

We will wait 10 minutes until 10:40 AM for all students to join into the meeting.

We will start the tutorial at **10:40 AM**.

This tutorial will be recorded.

# CS 3SD3 - Concurrent Systems Tutorial 1

---

Mahdee Jodayree.

McMaster University

September 14, 2021

# Outline

---

- I. Introduction
- II. Concurrent programs
  - I. Overview (understand this course)
  - II. Implementation
- III. Concurrency with Java Programming
- IV. Install LTSA tool and run simple codes on it.
- V. Questions and answers

# Introduction

---

A little about us:

There are four TAs for this course.

My name is Mahdee Jodayree and I am a PhD Candidate in Computer Science. Email: [mahdijaf@yahoo.com](mailto:mahdijaf@yahoo.com)

Tony (Tianyuan Zhang) Email: [zhatiayua59@gmail.com](mailto:zhatiayua59@gmail.com)

Jatin Chowdhary Email: [chowdhaj@mcmaster.ca](mailto:chowdhaj@mcmaster.ca)

Weijie Liang Email: [liangw27@mcmaster.ca](mailto:liangw27@mcmaster.ca)

Feel free to email us about office hour, any question or concerns, I will response in 1 day.

# Introduction (continued)

---

Information on tutorials.

- ❖ 110 minutes in length.
- ❖ Recommend that you attend tutorials.
  - ❖ Last year result from tests and assignments.
- ❖ A presentation on lecture problems, FSP, Java programming, tools (LTSA, SVN, ...), Hints for Assignments and tests and etc.
- ❖ Tutorial slides will be posted on the course website after the tutorial.
- ❖ **Mainly we will show you codes and examples on LTSA tool.**
- ❖ We discuss important fundamental material from lecture notes.
  - ❖ we try to clarify any ambiguous material.
- ❖ Sometimes we discuss entire lecture note (not always).

# Tutorial format

---

Information on tutorials.

- ❖ It is **not** mandatory to attend the tutorials in person, everything is accessible online.
- ❖ First week tutorials are online.
- ❖ From second week
  - ❖ I will teach new material on Tuesdays.
  - ❖ Tuesday's tutorial will be recorded.
  - ❖ All students can attend the Tuesday's class.
  - ❖ All students can watch the recorded video of Tuesday's tutorial at any time.
  - ❖ On Wednesday and Friday, for the first hour students will watch the video of Tuesdays tutorial and after that ask their questions.

---

Any questions so far?

# Concurrent programs: Overview

---

- ❖ Terms: Concurrent and Parallel programs and concurrent programming
- ❖ There are no concrete definitions for these terms.
  - ❖ Each definition is explaining concurrency from different perspective.
  - ❖ Concurrency is when **two or more tasks** can start, run, and complete in overlapping time periods in no specific order.
  - ❖ **This significantly improve overall speed of the execution.**
  - ❖ For example, multitasking on a single-core machine.
    - ❖ On Windows, you can surf the internet while you listen to music.
- ❖ Another definition for concurrency is that, concurrency is the act of managing and running multiple computations at the same time.
  - ❖ Real life example, going to passport office and glossary at the same time.



# What is Parallelism

---

- Parallelism is the act of running multiple computations simultaneously.
- For example: When tasks literally run at the same time, e.g., on a multicore processor.
- Parallelism means that an application splits its tasks up into smaller subtasks which can be processed in parallel on multiple CPUs at the exact same time.
- Parallelism requires hardware.
- Example of that is computing (adding 1 to 100).

# Difference between Concurrency and Parallelism

Concurrency		Parallelism
Significantly improves overall speed of the execution <b>without additional hardware.</b>		Parallelism requires hardware.
Context switch	Uses	Multiple CPU's for operating multiple processes.
Illusion of parallelism	Meaning	Quality of occurring at the same time.
Single CPU	No of processes needed	Multiple CPUS.

# What is a concurrent programming.

---

- ❖ This courses is all about concurrent programing.
- ❖ we must review what a **process** and a **thread** is.
- ❖ A **process** is the instance of a computer program that is being executed by one or many threads. It contains the program code and its activity it also includes the program counter, process stack, registers, and program code
  - ❖ For example, a Java code is a thread but when executed, a Java code becomes a process.
  - ❖ A process may be made up of multiple threads of execution that execute instructions concurrently.
    - ❖ For example Chrome or Firefox browser.
  - ❖ A **thread** is execution of the smallest sequence of programmed instructions that can be managed **independently by a scheduler**, which is typically a part of the operating system.

