Object Oriented Methodology

Week – *9*, Lecture – *1*

Expressing Interactions

SAURABH SRIVASTAVA

VISITING FACULTY

IIIT LUCKNOW

Interactions between Objects

A State Diagram shows how an object changes during the course of execution

- It involved showing prominent events that affected the state of the object …
- … along with the changes they brought upon over the object

But how can you show a broader picture of the system?

- For example, how do the states of different objects get affected on the occurrence of an event?

This is where, a Sequence Diagram can be helpful

While a Sequence Diagram can be used to show a variety of information …

- … in the most simple formulation, it shows the “interactions” between different objects in the system

A Sequence Diagram also provides a temporal understanding of the system’s behaviour

- This is because you can show the relative ordering of events in a Sequence Diagram

Sequence Diagrams

A Sequence Diagram can be extremely detailed

- This includes showing Actors – entities that are external to the system, but interacts with it
- We will talk about Actors in a while when we discuss Use Case Diagrams

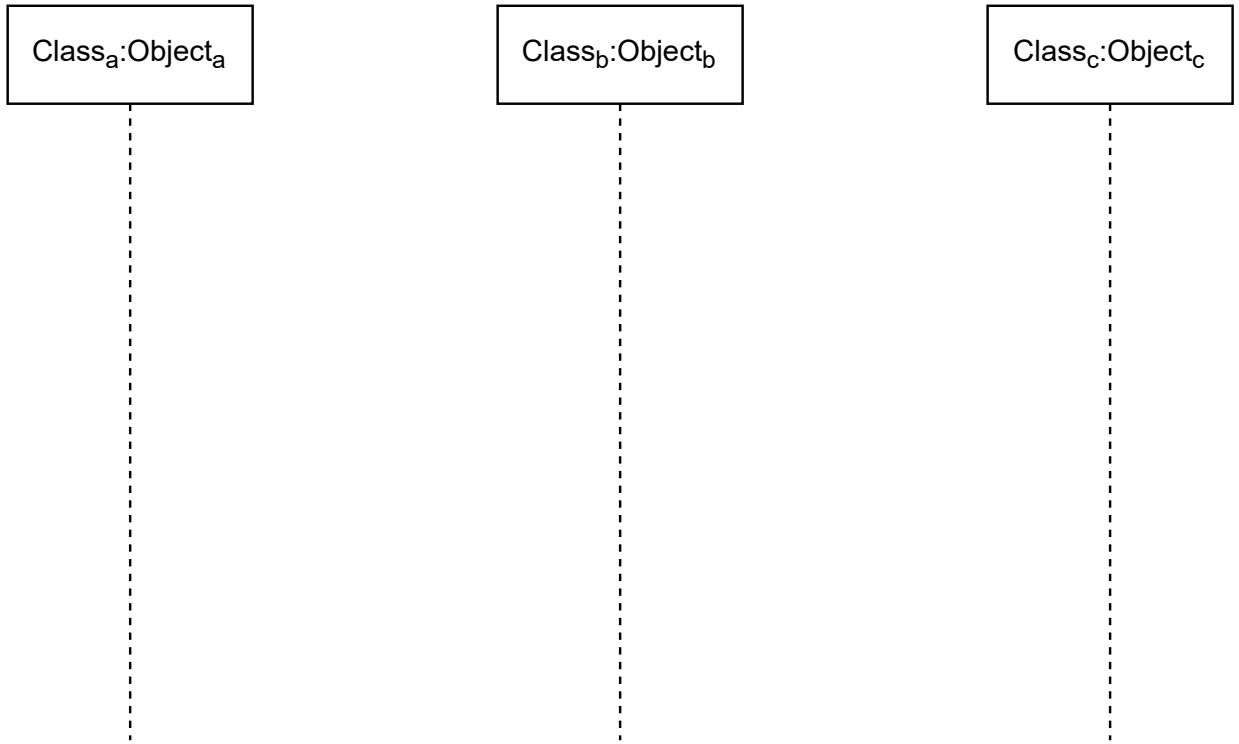
However, we are only going to discuss the most common usage for Sequence Diagram …

