Processes and Threads II

Lecture 3

Alma Rahat

CS-210: Concurrency

02 February, 2021

Questions and comments: shorturl.at/kCOW7

What did we do in the last session?

- Finite State Process (FSP) is a type of process algebra that helps us describe processes.
- Labelled Transition System (LTS) is a graphical version of the FSP, and allows us to examine the system interactions.
- LTSA tool can be used to effectively visualise a LTS.
- Design and implementation workflow:
  1. Deconstruct ✓
  2. Model ✓
  3. Implement ✗

In this lecture...

Learning outcomes.

1. To implement simple process in Java.
2. To explain and apply the modelling of choices in a process using FSP.

Outline.

1. Java implementation.
2. Incorporating choices in FSP.

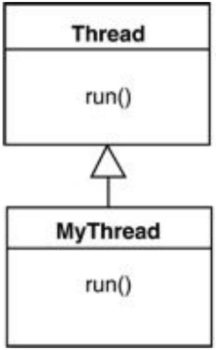
Threads in Java

- The `Thread` class is part of the `java.lang` package.
- An instance of the `Thread` class manages a single sequential thread of control.
- An instance may be created or deleted dynamically.
- An instance executes the content in `run` method once the `start` method is called.
- A `Thread` instance should be properly shut down through `InterruptedException`.
- `sleep` method causes a running thread to suspend for a specified period of time.

For complete documentation on the `Thread` class visit the following:
`docs.oracle.com/javase/8/docs/api/java/lang/Thread.html`

Two ways to define a thread

We can define a thread by inheriting from the `Thread` class, and overriding the `run` method.



```
class MyThread extends Thread {
    public void run() {
        //...
    }
}
```

Two ways to define a thread

Is extending a Thread class for defining behaviour a good idea? In a few words, tell us why or why not.

Please go to `www.menti.com` and enter the code 34 51 08 9.

Two ways to define a thread

Is extending a Thread class for defining behaviour a good idea? In a few words, tell us why or why not.

Please go to `www.menti.com` and enter the code 34 51 08 9.

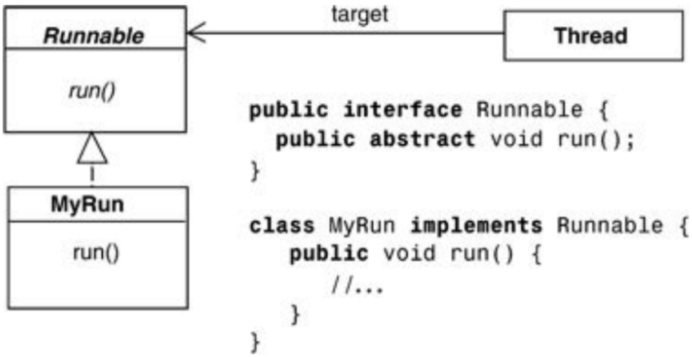
Not a good idea. We want:

- We ought to focus on behaviour implementation, rather than overriding. A client may perform a task using a thread behaviour, but is not necessarily a thread it self, and it does not need all the attributes of a thread.
- Java does not allow multiple inheritance – we may want to inherit from a more relevant class.

See the following for an interesting discussion:
`stackoverflow.com/questions/541487/implements-runnable-vs-extends-thread-in-java`

Two ways to define a thread

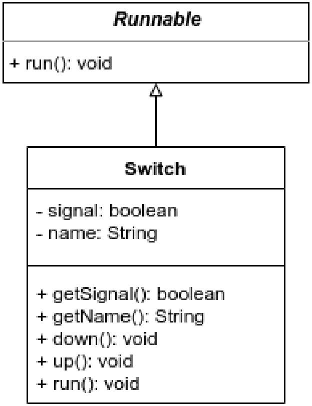
It is more robust to use `Runnable` interface for this.



```
public interface Runnable {
  public abstract void run();
}

class MyRun implements Runnable {
   public void run() {
       //...
   }
}
```

Step III: Implement



Please remember the main application thread should be separate from the worker thread.