# Processes vs. Threads

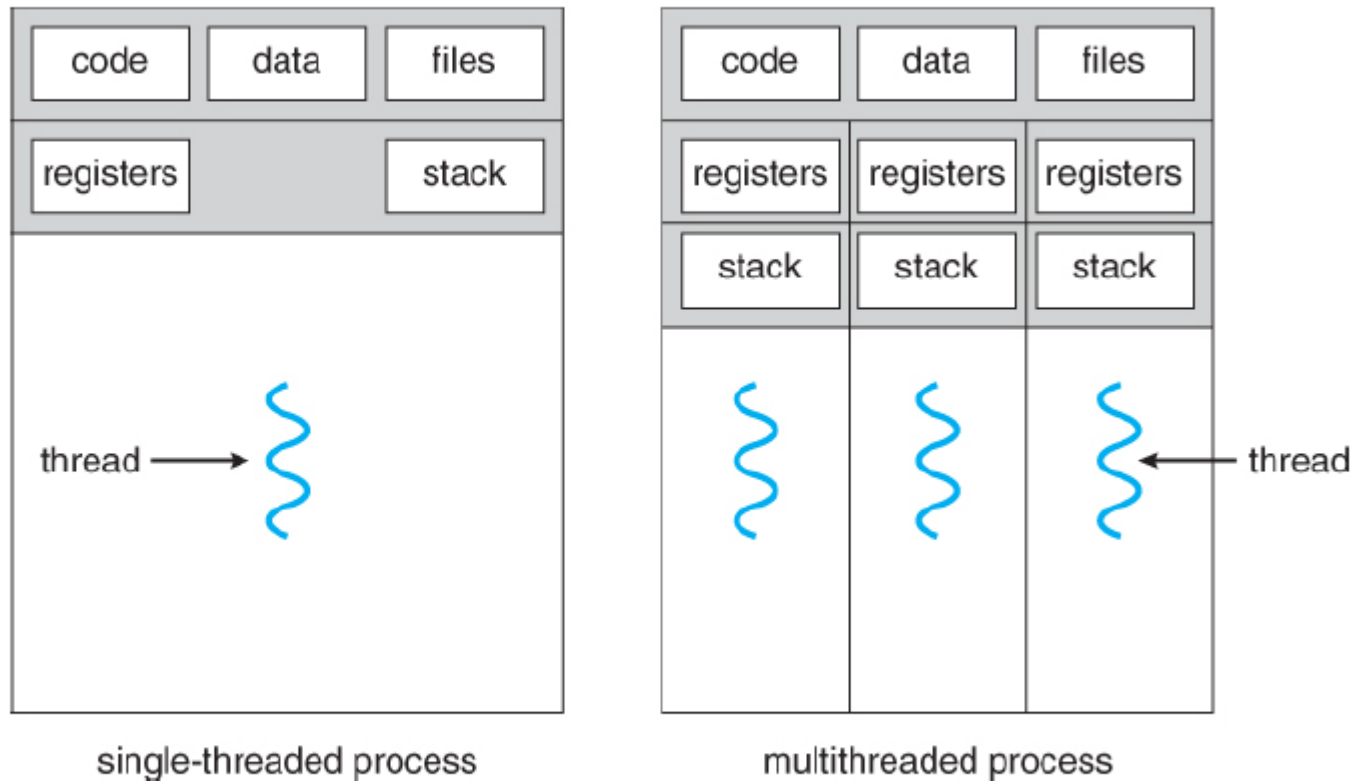


Figure: Process vs. Thread

# More examples of multithreading

---

- ❖ An Operating System

- ❖ You can move your cursor while you do anything else

- ❖ Writing an e-mail

- ❖ While an attachment is loading, you can still write your draft

# What will happen if we run this code with multithreading code?

---

```
class Counter {  
    private int c = 0;  
  
    public void incrementone() {  
        c++;  
    }  
  
    public void incrementBytwo() {  
        c=c+2;    }  
  
    public int value() {  
        return c;    }  
}
```

# Threads - Java

---

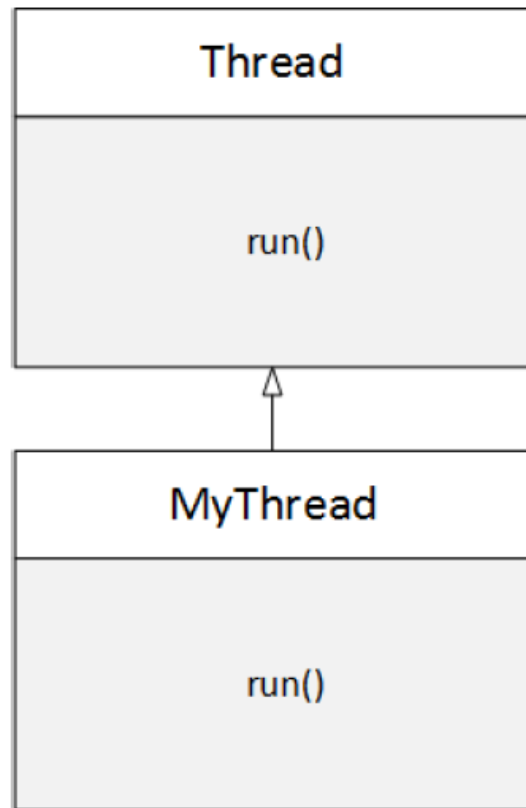
Actually two ways in Java:

- ❖ Extend the Thread class ( not a easy way)
- ❖ Implement the run() method derived from the Runnable interface

# Threads - Extending the Thread Class

## Not recommended

---



- Inherits the properties of the Java class `Thread`
- It *works*, but is considered to be the worse of the two options
- Doesn't allow for other inheritances, and does not improve on `Thread`



# Extending Java Thread

In this approach, you will create a thread by creating a new class that extends Thread, then override the run() method and then to create an instance of that class. The run() method is what is executed by the thread after you call start().

```
public class MyClass extends Thread {  
    public void run(){  
        System.out.println("MyClass running");  
    }  
}
```