- … i.e., showing the sequence of events in a system
- Implicitly, it also shows the objects which are affected during any event

A Sequence Diagram contains Vertical and Horizontal elements

- The Vertical Elements showcase an object (or an Actor)
- The Horizontal Elements can show an event or an interaction between different objects

Objects in a Sequence Diagrams



Objects in a Sequence Diagram - they can be of same or different classes

Sequence Diagrams

Despite its complex elements, we will discuss the most common usage for Sequence Diagram …

- … i.e., showing the sequence of events in a system (which is what you may need for your projects)
- Implicitly, it also shows the objects which are affected during any event

A Sequence Diagram contains Vertical and Horizontal elements

The Vertical Elements showcase an object (or an Actor)

- The Horizontal Elements can show an event or an interaction between different objects
- The objects have a *lifeline* – the dotted line that runs from top to bottom

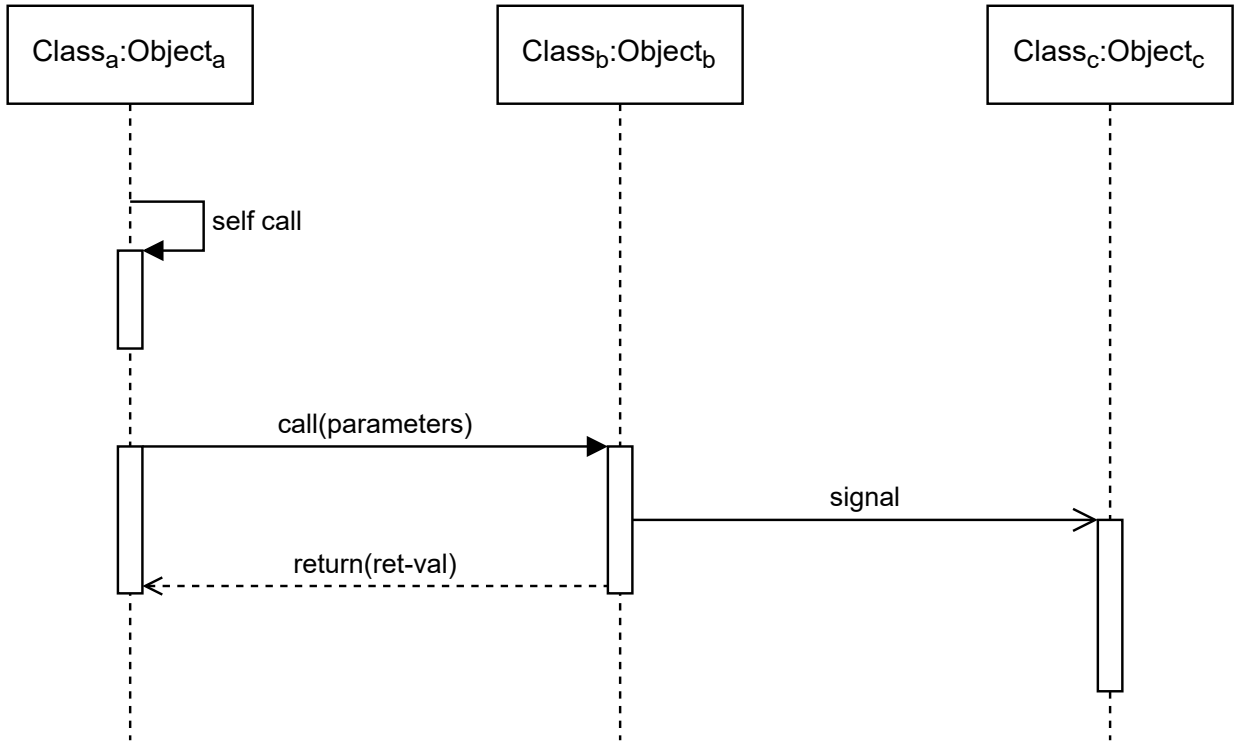
A rectangle is added to the lifeline, when we want to show its interaction with others

