Laboration 4 – Timers och trådar

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

Uppgift 1 – Använda java.util.Timer

Du ska skriva klassen **Exercise1** vilken ska använda en **Timer**, nämligen *java.util.Timer*. När timern startats så ska ett antal strängar skrivas ut i Output-fönstret (System.out.println...) med en sekunds mellanrum.

Vid konstruktion ska ett argument bifogas:

- en String-array som innehåller ett antal strängar som ska skrivas ut

Testkod

```
String[] strings = {"Måndag", "Tisdag", "Onsdag", "Torsdag", "Fredag",
                    "Lördag", "Söndag"};
Exercise1 ex1 = new Exercise1(strings);
```

Körresultat (tar ca 6 sekunder)

```
Måndag
Tisdag
Onsdag
Torsdag
Fredag
Lördag
Söndag
```

Testkod

```
String[] strings1 = {"Jag heter Rut", "Du heter Sven", "Han heter Ola"};
String[] strings2 = {"Måndag", "Tisdag", "Onsdag", "Torsdag", "Fredag",
                    "Lördag", "Söndag"};
Exercise1 ex1a = new Exercise1(strings1);
Exercise1 ex1b = new Exercise1(strings2);
```

Exempel på körresultat (tar ca 6 sekunder)

```
Jag heter Rut
Måndag
Du heter Sven
Tisdag
Onsdag
Han heter Ola
Torsdag
Fredag
Lördag
Söndag
```

Lägg gärna till en konstruktor i klassen

```
public Exercise1(String[] strings, long pause) {...}
```

och kör ovanstående program med olika långa pauser.

Uppgift 2a – Ärva Thread

Du ska skriva klassen **Exercise2a** vilken ska **ärva klassen Thread**. När tråden körs så ska ett antal strängar skrivas ut i Output-fönstret (System.out.println...) med en sekunds mellanrum.

Vid konstruktion ska ett argument bifogas:

- en String-array som innehåller ett antal strängar som ska skrivas ut

Testkod

```
String[] strings = {"Måndag", "Tisdag", "Onsdag", "Torsdag", "Fredag",
                    "Lördag", "Söndag"};
Exercise2a ex2a = new Exercise2a(strings);
ex2a.start();
```

Körresultat (tar ca 6 sekunder)

```
Måndag
Tisdag
Onsdag
Torsdag
Fredag
Lördag
Söndag
```

Testkod

```
String[] strings1 = {"Jag heter Rut", "Du heter Sven", "Han heter Ola"};
String[] strings2 = {"Måndag", "Tisdag", "Onsdag", "Torsdag", "Fredag",
                     "Lördag", "Söndag"};
Exercise2a ex2a = new Exercise2a(strings1);
Exercise2a ex2b = new Exercise2a(strings2);
ex2a.start();
ex2b.start();
```

Exempel på körresultat (tar ca 6 sekunder)

```
Jag heter Rut
Måndag
Du heter Sven
Tisdag
Onsdag
Han heter Ola
Torsdag
Fredag
Lördag
Söndag
```

Lägg gärna till en konstruktor i klassen

```
    public Exercise2a(String[] strings, long pause) {...}
```

och kör ovanstående program med olika långa pauser.

Uppgift 2b – Implementera Runnable

Denna uppgift är samma som Uppgift 2a men i denna uppgiften ska klassen **Exeercise2b implementera Runnable**. Sedan används en referens till ett objekt av klassen som argument till Thread-konstruktor.

Testkod

```
String[] strings = {"Måndag", "Tisdag", "Onsdag", "Torsdag", "Fredag",
                    "Lördag", "Söndag"};
Exercise2b u2b = new Exercise2b(strings); // implements Runnable
Thread t2 = new Thread(u2b);
t2.start();
```

Testkod

```
String[] strings1 = {"Jag heter Rut", "Du heter Sven", "Han heter Ola"};
String[] strings2 = {"Måndag", "Tisdag", "Onsdag", "Torsdag", "Fredag",
                     "Lördag", "Söndag"};
Thread t2a = new Thread(new Exercise2b(strings1));
Thread t2b = new Thread(new Exercise2b(strings2));
t2a.start();
t2b.start();
```

Uppgift 2c – Inre klass ärver Thread

Denna uppgift är samma som uppgift 2a men i denna uppgiften ska du låte **en inre klass ärva Thread**. Klassen ska vara inre klass i klassen **Exercise2c**. Du måste lägga till metoden

```
    public void start() {…}
```

för att starta tråden.