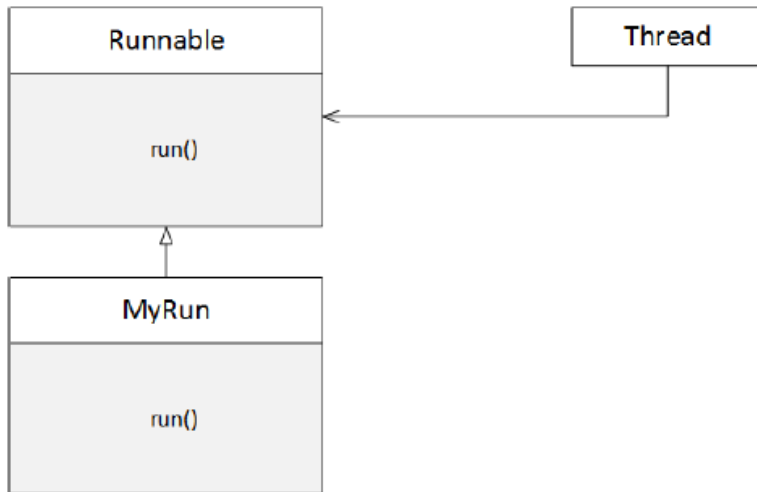
Not recommended

To create and start the above thread:

```
Myclass t1= new Myclass();  
T1.start();
```

When the run() method executes it will print out the text “**MyClass running**”.

# Threads - Implementing the Runnable Class



- Implementing the interface of `run()` from **Thread**
- Considered to be the cleaner method since it allows for composition

Recommended

# Runnable Interface

## Recommended

The easiest way to create a thread is to create a class that implements the **Runnable** interface.

To implement Runnable interface, a class need only implement a single method called run(), which is declared like this:

```
Public void run()
```

Inside run(), we will define the code that constitutes the new thread. Example:

```
public class MyClass implements Runnable {  
    public void run() {  
        System.out.println("MyClass running");  
    }  
}
```

To execute the run() method by a thread, pass an instance of MyClass to a Thread in its constructor (A **constructor in Java** is a block of code similar to a method that's called when an instance of an object is created). Here is how that is done:

```
Thread t1 = new Thread(new MyClass ());  
t1.start();
```

The Thread class defines several methods that help manage threads

Method	Meaning
getName	Obtain thread's name
getPriority	Obtain thread's priority
isAlive	Determine if a thread is still running
join	Wait for a thread to terminate
run	Entry point for the thread
sleep	Suspend a thread for a period of time
start	Start a thread by calling its run method

# Threads - States

---

Threads can be in any of the following states:

- **Created**: The thread is alive, and the `run()` method can be executed via `start()`
- **Runnable**: The thread is alive and been told to execute, but it has not been scheduled
- **Running**: The thread is currently executing `run()` and can be paused or terminated
- **Non-Runnable**: The thread is paused (or put to sleep), by resuming it can be made **Runnable** again
- **Terminated**: Either the `run()` method has completed or it is stopped via `stop()`

# There are no concrete definitions for concurrency.

---

- ❖ Each definition is explaining concurrency from different perspective.
  - ❖ 1<sup>st</sup> operating system perspective
    - ❖ Concurrency is when **two or more tasks** can start, run, and complete in overlapping time periods in no specific order. **This significantly improve overall speed of the execution.**
  - ❖ 2<sup>nd</sup> definition defines concurrency in general term.
    - ❖ Another definition for concurrency is that, concurrency is the act of managing and running multiple computations at the same time.
  - ❖ 3<sup>rd</sup> from programming perspective.

# What is LTSA and LTSA Tool.

---

**A system in LTSA** (Labelled Transition System Analyser) is modelled as a set of interacting finite state machines. The properties required of the system are also modelled as state machines.

More formally, each component of a specification is described as a Labelled Transition System (LTS), which contains all the states a component may reach and all the transitions it may perform.

**LTSA Tools** is a verification tool for concurrent systems.

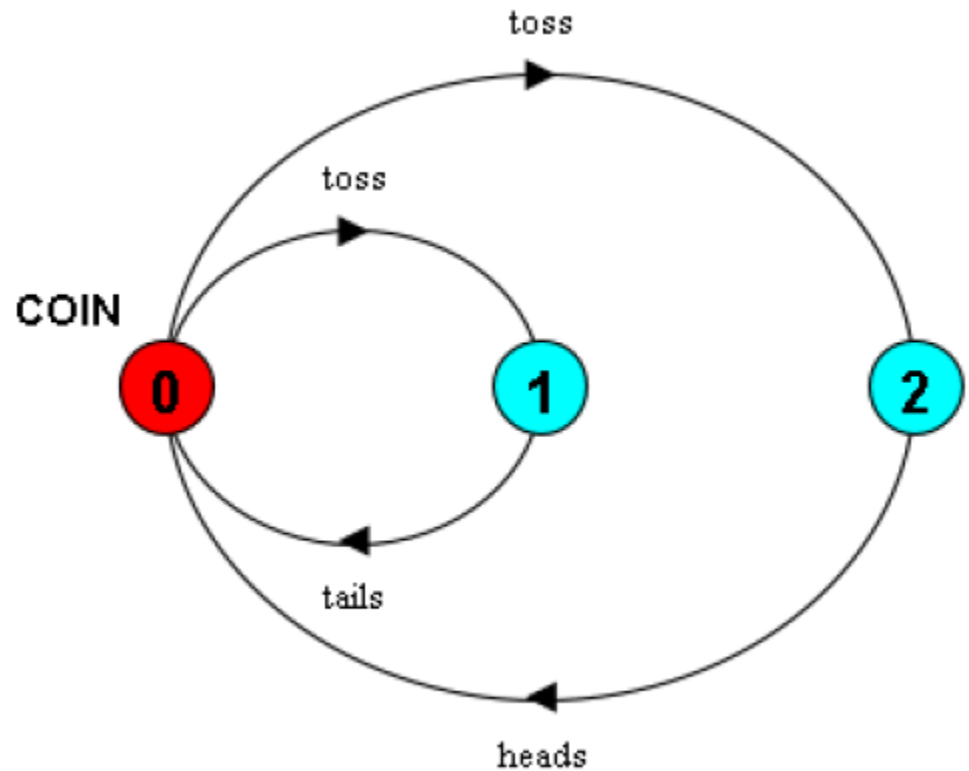
It mechanically checks that the specification of a concurrent system satisfies the properties required of its behaviour.