- This rectangle is called *activation* for the object

The Horizontal elements, i.e., the arrows, represent events or interactions

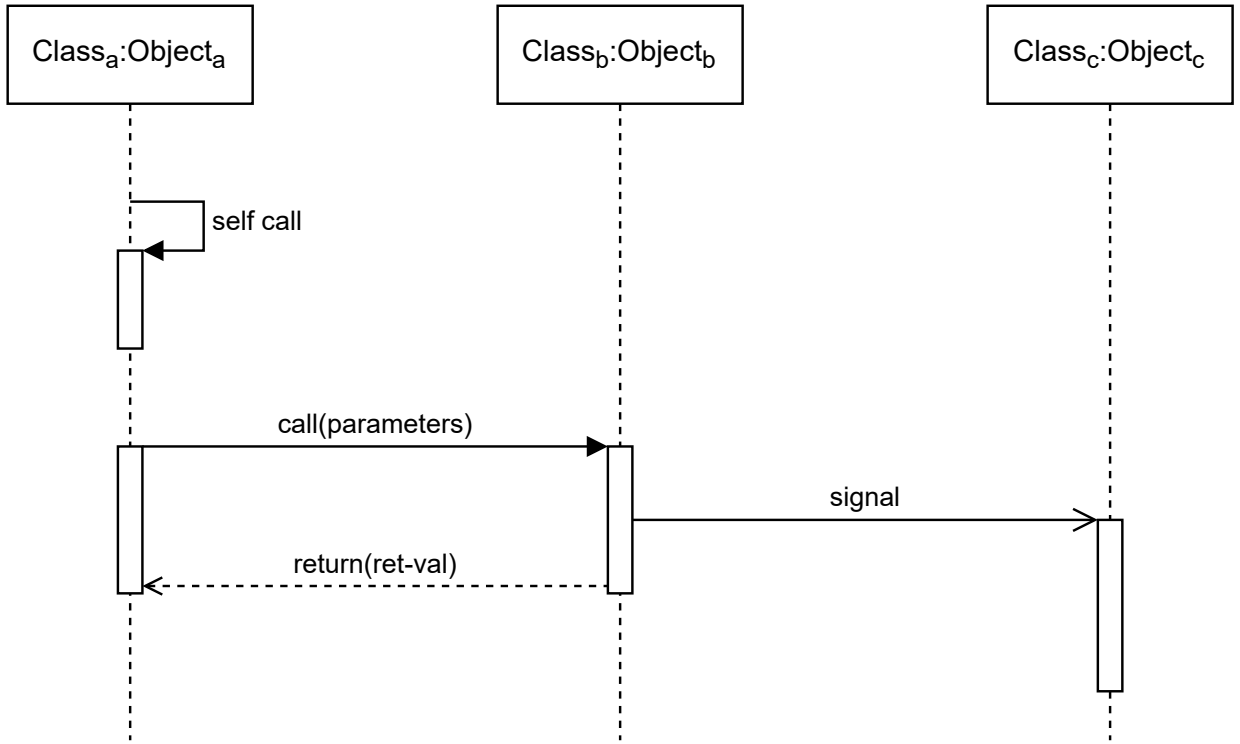
- A single, unidirectional solid arrow is a signal – which can be used to show events
- A pair of solid and dotted arrows can be used to show a *call and return* interaction between objects

Adding Interactions in a Sequence Diagram



Activations, calls, return messages and signals in a Sequence Diagram

Adding Interactions in a Sequence Diagram



An interaction that appears higher in the Sequence Diagram, happens *before* those who appear lower

However, they only show a relative ordering – the actual distance between the interactions do not represent any time duration as such

Activations, calls, return messages and signals in a Sequence Diagram

Describing the System's Use Cases

Till now, we have discussed things at a rather detailed level

- The class diagram gives you information about the classes and their relationships within the system
- The state diagrams show how objects change during the execution, albeit, one object at a time
- The sequence diagrams show the communication between the objects

Let us now take a step back, and try to see the system "as a whole"

- Basically, we want to see how the system interacts with the external entities – e.g. its users

