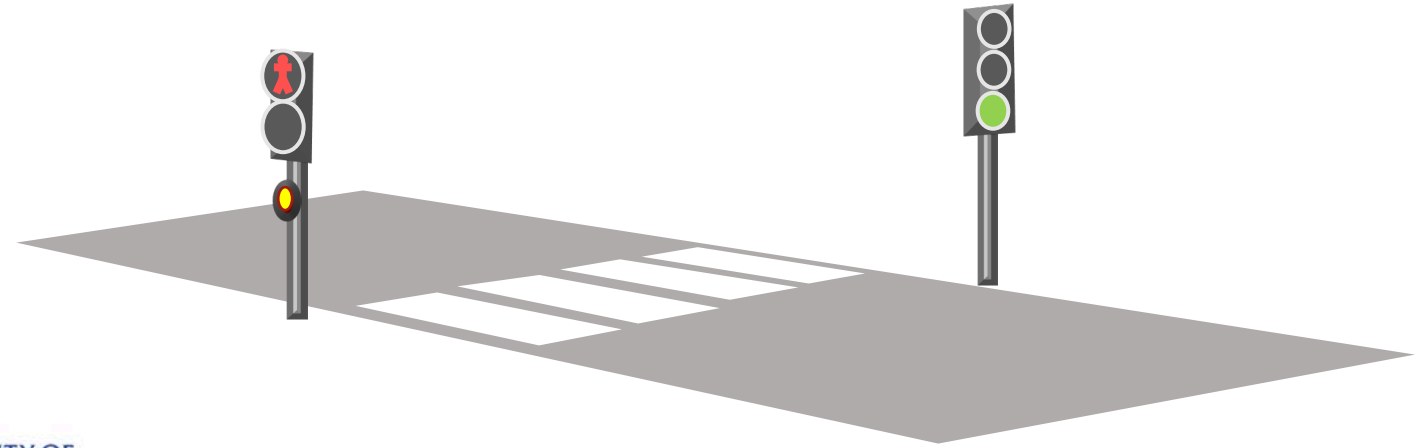
1495 UNIVERSITY OF ABERDEEN

JC2002 Java Programming

Lecture 29: Timed events and synchronisation

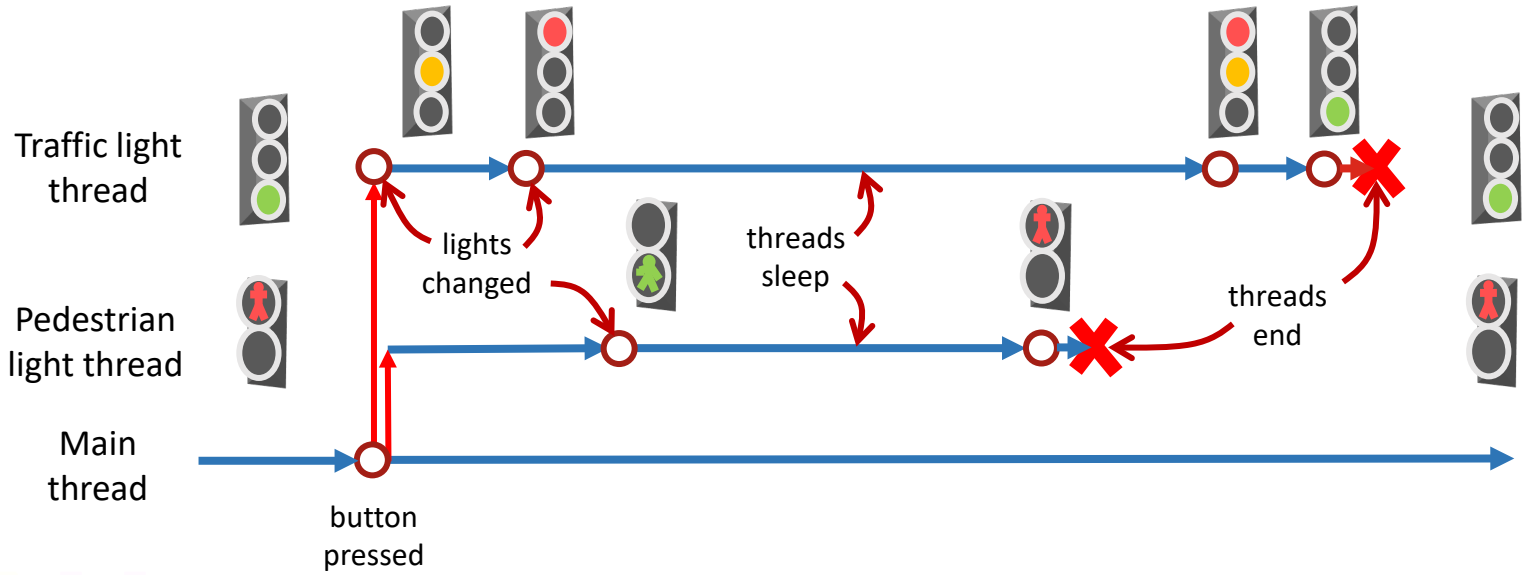
Multithreading for timed events

- Threads can be useful for implementing timed events
- Example: traffic lights, where pedestrian pushes a button to request green light and to initiate the light cycle



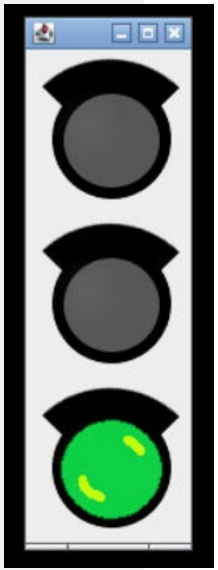
Example: traffic lights

- We could implement pedestrian and traffic light cycles as two threads



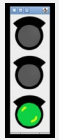
Example: define lights for cars (1)

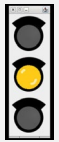
```
5   class TrafficLights extends JPanel {
6
7       JLabel topLight, middleLight, bottomLight;
8       ImageIcon off, red, yellow, green;
9
10      TrafficLights() {
11          off = new ImageIcon("traffic_off.png");
12          red = new ImageIcon("traffic_red.png");
13          yellow = new ImageIcon("traffic_yellow.png");
14          green = new ImageIcon("traffic_green.png");
15          topLight = new JLabel();
16          topLight.setIcon(off);
