ЛУ-10

1. Създайте програма, използваща свързан списък с 100 елемента. Всеки елемент да е дума. Подредете списъка и реализирайте добавяне и изтриване на елемент.

2. Анимиране на топче.

3. Напишете програма, която ползва методи от Runnable и Thread. Засечете time execution и използвайте JProfiler

Задача 2:

- Ще използваме отделни нишки за да управляваме движението на топчето по екрана.
- Необходими са следните файлове:
- ***anim.html***

```
<HTML>
<BODY>
<APPLET CODE="Animation.class" WIDTH=300 HEIGHT=400>
</APPLET>
</BODY>
```

- ***Animation.java***
- ```
  import java.awt.*;
  import java.awt.event.*;
  import javax.swing.*;
  ```
- ```
  public class Animation extends JApplet implements Runnable, ActionListener {
     int miFrameNumber = -1;
     int miTimeStep;
     Thread mAnimationThread;
     boolean mbIsPaused = false;
     Button mButton;
     Point mCenter;
     int miRadius;
     int miDX, miDY;
  ```
- ```
     public void init() {
        // Make the animation run at 20 frames per second.  We do this by
        // setting the timestep to 50ms.
        miTimeStep = 50;
  ```
- ```
        // Initialize the parameters of the circle.
        mCenter = new Point(getSize().width/2, getSize().height/2);
        miRadius = 15;
        miDX = 4;  // X offset per timestep.
        miDY = 3;  // Y offset per timestep.
  ```
- ```
        // Create a button to start and stop the animation.
        mButton = new Button("Stop");
        getContentPane().add(mButton, "North");
        mButton.addActionListener(this);
  ```
- ```
        // Create a JPanel subclass and add it to the JApplet.  All drawing
        // will be done here, do we must write the paintComponent() method.
        // Note that the anonymous class has access to the private data of
        // class Animation, because it is defined locally.
        getContentPane().add(new JPanel() {
           public void paintComponent(Graphics g) {
              // Paint the background.
              super.paintComponent(g);
  ```

```
        // Display the frame number.
            g.drawString("Frame " + miFrameNumber, getSize().width/2 - 40,
                        getSize().height - 15);

            // Draw the circle.
            g.setColor(Color.red);
            g.fillOval(mCenter.x-miRadius, mCenter.y-miRadius, 2*miRadius,
                    2*miRadius);
        }
    }, "Center");
}

public void start() {
    if (mbIsPaused) {
        // Don't do anything.  The animation has been paused.
    } else {
        // Start animating.
        if (mAnimationThread == null) {
            mAnimationThread = new Thread(this);
        }
        mAnimationThread.start();
    }
}

public void stop() {
    // Stop the animating thread by setting the mAnimationThread variable
    // to null.  This will cause the thread to break out of the while loop,
    // so that the run() method terminates naturally.
    mAnimationThread = null;
}

public void actionPerformed(ActionEvent e) {
    if (mbIsPaused) {
        mbIsPaused = false;
        mButton.setLabel("Stop");
        start();
    } else {
        mbIsPaused = true;
        mButton.setLabel("Start");
        stop();
    }
}
```

```
public void run() {
        // Just to be nice, lower this thread's priority so it can't
        // interfere with other processing going on.
        Thread.currentThread().setPriority(Thread.MIN_PRIORITY);

        // Remember the starting time.
        long startTime = System.currentTimeMillis();

        // Remember which thread we are.
        Thread currentThread = Thread.currentThread();

        // This is the animation loop.
        while (currentThread == mAnimationThread) {
            // Advance the animation frame.
            miFrameNumber++;

            // Update the position of the circle.
            move();

            // Draw the next frame.
            repaint();

            // Delay depending on how far we are behind.
            try {
                startTime += miTimeStep;
                Thread.sleep(Math.max(0,
                        startTime-System.currentTimeMillis()));
            }
            catch (InterruptedException e) {
                break;
            }
        }
    }
```

```
// Update the position of the circle.
  void move() {
      mCenter.x += miDX;
      if (mCenter.x - miRadius < 0 ||
          mCenter.x + miRadius > getSize().width) {
          miDX = -miDX;
          mCenter.x += 2*miDX;
      }
      mCenter.y += miDY;
      if (mCenter.y - miRadius < 0 ||
          mCenter.y + miRadius > getSize().height) {
          miDY = -miDY;
          mCenter.y += 2*miDY;
      }
  }
}
```