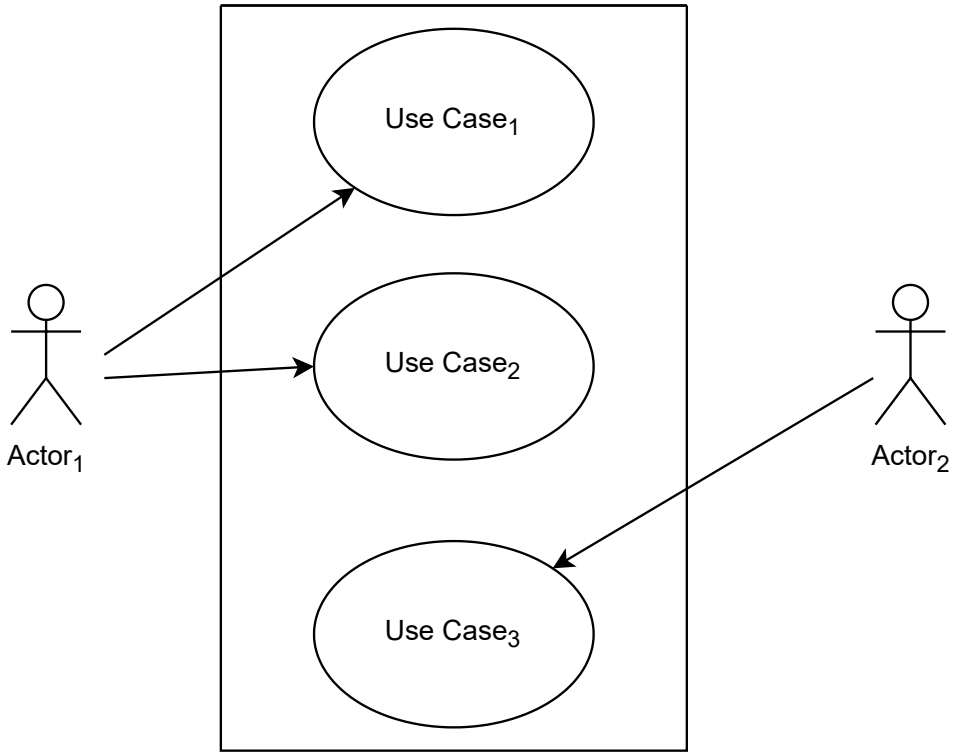
These interactions are often described as a *use case*

- While you often define the details in text, there is a diagram to show the big picture – Use Case Diagram

Again, use case diagrams can be used to display complex scenarios …

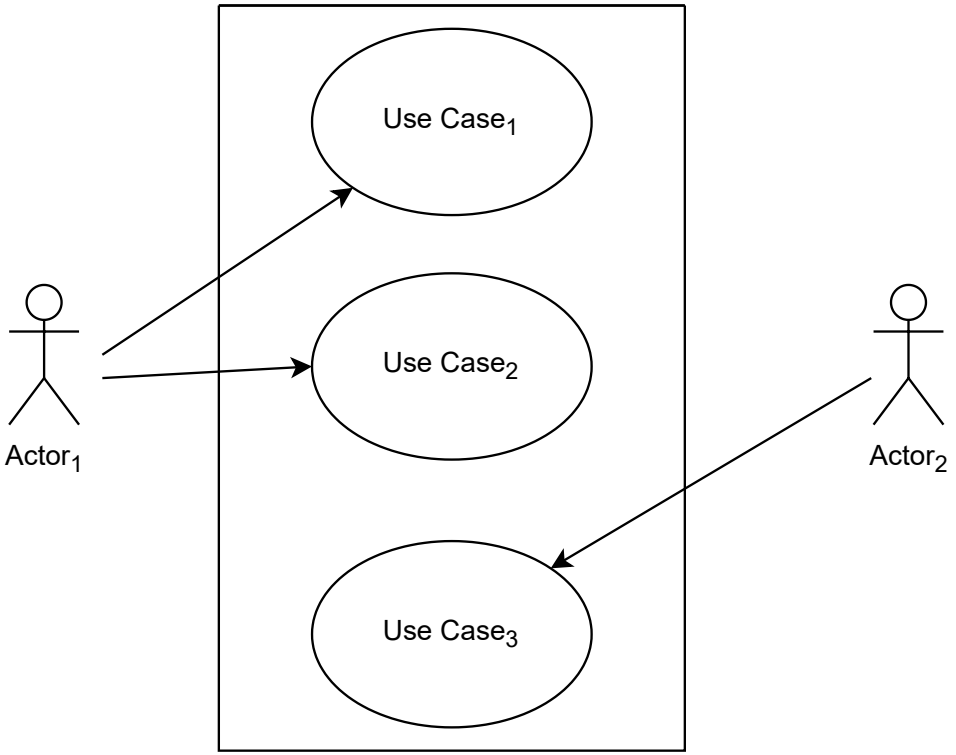
- … but for our purpose, just showing the relationship between the Actors and the system suffices
- *Actors* are entities that are not a part of the system, but interact with it for various reasons
- The system comprises of a box containing oval boxes representing use cases, linked to different Actors