17          middleLight = new JLabel();
18          middleLight.setIcon(off);
19          bottomLight = new JLabel();
20          bottomLight.setIcon(green);
21          GridLayout gridlayout = new GridLayout(3,1);
22          setLayout(gridlayout);
23          add(topLight);
24          add(middleLight);
25          add(bottomLight);
26      }
```



Example: define lights for cars (2)

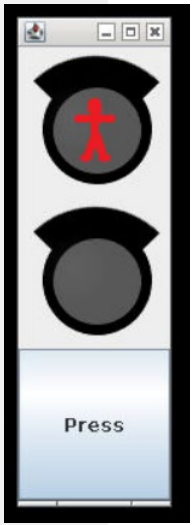
```
27      public void setGreen() {
28          topLight.setIcon(off);
29          middleLight.setIcon(off);
30          bottomLight.setIcon(green);
31      }
32      public void setYellow() {
33          topLight.setIcon(off);
34          middleLight.setIcon(yellow);
35          bottomLight.setIcon(off);
36      }
37      public void setRed() {
38          topLight.setIcon(red);
39          middleLight.setIcon(off);
40          bottomLight.setIcon(off);
41      }
42      public void setRedAndYellow() {
43          topLight.setIcon(red);
44          middleLight.setIcon(yellow);
45          bottomLight.setIcon(off);
46      }
47  }
```