Step III: Implement

Live demonstration

Git repository: `github.com/AlmaRahat/CS-210-Concurrency`

Any questions?

Choices in FSP

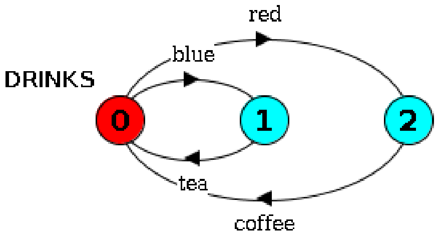
If x and y are actions then $(x- > P|y- > Q)$ describes a process which initially engages in either of the actions x or y. After the first action, the subsequent behaviour is described by P if x was the first action and Q is the first action was y instead.

A drink dispenser

Scenario.
A drink dispenser has two buttons: red and blue. If the red one is pressed it produces coffee, and if the blue button is pressed it produces tea.

FSP:

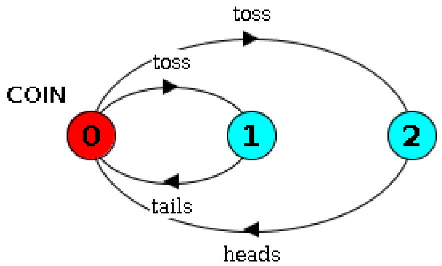
```
DRINKS = (red -> coffee -> DRINKS | blue -> tea ->
DRINKS).
```



Tossing a coin

Scenario.
If you toss a coin it either produces a head or tails.
FSP:

```
COIN = (toss -> heads -> COIN | toss -> tails -> COIN).
```

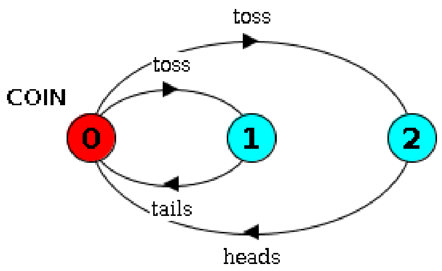


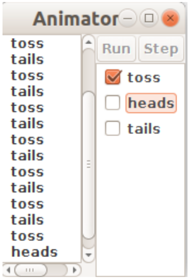
Same action, but different behaviour. Any issues?

Tossing a coin

Scenario.
If you toss a coin it either produces a head or tails.
FSP:

```
COIN = (toss -> heads -> COIN | toss -> tails -> COIN).
```





```
toss
tails
toss
tails
toss
tails
toss
tails
toss
tails
toss
tails
toss
heads
```

Same action, but different behaviour. Any issues? Gives rise to non-deterministic behaviour.

Exercise: Cruise Control

Consider a cruise control system: The engine of the car is initially off. When the engine is on, it the system will make the car reach and maintain a speed.

Step I: Deconstruct. What is an action in this scenario?

Please go to `www.menti.com` and enter the code 30 99 34 0.

Exercise: Cruise Control

Consider a cruise control system: The engine of the car is initially off. When the engine is on, it the system will make the car reach and maintain a speed.

Step I: Deconstruct. What is an action in this scenario?

Please go to `www.menti.com` and enter the code 30 99 34 0.

- engineOn
- engineOff
- speed

Exercise: Cruise Control

Consider a cruise control system: The engine of the car is initially off. When the engine is on, it the system will make the car reach and maintain a speed.

Step I: Deconstruct. what would be an appropriate intermediary state?

Please go to `www.menti.com` and enter the code 30 99 34 0.

Exercise: Cruise Control

Consider a cruise control system: The engine of the car is initially off. When the engine is on, it the system will make the car reach and maintain a speed.

Step I: Deconstruct. what would be an appropriate intermediary state?

Please go to `www.menti.com` and enter the code 30 99 34 0.

- Off
- CheckSpeed

Exercise: Cruise Control

Consider a cruise control system: The engine of the car is initially off. When the engine is on, it the system will make the car reach and maintain a speed.

Step II: Process alphabets, FSP and LTSA
Process: CruiseControl
Actions:

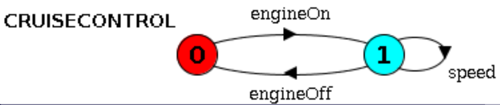
- engineOn
- engineOff
- speed

Sub-processes: Off and CheckSpeed.