In addition, LTSA supports specification animation to facilitate interactive exploration of system behaviour.

# LTSA Tool (Labelled Transition System Analyser)

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

```
menu RUN = {toss}
```





# Download link for LTSA tool

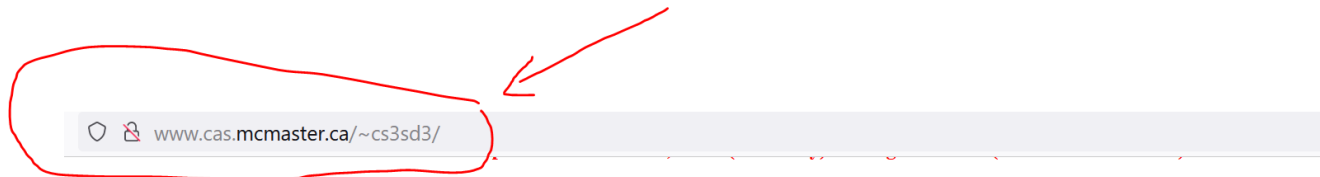
---

LTSA tool from <http://www.doc.ic.ac.uk/~jnm/book>

**Direct download link**

<http://www.doc.ic.ac.uk/~jnm/book/ltsa/ltsatool.zip>

# Screenshot from course website.



## Course Outline (Tentative)

- Nature of Concurrency. Processes and Threads. Concurrent Execution. Labelled Transition Systems. Shared Objects and Mutual Exclusion. Monitors and Condition Synchronization. Deadlock. Safety and Liveness Properties. Model Based Design. Dynamic Systems. Message Passing. Concurrent Architectures. Timed Systems. Program Verification. Logical Properties. Perti Nets and other models.

## Texts:

- J. Magee, J. Kramer, *Concurrency: State Models & Java Programming*, 2nd Edition, J. Wiley 2006.

*The course may not always follow the text-book closely.*

Lecture Notes (copies of transparencies) will be on the website a few days after a class.

Software Tools: You will have to obtain the LTSA tool from <http://www.doc.ic.ac.uk/~jnm/book>

## Calendar Description:

- Processes, threads, concurrency; synchronization mechanisms, resource management and sharing; objects and concurrency; design, architecture and testing of concurrent systems.

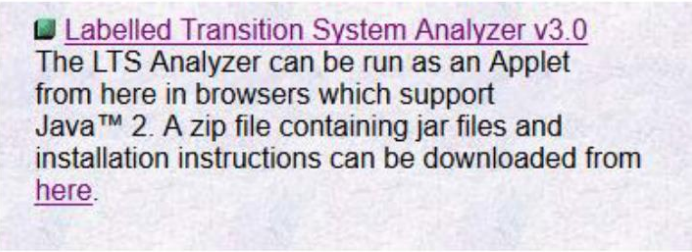
## Mission:

- The mission of this course is to give students an understanding of the basic techniques of concurrent system design and to introduce students to the appropriate modelling and analysis techniques.

# To install LTSA tool

---

- Download LTSA tool from
- <http://www.doc.ic.ac.uk/~jnm/book>



Labelled Transition System Analyzer v3.0  
The LTS Analyzer can be run as an Applet from here in browsers which support Java™ 2. A zip file containing jar files and installation instructions can be downloaded from [here](#).

- 
- All Examples of the textbook are already inside the tool

Textbook Chapter 2.1.3 page 18

You must install JAVA, in order run LTSA tool.

**Most windows 10 machines already have Java installed.**

- You can download and install it from here
- You can learn more LTSA command from this website.
- Very useful for your assignments.

Download the latest version of Java SE Development Kit (SDK) from the following link. As of December 2020 the latest version is version 15.01.

