Actors

Lecture Question

Task: Create and test an actor class to set and sound an alarm

- In a package named actors, create a class named AlarmActor that extends Actor
- Create the following case class in the actors package that will be used as messages
  - A case class named SetAlarm that takes a Double and a String in its constructor
  - A case class named Alarm that takes a String in its constructor
- The AlarmActor class must respond to messages of type SetAlarm by:
  - Interpreting the Double as the number of seconds to wait before sounding the alarm
  - When the alarm "sounds," send an Alarm message back to the actor that sent the SetAlarm message containing the same String that was in the SetAlarm message
- **Testing**: Write a test suite named tests.TestAlarmActor to test this functionality

Stock Trader Example

- Simulate stocks changing prices and a trader purchasing stocks

- Stock
  - Receives a Tick message and changes its price
    - Price changes are random for this simulation
    - Tick messages are sent to all stocks at regular intervals
  - Receives a GetPrice message and responds with its current price

- Trader
  - Knows each stock's ticker symbol and actor reference
  - Receives a CheckStocks message and checks the price of all known stocks
  - Receives Price messages from stocks and decides [randomly] whether or not to buy some shares

To the Code

Traffic Example

- Intersections
  - North/South road intersects with an East/West road
  - Initially Green light East/West
  - Alternate green light at a fixed interval

- Cars
  - Have a List of directions to follow
  - Each direction has:
    - A destination Intersection
    - The time it takes to reach that Intersection
    - Whether the intersection will be approached from the East/West or North/South

Traffic Example

- When a car approaches an intersection:
  - Inform the Intersection
  - Intersection gives the GreenLight for the car to go
    - If the light is red, the Intersection will wait until it's green before sending the GreenLight message

- When a car receives a GreenLight message
  - Proceed to next Intersection in the directions list

Delayed Messages

- There are two cases where we'll have messages that need to be sent with a delay
  - Intersections changing lights at fixed intervals
  - Cars waiting the appropriate amount of time before checking an intersection for a green light
- Similar syntax as repeatedly sending a message (Supervisor example)
- Use scheduleOnce

```
import context.dispatcher
// ...
context.system.scheduler.scheduleOnce(
  timeInterval.milliseconds,
  self,
  ChangeLight
)
```

Delayed Messages

- Must import the dispatcher from inside the Actor's class
- Specify:
  - The time delay
  - The recipient
    - self contains an ActorRef to this actor
  - The message to be sent

```
import context.dispatcher
// ...
this.context.system.scheduler.scheduleOnce(
  timeInterval.milliseconds,
  self,
  ChangeLight
)
```

Actors Creating Actors

- this.context can be used to add new actors to the system
- Use the same "actorOf" syntax as used when adding actors using the system directly

```
this.context.actorOf(Props(classOf[TimerActor]))
```

To the Code

Lecture Question

Task: Create and test an actor class to set and sound an alarm

- In a package named actors, create a class named AlarmActor that extends Actor
- Create the following case class in the actors package that will be used as messages
  - A case class named SetAlarm that takes a Double and a String in its constructor
  - A case class named Alarm that takes a String in its constructor
- The AlarmActor class must respond to messages of type SetAlarm by:
  - Interpreting the Double as the number of seconds to wait before sounding the alarm
  - When the alarm "sounds," send an Alarm message back to the actor that sent the SetAlarm message containing the same String that was in the SetAlarm message
- **Testing**: Write a test suite named tests.TestAlarmActor to test this functionality