Exercise: Cruise Control

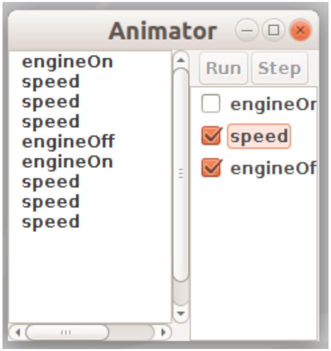
Consider a cruise control system: The engine of the car is initially off. When the engine is on, it the system will make the car reach and maintain a speed.

```
\\FSP for CRUISECONTROL
CRUISECONTROL = OFF,\\initial state is set to off.
OFF = (engineOn -> CHECKSPEED), \\when engine turns on it checks speed and progresses forward.
CHECKSPEED = (speed -> CHECKSPEED | engineOff -> OFF).
\\conditional change of states
```

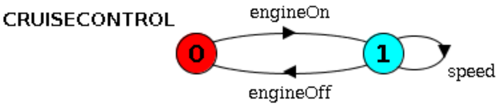


Do you see any issues?

Is this system safe? Let's have a look at the animator traces...



```
engineOn
speed
speed
speed
engineOff
engineOn
speed
speed
speed
```

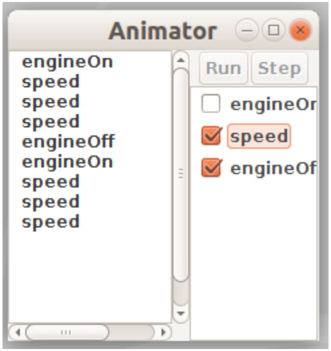


$$\begin{tikzcd}
0 \arrow[r, bend left, "engineOn"] & 1 \arrow[l, bend left, "engineOff"] \arrow[loop right, "speed"]
\end{tikzcd}$$

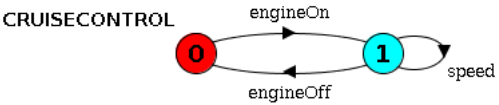
Please go to `www.menti.com` and use the code 72 75 87

Do you see any issues?

Is this system safe? Let's have a look at the animator traces...

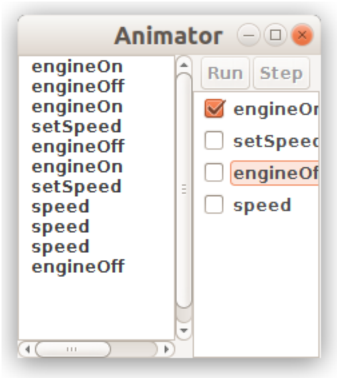

Animator

```
engineOn
speed
speed
speed
engineOff
engineOn
speed
speed
speed
```

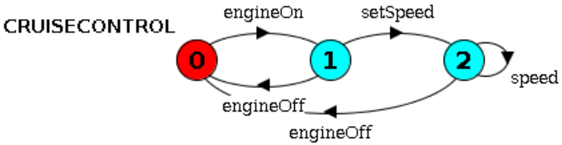


- After turning engine off and then on, we are still allowed to speed! This can cause a serious accident. The LTS clearly shows that.
- How could we solve this?

Do you see any issues?



```
engineOn
engineOff
engineOn
setSpeed
engineOff
engineOn
setSpeed
speed
speed
speed
engineOff
```



Potentially introduce a further state to ensure that action of setting speed is performed after turning on. Can you write down the FSP for this?

You can write on: `shorturl.at/kCOW7`

Do you see any issues?

Solution.

```
CRUISECONTROL = OFF,
OFF = (engineOn -> ON),
ON = (setSpeed -> CHECKSPEED | engineOff -> OFF),
CHECKSPEED = (speed -> CHECKSPEED | engineOff ->
OFF).
```

Any questions?

Summary

- For implementing choices we use the following syntax: `(x -> P | y-> Q)`.
- Animation or looking at traces can indicate potential issues with a system.