## Download Link

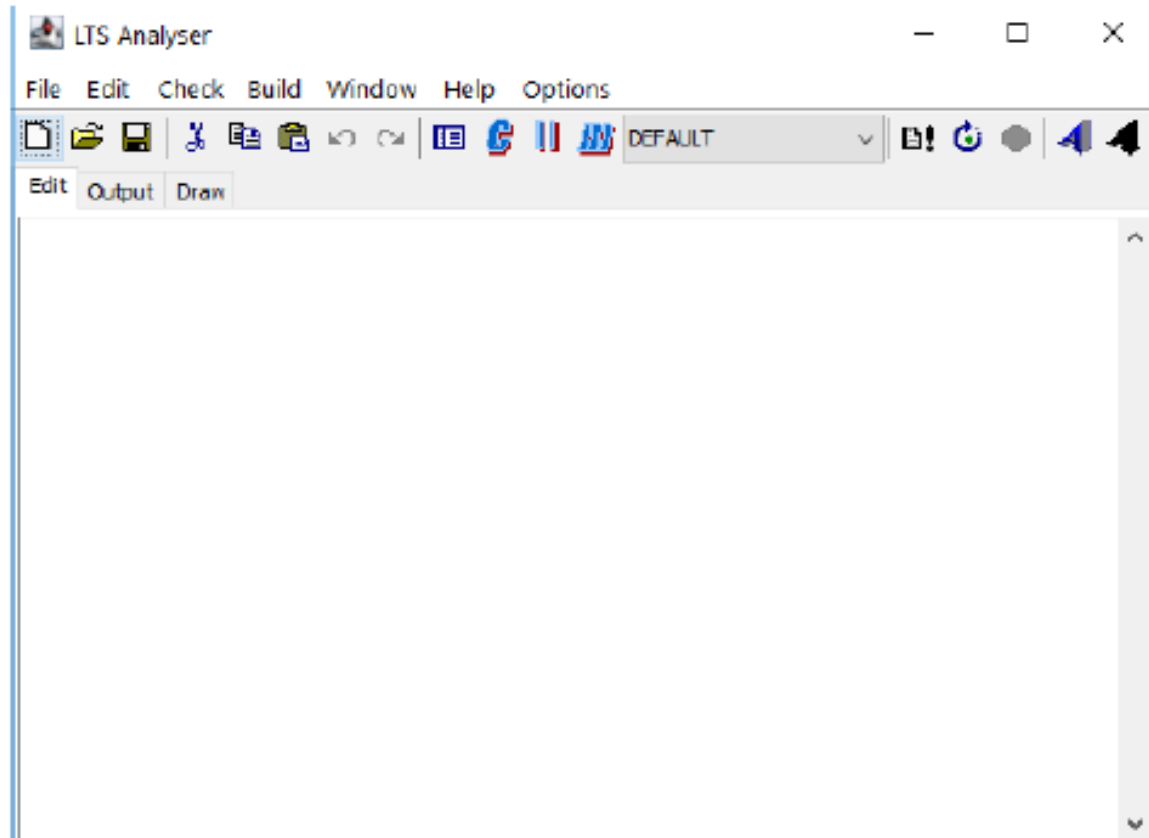
or You can install java from here

<https://ninite.com> (I suggest this website)

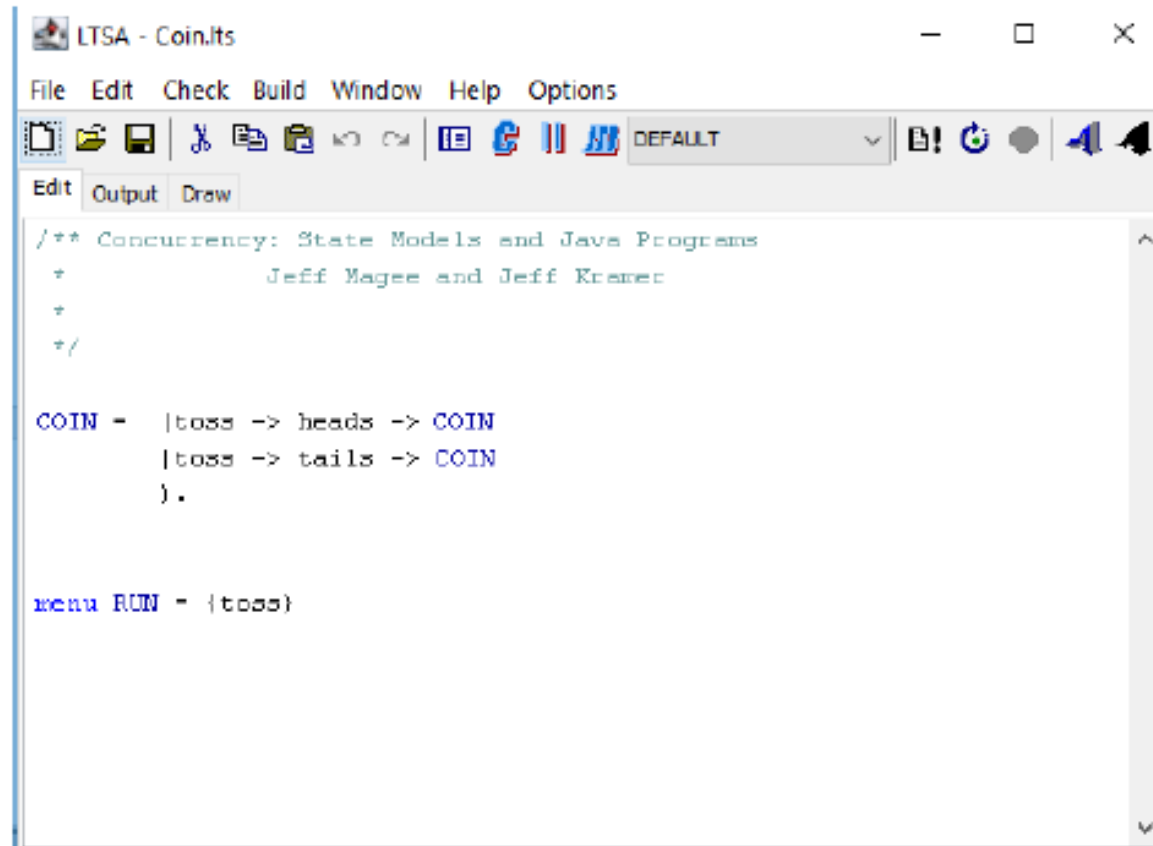
- After you install JAVA correctly on you system.
- Download and unzip the tool to your computer
- And Click on **Itsa.jar**