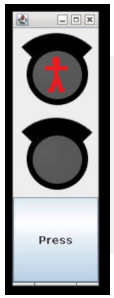
Example: define lights for pedestrians (1)

```
49  class PedestrianLights extends JPanel
50                          implements ActionListener {
51      JLabel topLight, bottomLight;
52      JButton button;
53      ImageIcon off, wait, go;
54      String status;
55      TrafficLights trafficLights;
56      PedestrianLights(TrafficLights tl) {
57          trafficLights = tl;
58          status = new String("wait");
59          off = new ImageIcon("traffic_off.png");
60          wait = new ImageIcon("traffic_wait.png");
61          go = new ImageIcon("traffic_go.png");
62          topLight = new JLabel();
63          topLight.setIcon(wait);
64          bottomLight = new JLabel();
65          bottomLight.setIcon(off);
66          button = new JButton("Press");
67          GridLayout gridlayout = new GridLayout(3,1);
68          setLayout(gridlayout);
69          add(topLight);
70          add(bottomLight);
71          add(button);
```



Example: define lights for pedestrians (2)

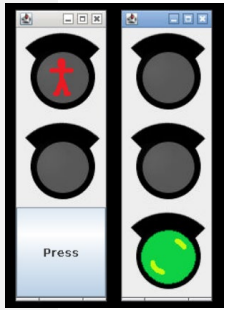
```
72          button.addActionListener(this);
73      }
74      public void setGo() {
75          topLight.setIcon(off);
76          bottomLight.setIcon(go);
77          status = new String("go");
78      }
```




```
79      public void setWait() {
80          topLight.setIcon(wait);
81          bottomLight.setIcon(off);
82          status = new String("wait");
83      }
```

Example: create and show GUI

```
129 public class TrafficLightExample {
130   private static void createAndShowGUI() {
131     JComponent trafficLights = new TrafficLights();
132     trafficLights.setOpaque(true);
133     JComponent pedestrianLights = new PedestrianLights((TrafficLights)trafficLights);
134     pedestrianLights.setOpaque(true);
135     JFrame plFrame = new JFrame("Pedestrian Lights");
136     plFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
137     JFrame tlFrame = new JFrame("Traffic Lights");
138     tlFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
139     plFrame.add(pedestrianLights);
140     plFrame.pack();
141     plFrame.setLocation(50,50);
142     plFrame.setVisible(true);
143     tlFrame.add(trafficLights);
144     tlFrame.pack();
145     tlFrame.setLocation(300,50);
146     tlFrame.setVisible(true);
147   }
```




```
149   public static void main(String[] args) {
150     javax.swing.SwingUtilities.invokeLater(new Runnable() {
151       public void run() {
152         createAndShowGUI();
153       }
154     });
155   }
156 }
```

Example: run cycle for cars

```
49  class PedestrianLights extends JPanel
50                         implements ActionListener {
...
...
84      public void startCycle() {
85          Thread trafficThread = new Thread(new Runnable() {
86              @Override
87              public void run()
88              {
89                  try {
90                      trafficLights.setYellow();
91                      Thread.sleep(2000);
92                      trafficLights.setRed();
93                      Thread.sleep(5000);
94                      trafficLights.setRedAndYellow();
95                      Thread.sleep(1000);
96                      trafficLights.setGreen();
97                      button.setEnabled(true);
98                  }
99                  catch (InterruptedException e) {
100                     e.printStackTrace();
101                 }
102             }
103         });
```