Sr.No.	Method & Description
1	**public void start()** Starts the thread in a separate path of execution, then invokes the run() method on this Thread object.
2	**public void run()** If this Thread object was instantiated using a separate Runnable target, the run() method is invoked on that Runnable object.
3	**public final void setName(String name)** Changes the name of the Thread object. There is also a getName() method for retrieving the name.
4	**public final void setPriority(int priority)** Sets the priority of this Thread object. The possible values are between 1 and 10.
5	**public final void setDaemon(boolean on)** A parameter of true denotes this Thread as a daemon thread.
6	**public final void join(long millisec)** The current thread invokes this method on a second thread, causing the current thread to block until the second thread terminates or the specified number of milliseconds passes.
7	**public void interrupt()** Interrupts this thread, causing it to continue execution if it was blocked for any reason.
8	**public final boolean isAlive()** Returns true if the thread is alive, which is any time after the thread has been started but before it runs to completion.

```java
// File Name : DisplayMessage.java
// Create a thread to implement Runnable

public class DisplayMessage implements Runnable {
   private String message;

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

   public void run() {
      while(true) {
         System.out.println(message);
      }
   }
}
```

```java
// File Name : GuessANumber.java
// Create a thread to extentd Thread

public class GuessANumber extends Thread {
   private int number;
   public GuessANumber(int number) {
      this.number = number;
   }

   public void run() {
      int counter = 0;
      int guess = 0;
      do {
         guess = (int) (Math.random() * 100 + 1);
         System.out.println(this.getName() + " guesses " + guess);
         counter++;
      } while(guess != number);
      System.out.println("** Correct!" + this.getName() + "in" + counter + "guesses.**");
   }
}
```

```java
// File Name : ThreadClassDemo.java
public class ThreadClassDemo {

   public static void main(String [] args) {
      Runnable hello = new DisplayMessage("Hello");
      Thread thread1 = new Thread(hello);
      thread1.setDaemon(true);
      thread1.setName("hello");
      System.out.println("Starting hello thread...");
      thread1.start();

      Runnable bye = new DisplayMessage("Goodbye");
      Thread thread2 = new Thread(bye);
      thread2.setPriority(Thread.MIN_PRIORITY);
      thread2.setDaemon(true);
      System.out.println("Starting goodbye thread...");
      thread2.start();

      System.out.println("Starting thread3...");
      Thread thread3 = new GuessANumber(27);
      thread3.start();
      try {
         thread3.join();
      } catch (InterruptedException e) {
         System.out.println("Thread interrupted.");
      }
      System.out.println("Starting thread4...");
      Thread thread4 = new GuessANumber(75);

      thread4.start();
      System.out.println("main() is ending...");
   }
}
```

Задача 3:

```
class Runner1 extends Thread {

    @Override
    public void run(){
        for(int i=0;i<100;++i){
            System.out.println("Runner1: "+i);
            try {
                Thread.sleep(100);
            }catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
}
class Runner2 extends Thread {

    @Override
    public void run() {
        for (int i = 0; i < 100; ++i) {
            System.out.println("Runner2: " + i);
            try {
                Thread.sleep(100);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
}
}
        }

    }
}
public class App {
    public static void main(String[] args) {

   Worker worker = new Worker();
   Thread t1 = new Thread(worker);
   t1.start();
    }
}
```

Добавете класа Worker

```
class Worker implements Runnable {
    private boolean isTerminated = false;

    public boolean isTerminated() {
        return isTerminated;
    }

    public void setTerminated(boolean terminated) {
        isTerminated = terminated;
    }

    @Override
    public void run() {
        while (!isTerminated) {
            System.out.println("Hello from worker class ...");
            try {
                Thread.sleep(300);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }

    }
}
public class App {
    public static void main(String[] args) {

   Worker worker = new Worker();
   Thread t1 = new Thread(worker);
   t1.start();
    }
}
```