Chapter_examples	4/18/2006 7:45 AM	File folder	
create_file_assoc.vbs	4/19/2006 7:37 AM	VBScript Script File	1 KB
create_shortcut.vbs	4/19/2006 7:37 AM	VBScript Script File	1 KB
Itl2buchi.jar	10/8/2003 7:41 AM	WinRAR archive	79 KB
Itsa.bat	4/18/2006 9:11 AM	Windows Batch File	1 KB
Itsa.ico	12/4/1998 2:58 PM	Icon	1 KB
<b>Itsa.jar</b>	4/19/2006 5:49 PM	WinRAR archive	440 KB
Itsafile.ico	12/4/1998 2:57 PM	Icon	1 KB
README.txt	4/18/2006 2:20 PM	Text Document	1 KB

- 
- And you will get this window.



-You can find textbook example in **File ->Examples ->Chapter 2->Coin**



The screenshot shows a window titled "LTSA - Coin.lts" with a menu bar (File, Edit, Check, Build, Window, Help, Options) and a toolbar. Below the toolbar are tabs for "Edit", "Output", and "Draw". The main text area contains the following code:

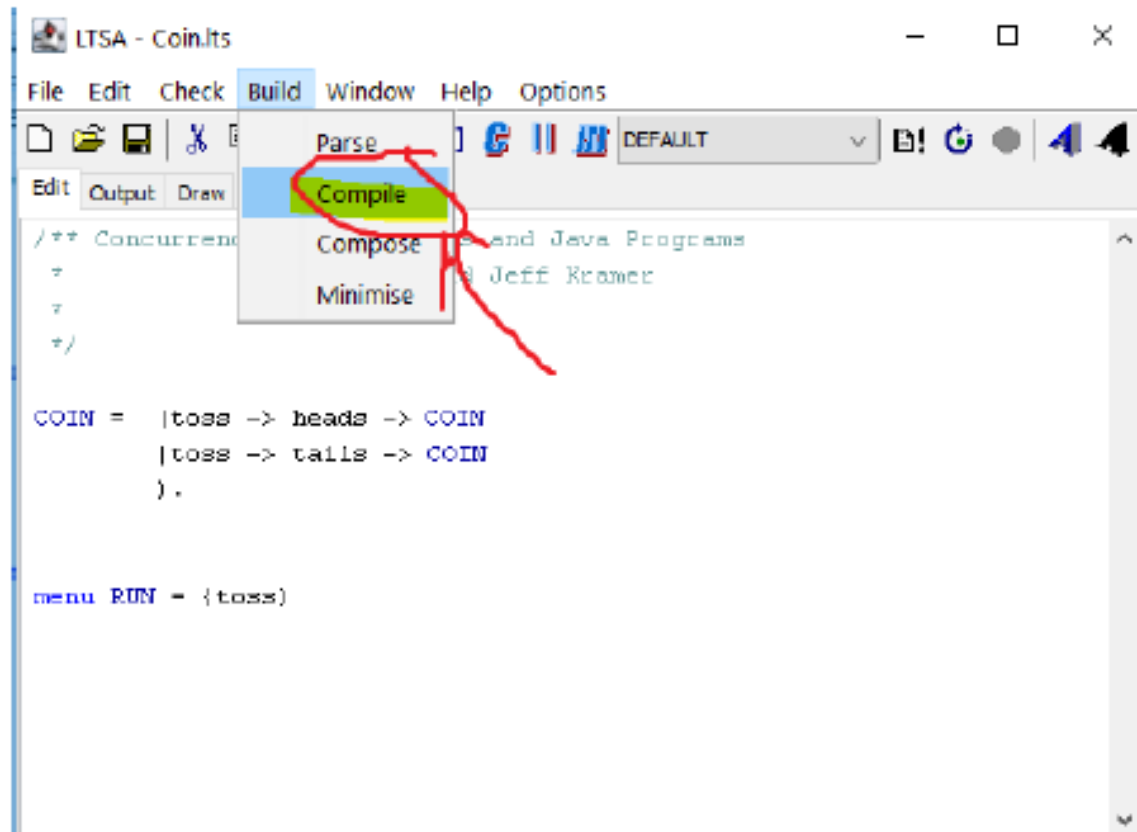
```
/** Concurrency: State Models and Java Programs
 *      Jeff Magee and Jeff Kramer
 */

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

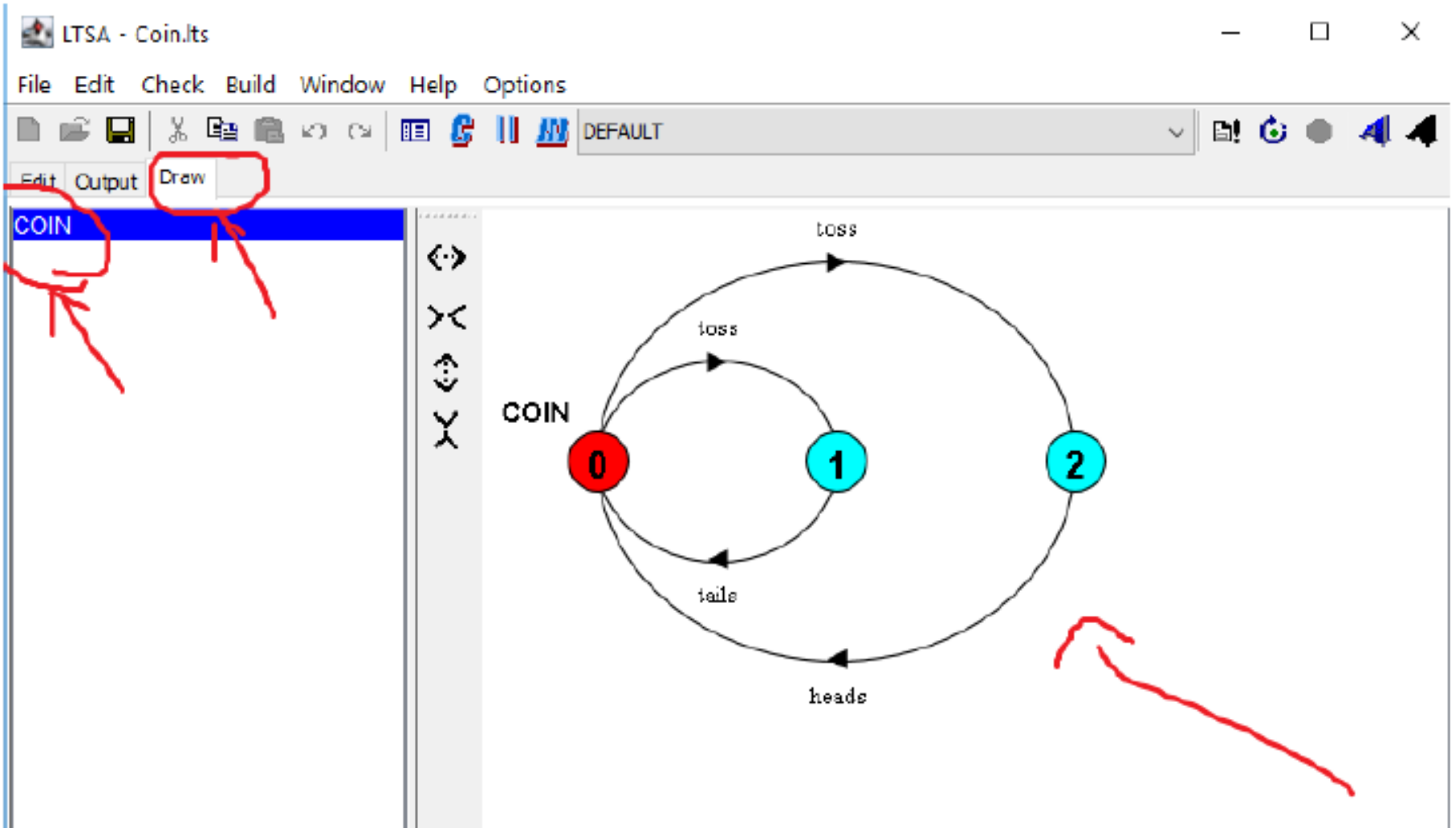
menu RUN = {toss}
```