Example: run cycle for pedestrians

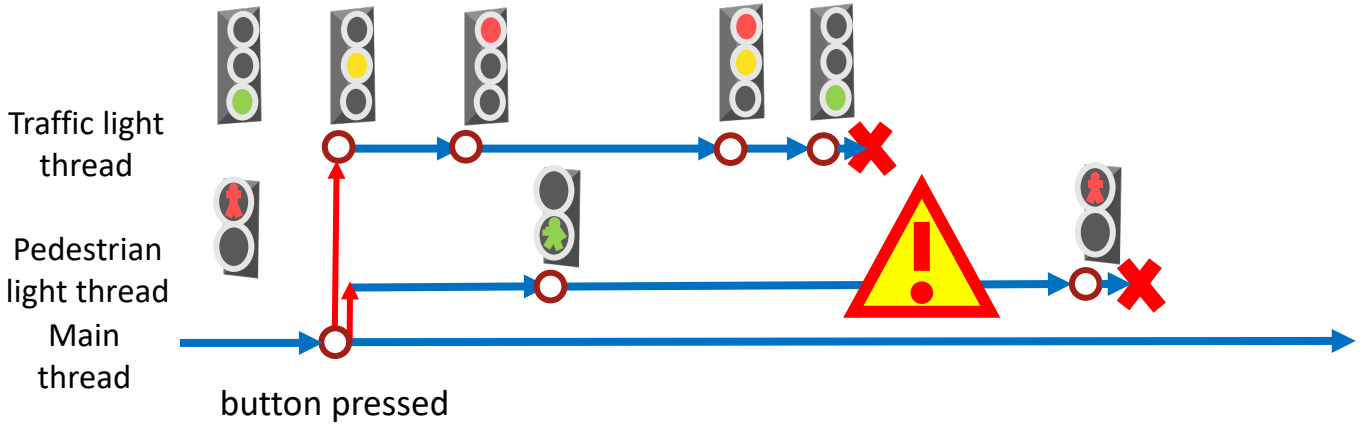
```
104     Thread pedestrianThread = new Thread(new Runnable() {
105       @Override
106       public void run()
107       {
108         try {
109           button.setEnabled(false);
110           Thread.sleep(3000);
111           setGo();
112           Thread.sleep(3000);
113           setWait();
114         }
115         catch (InterruptedException e) {
116           e.printStackTrace();
117         }
118       }
119     });
120     trafficThread.start();
121     pedestrianThread.start();
122   }
```

Button is disabled to avoid new cycle to start before the old has stopped


```
124     public void actionPerformed(ActionEvent e) {
125       startCycle();
126     }
127   }
```

Synchronisation in traffic light example

- When traffic light threads run independently and timings are wrong, there is a risk that the light for cars turns green too early
- We should make sure the lights are not in conflict with each other