A Sample Use Case Diagram



A use case diagram with two actors and three use cases

A Sample Use Case Diagram



A use case diagram with two actors and three use cases

Examples of Actors – `customer` and `administrator`

Examples of Use cases – "`create a service request`" and "`approve a service request from a customer`"

One more Interaction Diagram !!

We have probably already talked a bit too much about interactions

But there is one more diagram that you should be aware of :P

- It is called the *Activity Diagram*

An Activity Diagram allows you a completely different way to express interactions

- In layman's terms, an Activity Diagram is a distant cousin of a Flowchart !!

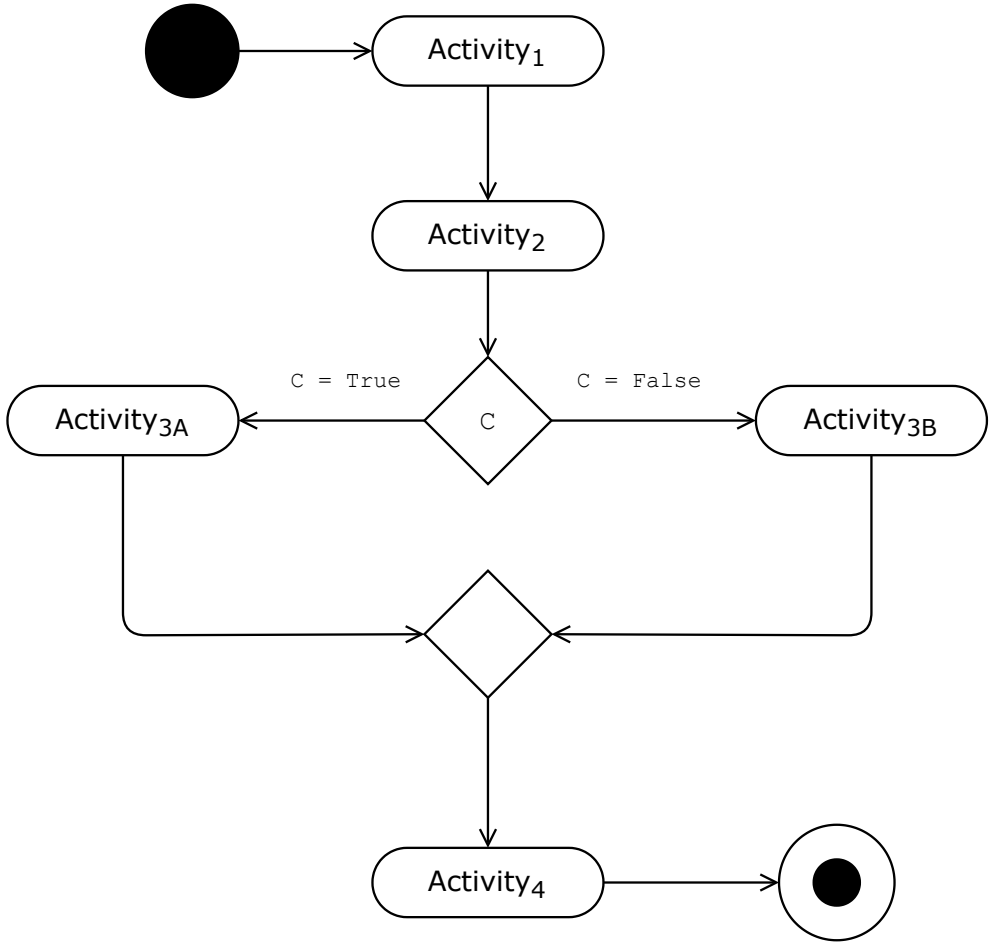
The boxes in an Activity Diagram include activities, branching elements and parallelism indicators