You can Compile you code from

**Build -> Compile**



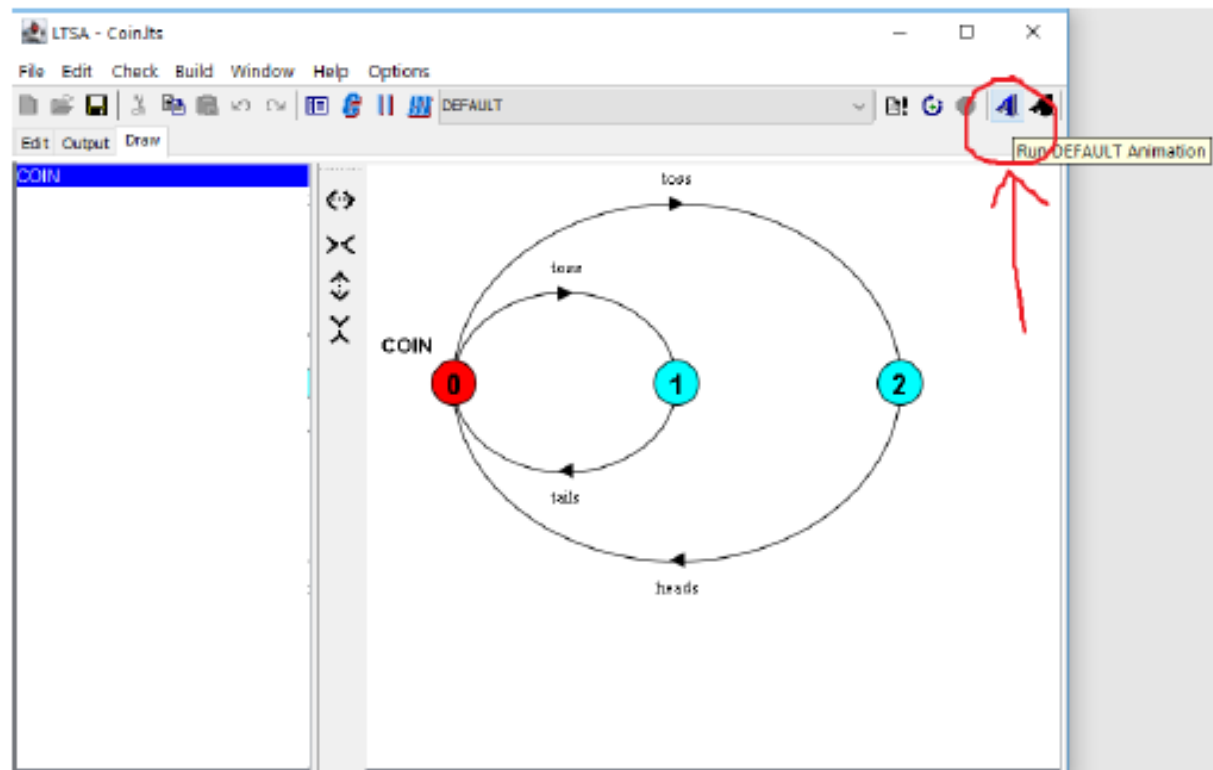
Then click on Draw and select your code (coin) and you should be able to see the diagram