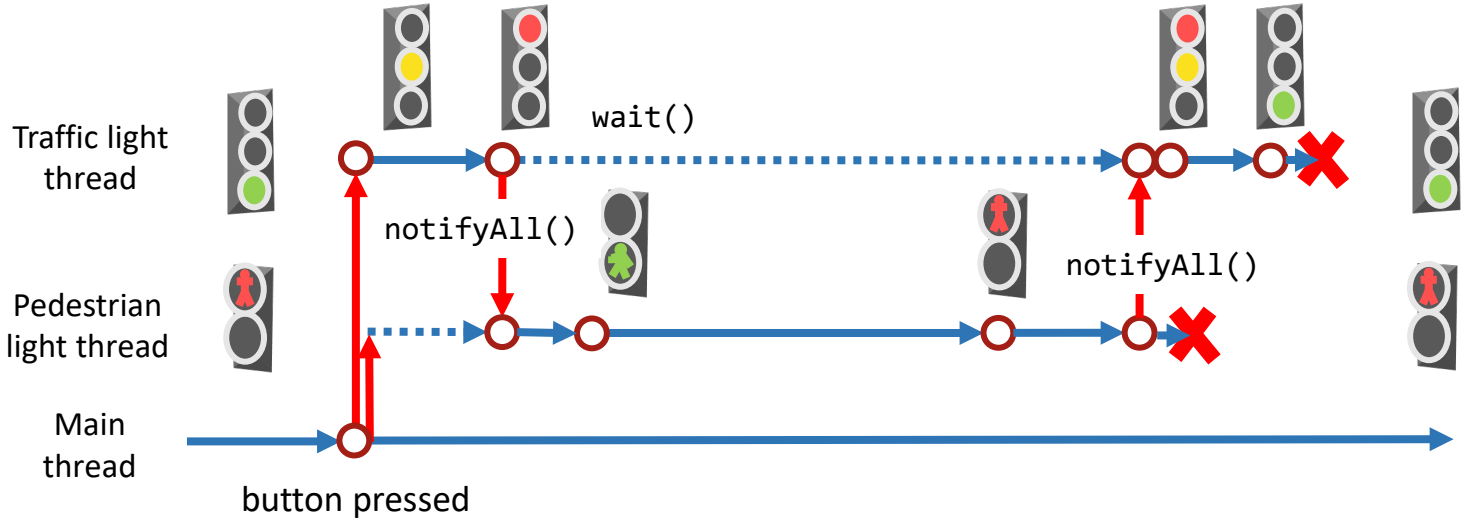


Using wait() and notify()

- Access to any object of class or subclass of `Object` (basically any object) from a thread can be controlled by **`Object.wait()`** and **`Object.notifyAll()`**
  - Method `wait()` suspends the current thread until another thread issues a notification for the same object so that it can resume
  - Method `notifyAll()` sends a notification to all the other threads so that they can resume
  - Method `notify()` is similar to `notifyAll()`, but only notifies one thread chosen randomly

Example: synchronised traffic lights

- We could synchronize threads so that traffic lights for cars always waits until pedestrian lights have been turned red again



Traffic lights with synchronisation (1)

```
...
88      public void startCycle() {
89          TrafficLightThread tlCycle = new TrafficLightThread(this,tl);
90          PedestrianLightThread plCycle = new PedestrianLightThread(this);
91          tlCycle.start();
92          plCycle.start();
93      }
94
95      public void actionPerformed(ActionEvent e) {
96          disableButton();
97          startCycle();
98      }
99  }
```

Note that the threads are synchronized, and therefore `plCycle` starts in waiting state

We define custom subclasses of `Thread` for traffic lights and pedestrian lights

Traffic lights with synchronisation (2)

```
101 class TrafficLightThread extends Thread {
102   private PedestrianLights pl;
103   private TrafficLights tl;
104   public TrafficLightThread(PedestrianLights pl, TrafficLights tl) {
105     this.pl = pl;  this.tl = tl;
106   }
107   @Override
108   public void run() {
109     synchronized (pl) {
110       try {
111         tl.setYellow();
112         Thread.sleep(2000);
113         tl.setRed();
114         pl.notifyAll();
115         pl.wait();
116         tl.setRedAndYellow();
117         Thread.sleep(1000);
118         tl.setGreen();
119         pl.enableButton();
120       }
121       catch (InterruptedException e) {
122         e.printStackTrace();
123       }
124     }
125   }
126 }
```

Methods `notifyAll()` and `wait()` must be inside `synchronized` block to avoid exception

Traffic lights with synchronisation (3)

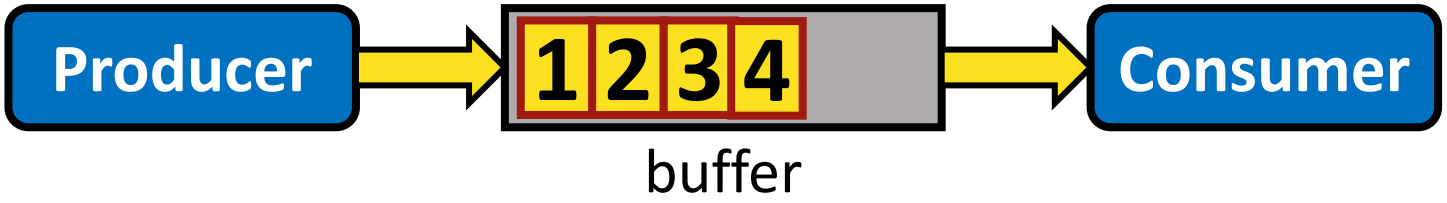
```
128 class PedestrianLightThread extends Thread {
129     private PedestrianLights pl;
130     public PedestrianLightThread(PedestrianLights pl) {
131         this.pl = pl;
132     }
133     @Override
134     public void run() {
135         synchronized (pl) {
136             try {
137                 Thread.sleep(3000);
138                 pl.setGo();
139                 Thread.sleep(5000);
140                 pl.setWait();
141                 Thread.sleep(3000);
142                 pl.notifyAll();
143             }
```