Testkod

```
String[] strings = {"Måndag", "Tisdag", "Onsdag", "Torsdag", "Fredag",
                    "Lördag", "Söndag"};
Exercise2c ex2 = new Exercise2c(strings);
ex2.start(); // anrop av metoden start() i klassen Exercise2c
```

Testkod

```
String[] strings1 = {"Jag heter Rut", "Du heter Sven", "Han heter Ola"};
String[] strings2 = {"Måndag", "Tisdag", "Onsdag", "Torsdag", "Fredag",
                     "Lördag", "Söndag"};
Exercise2c ex2a = new Exercise2c(strings1);
Exercise2c ex2b = new Exercise2c(strings2);
ex2a.start();
ex2b.start();
```

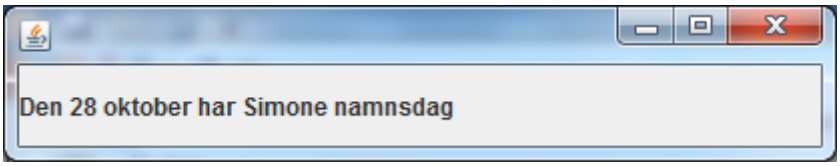
Uppgift 3 – Göra utskrifterna i JLabel

Klassen **Exercise3** ska fungera som Uppgift 1 men med ett par skillnader:

- konstruktorn ska ta emot en *JLabel* i vilken meddelandena ska skrivas och *intervallet* mellan utskrifterna
- när alla texterna visats en gång så ska texterna visas från början igen.

Testkod

```
public static void main(String[] args) {
    SwingUtilities.invokeLater(new Runnable() {
        public void run() {
            JFrame frame = new JFrame();
            JLabel lblText = new JLabel();
            String[] texter = {"Det är två månader kvar på året",
                               "23 * 6735 = 154905",
                               "Den 28 oktober har Simone namnsdag"};
            lblText.setPreferredSize(new Dimension(400,40));
            frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
            frame.add(lblText);
            frame.pack();
            frame.setVisible(true);
            Exercise3 ex3 = new Exercise3(texter, lblText, 3000);
            ex3.start();
        }
    });
}
```



Uppgift 4a – Start/stopp

Ändra dina lösning i klassen *Exercise2c* så att det går att stoppa utskrifterna och sedan på nytt starta dem. Din lösning kan heta *Exercise4a*.

Ändringen innebär att du metoderna *start* och *stop* ska vara implementerade i klassen på ett vettigt sätt. (*start* redan implementerad på ett sätt – duger det?)

Testkod

```
public static void main(String[] args) {
    String[] strings1 = {"Jag heter Rut", "Du heter Sven", "Han heter Ola"};
    String[] strings2 = {"Måndag", "Tisdag", "Onsdag", "Torsdag", "Fredag",
                         "Lördag", "Söndag"};
    Exercise4a ex4a = new Exercise4a(strings1);
    Exercise4a ex4b = new Exercise4a(strings2);
    ex4a.start();
    ex4b.start();
    try {
        Thread.sleep(2000);
    } catch(InterruptedException e) {}
    ex4a.stop();
    try {
        Thread.sleep(2000);
    } catch(InterruptedException e) {}
    ex4a.start();
    ex4b.stop();
    try {
        Thread.sleep(2000);
    } catch(InterruptedException e) {}
    ex4b.start();
}
```

Exempel på körresultat (tar ca 6 sekunder)

```
Jag heter Rut
Måndag
Du heter Sven
Tisdag
Onsdag
Torsdag
Han heter Ola
Fredag
Lördag
Söndag
```

Uppgift 4b – Start/stopp

Ändra *TwoIcon2*-klassen från föreläsningen så det går att stoppa/starta bildväxlingen. Det innebär att klassen ska implementera interfacet StartStopListener.
Du kan testa funktionaliteten med *TIController*.

Uppgift 4c – Start/stopp

Ändra *TwoIcon4*-klassen från föreläsningen så det går att stoppa/starta bildväxlingen.

Lösningar

Uppgift 1

```
import java.util.Timer;
import java.util.TimerTask;

public class Exercise1 {
    private String[] strings;
    private long pause;
    private Timer timer;

    public Exercise1(String[] strings) {
        this(strings, 1000);
    }

    public Exercise1(String[] strings, long pause) {
        this.strings = strings.clone();
        this.pause = pause;
        timer = new Timer();
        timer.schedule(new ToDo(), 0, 1000);
    }

    private class ToDo extends TimerTask {
        private int index=0;

        public void run() {
            if( index < strings.length ) {
                System.out.println(strings[index]);
                index++;
            } else {
                timer.cancel();
            }
        }
    }
}
```

Uppgift 2a

```
public class Exercise2a extends Thread {
    private String[] strings;
    private long pause;
    private int index=0;

    public Exercise2a(String[] strings) {
        this(strings, 1000);
    }

    public Exercise2a(String[] strings, long pause) {
        this.strings = strings.clone();
        this.pause = pause;
    }

    public void run() {
        while( index < strings.length ) {
            System.out.println(strings[index]);
            index++;
            try {
                Thread.sleep(pause);
            } catch (InterruptedException e) {}
        }
    }
}
```