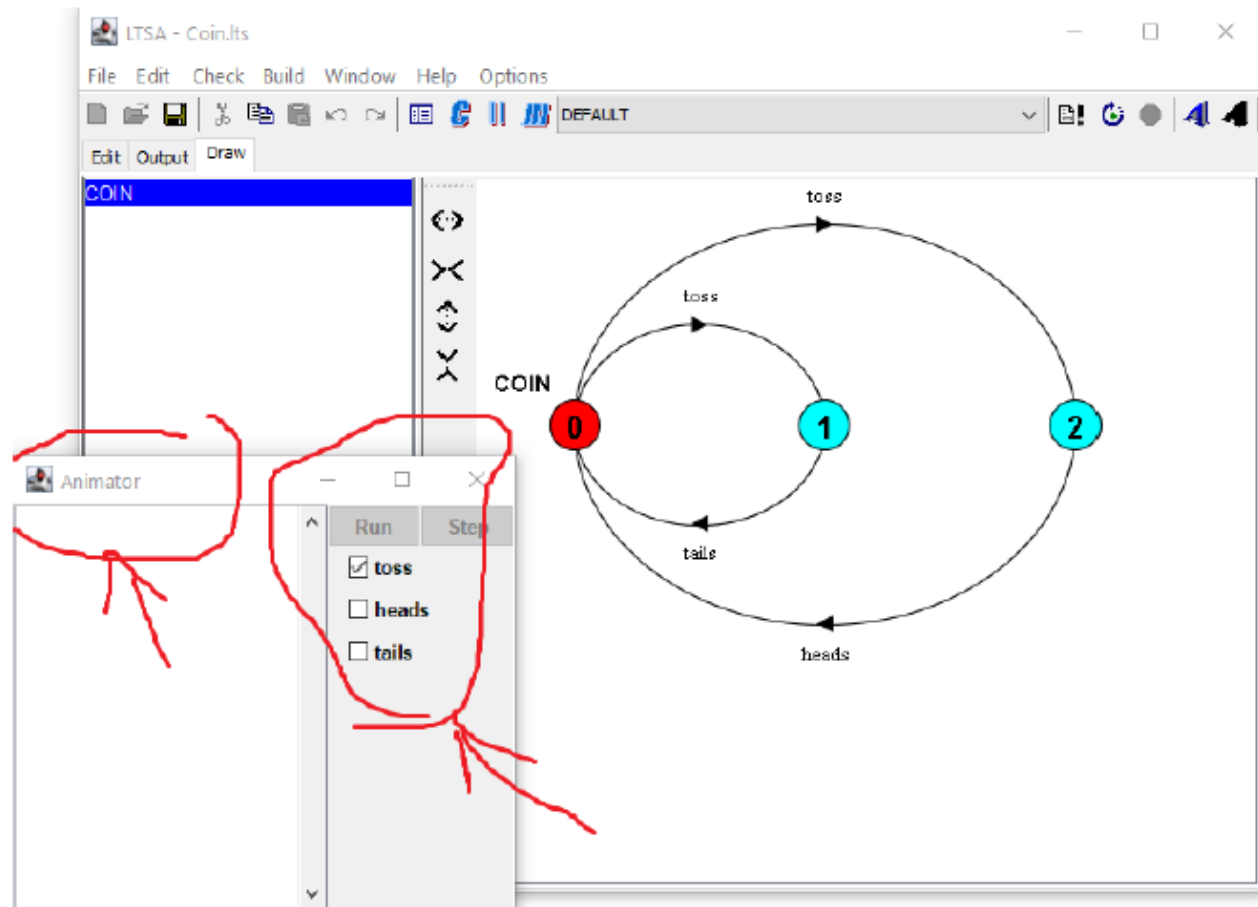




The Animator tool can be accessed by clicking on the **Blue (A)** icon on the top right side of the program.



The animator window will look like this



- The animator tool allows the LTS corresponding to a FSP specification to be viewed graphically

# More informational links

---

[https://www.doc.ic.ac.uk/~jnm/book/book\\_applets/concurrency.html](https://www.doc.ic.ac.uk/~jnm/book/book_applets/concurrency.html)

# Two examples in LTSA

---

Coin.

Drinks.

---

# Any Questions?