Note that this thread starts in waiting state, because it is synchronized and started after the other thread!

```
144         catch (InterruptedException e) {
145           e.printStackTrace();
146         }
147       }
148   }
149 }
```

Producer-consumer model

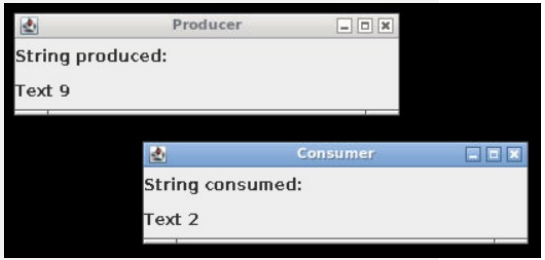
- A common example of the importance of synchronization is *producer-consumer problem*
  - *Producer* thread produces data and adds it in buffer
  - *Consumer* thread consumes data and removes it from buffer
  - Without synchronization, threads may try to add and remove data at the same time, leading to problems



Producer-consumer example: GUI

```
58  public class ProducerConsumerExample {
59    private static void createAndShowGUI() {
60      ArrayList<String> buffer = new ArrayList<>();
61      JPanel producerPanel = new JPanel();
62      producerPanel.setOpaque(true);
63      JLabel prodInfo = new JLabel("String produced:");
64      JLabel prodLabel = new JLabel();
...  ...
76      JPanel consumerPanel = new JPanel();
77      consumerPanel.setOpaque(true);
78      JLabel consInfo = new JLabel("String consumed:");
79      JLabel consLabel = new JLabel();
...  ...
91      Producer prodThread = new Producer(buffer, prodLabel);
92      prodThread.start();
93      Consumer consThread = new Consumer(buffer, consLabel);
94      consThread.start();
95    }
...  ...
```

Create buffer



Create and start producer and consumer threads

Producer thread

```
6   class Producer extends Thread {
7     private ArrayList<String> buffer;
8     private JLabel label;
9     public Producer(ArrayList<String> buffer, JLabel label) {
10      this.buffer = buffer;
11      this.label = label;
12    }
13    @Override
14    public void run() {
15      for(int i = 0; i < 100; i++) {
16        synchronized (buffer) {
17          String text = new String("Text “ + i);
18          System.out.println("Produced text: " + text);
19          buffer.add(text);
20          label.setText(text);
21        }
22        try {
23          Random r = new Random();
24          Thread.sleep(r.nextInt(800));
25        }
26        catch(InterruptedException e) {
26          e.printStackTrace();
27        }
28      }
29    }
30  }
```

Simulate random intervals between produced text items

Consumer thread

```
32  class Consumer extends Thread {
33    private ArrayList<String> buffer;
34    private JLabel label;
35    public Consumer(ArrayList<String> buffer, JLabel label) {
36      this.buffer = buffer;
37      this.label = label;
38    }
39    @Override
40    public void run() {
41      while(true) {
42        synchronized (buffer) {
43          if(!buffer.isEmpty()) {
44            String text = buffer.remove(0);
45            System.out.println("Consumed text: " + text);
46            label.setText(text);
47          }
48        }
49        try {
50          Thread.sleep(1000);
51        }
52        catch(InterruptedException e) {
53          e.printStackTrace();
54        }
55      }
56    }
57  }
```

```
$ java ProducerConsumerExample
Produced text: Text 0
Consumed text: Text 0
Produced text: Text 1
Produced text: Text 2
Produced text: Text 3
Produced text: Text 4
Consumed text: Text 1
Produced text: Text 5
Produced text: Text 6
Produced text: Text 7
Consumed text: Text 2
```

Simulate one second processing time for consumed items

Questions, comments?