Uppgift 2b

```
public class Exercise2b implements Runnable {
    private String[] strings;
    private long pause;
    private int index=0;
```

```
    public Exercise2b(String[] strings) {
        this(strings, 1000);
    }

    public Exercise2b(String[] strings, long pause) {
        this.strings = strings.clone();
        this.pause = pause;
    }

    public void run() {
        while( index < strings.length ) {
            System.out.println(strings[index]);
            index++;
            try {
                Thread.sleep(pause);
            } catch (InterruptedException e) {}
        }
    }
}
```

Uppgift 2c

```
public class Exercise2c {
    private Activity thread;
    private String[] strings;
    private long pause;

    public Exercise2c(String[] strings) {
        this(strings, 1000);
    }

    public Exercise2c(String[] strings, long pause) {
        this.strings = strings.clone();
        this.pause = pause;
    }

    public void start() {
        if( thread == null ) {
            thread = new Activity();
            thread.start();
        }
    }

    private class Activity extends Thread {
        private int index=0;

        public void run() {
            while( index < strings.length ) {
                System.out.println(strings[index]);
                index++;
                try {
                    Thread.sleep(pause);
                } catch (InterruptedException e) {}
            }
        }
    }
}
```

Uppgift 3

```
public class Exercise3 extends Thread {
    private String[] messages;
    private JLabel lblText;
    private long pause;
    private boolean argumentsOK;
```

```
    private int index = 0;

    public Exercise3(String[] messages, JLabel label, long pause) {
        this.messages = messages;
        this.lblText = label;
        this.pause = pause;
        argumentsOK = (messages!=null) && (messages.length>0) &&
                      (label!=null) && (pause>=0);
    }

    public void run() {
        while( argumentsOK && !interrupted()) {
            // UI-tråden ska användas
            SwingUtilities.invokeLater( new Write(messages[index]) );
            index = (index + 1) % messages.length;
            try {
                Thread.sleep(pause);
            } catch(InterruptedException e) {
                break;
            }
        }
    }

    private class Write implements Runnable {
        private String message;

        public Write(String message) {
            this.message = message;
        }

        public void run() {
            lblText.setText(message);
        }
    }

}
```

Uppgift 4a

```
public class Exercise4a {
    private Activity thread;
    private String[] strings;
    private long pause;
    private int index=0;

    public Exercise4a(String[] strings) {
        this(strings, 1000);
    }

    public Exercise4a(String[] strings, long pause) {
        this.strings = strings.clone();
        this.pause = pause;
    }

    public void start() {
        if( thread == null ) {
            thread = new Activity();
            thread.start();
        }
    }

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

```
    private class Activity extends Thread {
        public void run() {
            while( index < strings.length && !Thread.interrupted() ) {
                System.out.println(strings[index]);
                index++;
                try {
                    Thread.sleep(pause);
                } catch (InterruptedException e) {
                        break;
                }
            }
            thread=null;
        }
    }
}
```

Uppgift 4b

```
public class TwoIcon2 extends TwoIconLabel implements StartStopListener {
    private java.util.Timer timer;
    private int delay;

    public TwoIcon2(Icon icon1, Icon icon2) {
        this(icon1, icon2, 500);
    }

    public TwoIcon2(Icon icon1, Icon icon2, int delay) {
        this(icon1, icon2, 200, 200, delay);
    }

    public TwoIcon2(Icon icon1, Icon icon2, int width, int height, int delay) {
       super(icon1, icon2, width, height);
       this.delay = delay;
       ChangeIcon cc = new ChangeIcon();
    }

    public void start() {
       if(timer==null) {
            timer = new java.util.Timer();
            timer.schedule(new ToDo(),delay,delay);
       }
    }

    public void stop() {
       if(timer!=null) {
          timer.cancel();
          timer = null;
       }
    }

    private class ToDo extends TimerTask {
       private ChangeIcon ci = new ChangeIcon();

       public void run() {
          SwingUtilities.invokeLater(ci);
       }
    }

    private class ChangeIcon implements Runnable {
        public void run() {
           changeIcon();
        }
    }
}
```

Uppgift 4c

```
public class TwoIcon4 extends TwoIconLabel implements Runnable, StartStopListener {
    private long delay;
    private Thread thread;
```

```
    private Runnable changeIcon = new ChangeIcon();

    public TwoIcon4(Icon icon1, Icon icon2) {
        this(icon1, icon2, 500);
    }

    public TwoIcon4(Icon icon1, Icon icon2, int delay) {
        this(icon1, icon2, 200, 200, delay);
    }

    public TwoIcon4(Icon icon1, Icon icon2, int width, int height, int delay) {
        super(icon1, icon2, width, height);
        this.delay = delay;
    }

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

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

    public void run() {
        while(!Thread.interrupted()) {
            try {
                Thread.sleep(delay);
            } catch(InterruptedException e) {
                break;
            }
            SwingUtilities.invokeLater( changeIcon );
        }
        thread = null;
    }

    private class ChangeIcon implements Runnable {
        public void run() {
            changeIcon();
        }
    }
}
```