The lines in the diagram show progression – from one activity to another

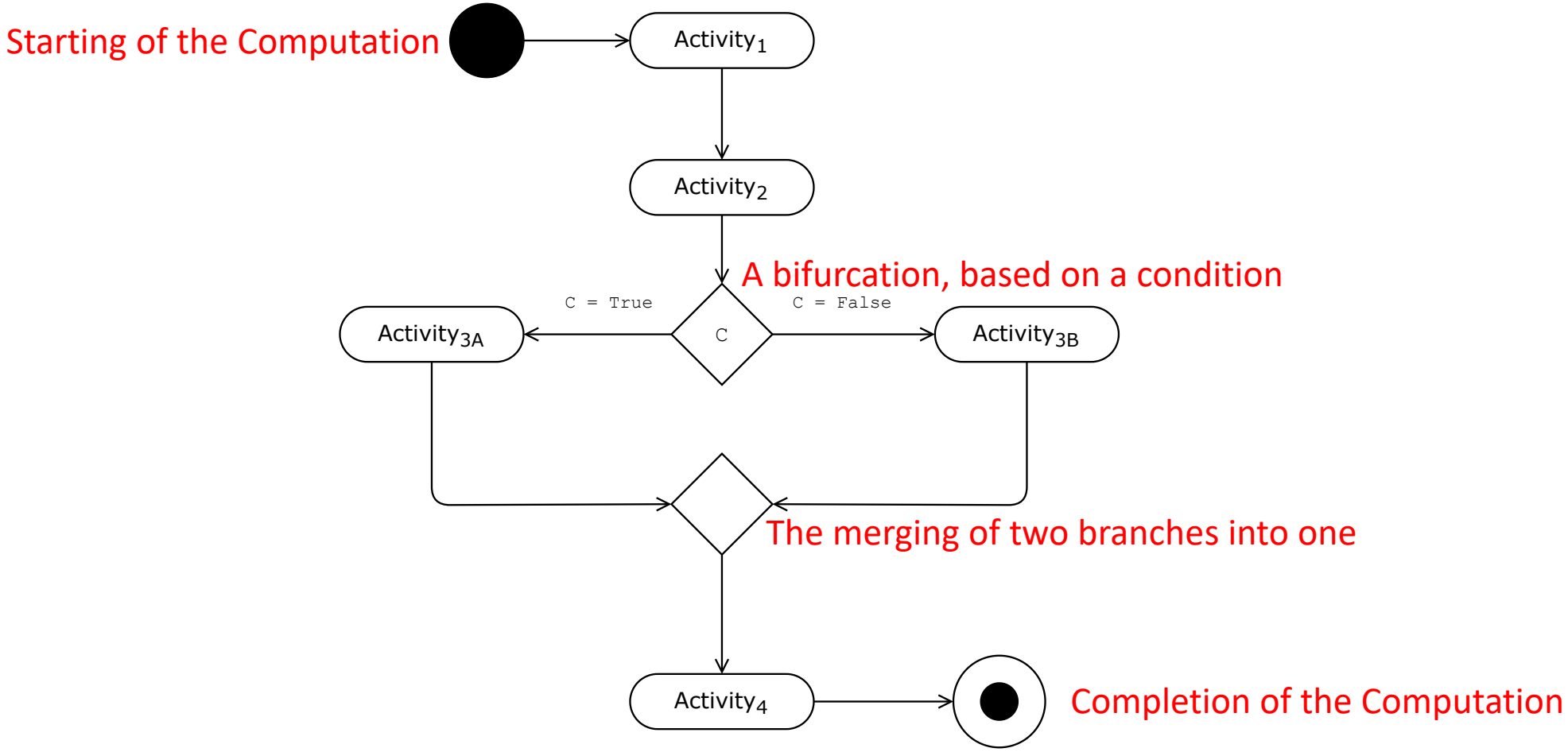
You can also *dissect* an Activity and show sub-activities and the progression between them

