this.controller = controller;
        Container c = frame.getContentPane();
        c.setLayout(null);
        frame.setSize(205,89);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setResizable(false);
        btnStartaBildspel.setBounds(8,10,180,20);
        btnStoppaBildspel.setBounds(8,35,180,20);
        btnStoppaBildspel.setEnabled( false );
        btnStartaBildspel.addActionListener(new Starta());
        btnStoppaBildspel.addActionListener(new Stoppa());
        c.add(btnStartaBildspel);
        c.add(btnStoppaBildspel);
        frame.setVisible(true);
    }

    private class Starta implements ActionListener {
        public void actionPerformed(ActionEvent e) {
            btnStartaBildspel.setEnabled( false );
            btnStoppaBildspel.setEnabled( true );
            controller.startaBildspel();
        }
    }
    private class Stoppa implements ActionListener {
        public void actionPerformed(ActionEvent e) {
            btnStoppaBildspel.setEnabled( false );
            btnStartaBildspel.setEnabled( true );
            controller.stoppaBildspel();
        }
    }
}
```

Bildspel4

```
package labB;
import java.awt.event.*;
import javax.swing.Timer;
import labA.Album;
import labA.Foto;

public class Bildspel4  implements ActionListener {
    private Bildspel4Input input = new Bildspel4Input(this);
    private Album album;
    private Timer timer;
    private Fotovisare visare;
    private int bildNr = 0;

    public Bildspel4( Album inAlbum, int inDelay ) {
        album = inAlbum;
        visare = new Fotovisare( ScaleIcon.SCALE );
        visare.setFoto( album.hämtaFoto( bildNr ) );
        visare.setVisible( true );
        timer = new Timer(inDelay, this);
    }

    public void startaBildspel() {
        timer.start();
    }

    public void stoppaBildspel() {
        timer.stop();
    }
```

```
    public void actionPerformed(ActionEvent e) {
        bildNr = ( bildNr + 1 ) % album.antalFoto();
        visare.setFoto( album.hämtaFoto( bildNr ) );
    }
}
```

Bildspel4Input

```
package labB;
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

public class Bildspel4Input {
    private Bildspel4 controller;
    private JFrame frame = new JFrame("Bildspel3Input");
    private JButton btnStartaBildspel = new JButton("Starta bildspel");
    private JButton btnStoppaBildspel = new JButton("Stoppa bildspel");

    public Bildspel4Input(Bildspel4 controller) {
        this.controller = controller;
        Container c = frame.getContentPane();
        c.setLayout(null);
        frame.setSize(205,89);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setResizable(false);
        btnStartaBildspel.setBounds(8,10,180,20);
        btnStoppaBildspel.setBounds(8,35,180,20);
        btnStoppaBildspel.setEnabled( false );
        btnStartaBildspel.addActionListener(new Starta());
        btnStoppaBildspel.addActionListener(new Stoppa());
        c.add(btnStartaBildspel);
        c.add(btnStoppaBildspel);
        frame.setVisible(true);
    }

    private class Starta implements ActionListener {
        public void actionPerformed(ActionEvent e) {
            btnStartaBildspel.setEnabled( false );
            btnStoppaBildspel.setEnabled( true );
            controller.startaBildspel();
        }
    }
    private class Stoppa implements ActionListener {
        public void actionPerformed(ActionEvent e) {
            btnStoppaBildspel.setEnabled( false );
            btnStartaBildspel.setEnabled( true );
            controller.stoppaBildspel();
        }
    }
}
```

Animation

```
package labB;
import java.awt.*;
import javax.swing.*;

public class Animation extends JLabel implements Runnable {
    private Icon[] iconList;
    private long[] delay;
    private Thread thread;
//    private Animate animate = new Animate();
    private int iconIndex = 0;
    private int delayIndex = 0;
```

```
    public Animation( Icon[] iconList, long[] delay ) {
        this.iconList = iconList;
        this.delay = delay;
        setIcon(iconList[iconIndex]);
    }

    public void startAnimation() {
        if(thread==null) {
            thread = new Thread(this);
            thread.start();
        }
    }

    public void stopAnimation() {
        if(thread!=null) {
            thread.interrupt();
            thread = null;
        }
    }

    public void run() {
        while( thread != null ) {
            try {
                Thread.sleep(delay[delayIndex]); // Tråden ska vänta
                delayIndex = (delayIndex+1)%delay.length;
                iconIndex = (iconIndex+1)%iconList.length;
                setIcon(iconList[iconIndex]); // Avmarkera I trådsäker lösning
//              SwingUtilities.invokeLater(animate); // Grafikstråden uppdaterar
            }catch(InterruptedException e) {
                thread = null;
            }
        }
    }

//    private class Animate implements Runnable {
//        public void run() {
//            setIcon(iconList[iconIndex]);
//        }
//    }
}
```