- Often, these sub-activities have close relationship with events from State and Sequence Diagrams
- They are also called *Actions,* which are basically the operations taken as part of completing an activity

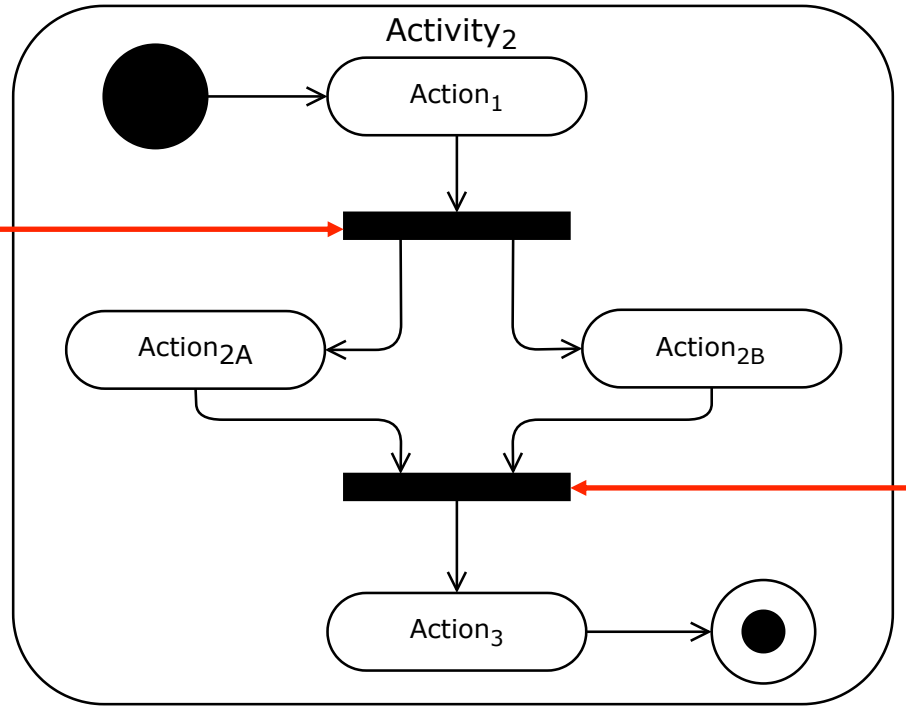
Activity Diagram as System Overview



Activity Diagram as System Overview



Activity Diagram as System Overview



A *fork* in computation – indicating Parallel Processing

A *join* in computation – merging previously forked Processing

Finer details of Activity$_2$

Representing Interactions – In a nutshell

For you projects, a good way to go about could be the following sequence

Start with the use cases for your project

- If the use cases can be expressed in short with a few words, draw the Use Case Diagram …
- … since it can show the use cases and the associated actors in a compact fashion

Next, you may choose to create Sequence Diagrams

- It is a good idea to first prepare Class Diagram(s) and State Diagrams

Activity Diagrams provide a different perspective of the System's behaviour

- Instead of focussing on objects, it focuses on operations
- Use Activity Diagrams, only if you feel that they contribute towards a better understanding

Homework !!

Figure out what diagrams can represent the behavioural aspects of your project better

- It is not necessary that you use the diagrams that we covered
- You can have a look at other behavioural modelling diagrams here: https://sparxsystems.com/resources/tutorials/uml2/index.html
- **You must have at least one behavioural diagram in the project report though**