Software Engineering

Lecture 4 –
Use-case specification with activity diagrams

Bogumiła Hnatkowska

Lecture objectives

- To present activity diagrams
- To present how to use activity diagrams to use-case specification

How to specify use-cases?

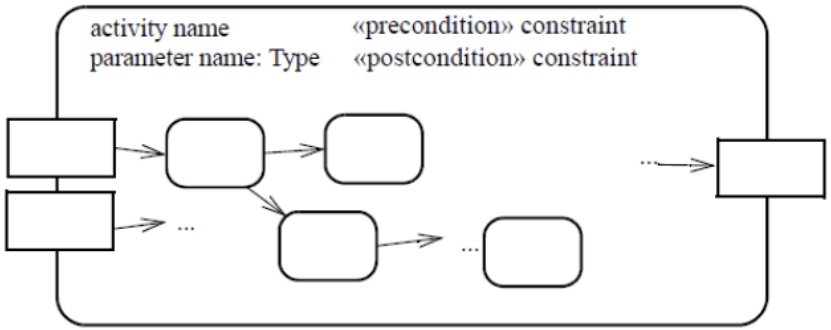
- Textual descriptions in natural language – use case specification
- **Activity diagrams**
- Sequence diagrams

The usage of activity diagrams

- Activity diagram can be used for modeling:
  - computations (of programs or its pars, e.g. methods)
  - workflows in human organizations (business processes)
  - use-case behavior
- Activity diagram allows to present both control-flows and data flows

Activity (diagram)

- Activity – a behaviour represented by a graph of activity nodes and activity edges
- May have input parameters and produce some output values
- May have some preconditions/postconditions defined

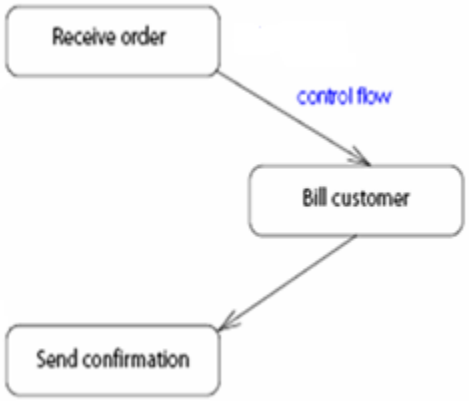


Activity (diagram) elements

- Activity nodes:
  - Executable nodes – description of behaviour
  - Control nodes – „traffic switches” for control tokens
  - Object nodes – holders for object tokens
- Activity edges:
  - Control flow – to pass control tokens
  - Object flow – to pass object tokens

Executable nodes and basic control flow

- Executable node - a kind of activity node that models the individual step in the behaviour specified by activity diagram
- The step can be a primitive computational step (action), e.g. variable modification, or a group of actions (activity), e.g. a task in a workflow like „bring a coffee” or a procedure call
- The step to be executed must be offered tokens from all incoming control flows, and after its completion offers tokens to all outgoing control flows
- Control flow – an arrow which links 2 action nodes (obligatory element)

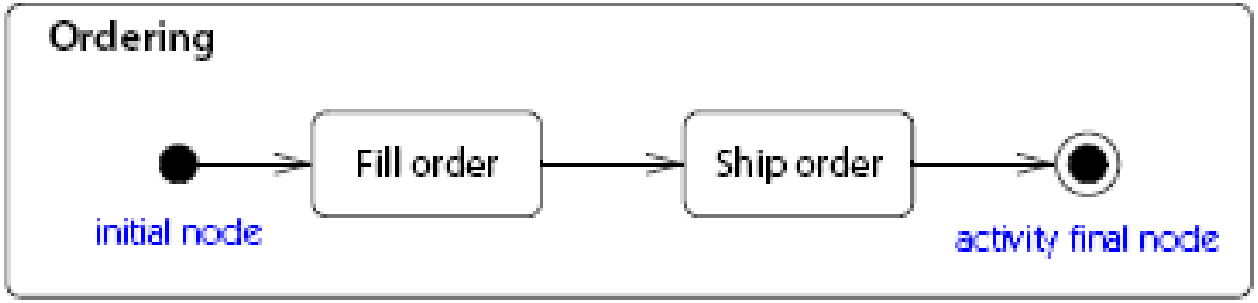


Control nodes

- Transfers control tokens between execution nodes:
  - Initial + Activity final + Flow final
  - Decision + Merge
  - Fork + Join
  - Flow final

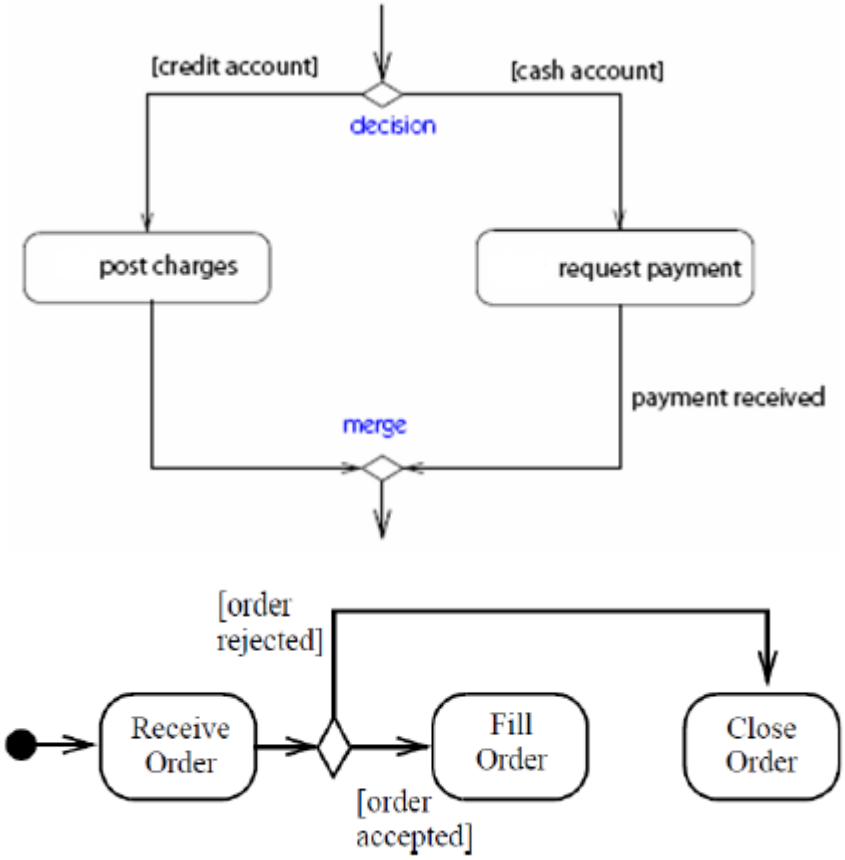
Initial + Activity final

- Initial node – has no input edge and one output edge
  - Represents the default starting point of the activity (diagram)
  - When the activity (diagram) is invoked, the output of the initial node becomes enabled (the token is transferred)
- Activity final node – finishes the whole activity (all flows); has one or more input edges and no output edge
  - If any of the input edges are enabled the execution of the enclosing activity is terminated
  - It is possible to have many activity final nodes

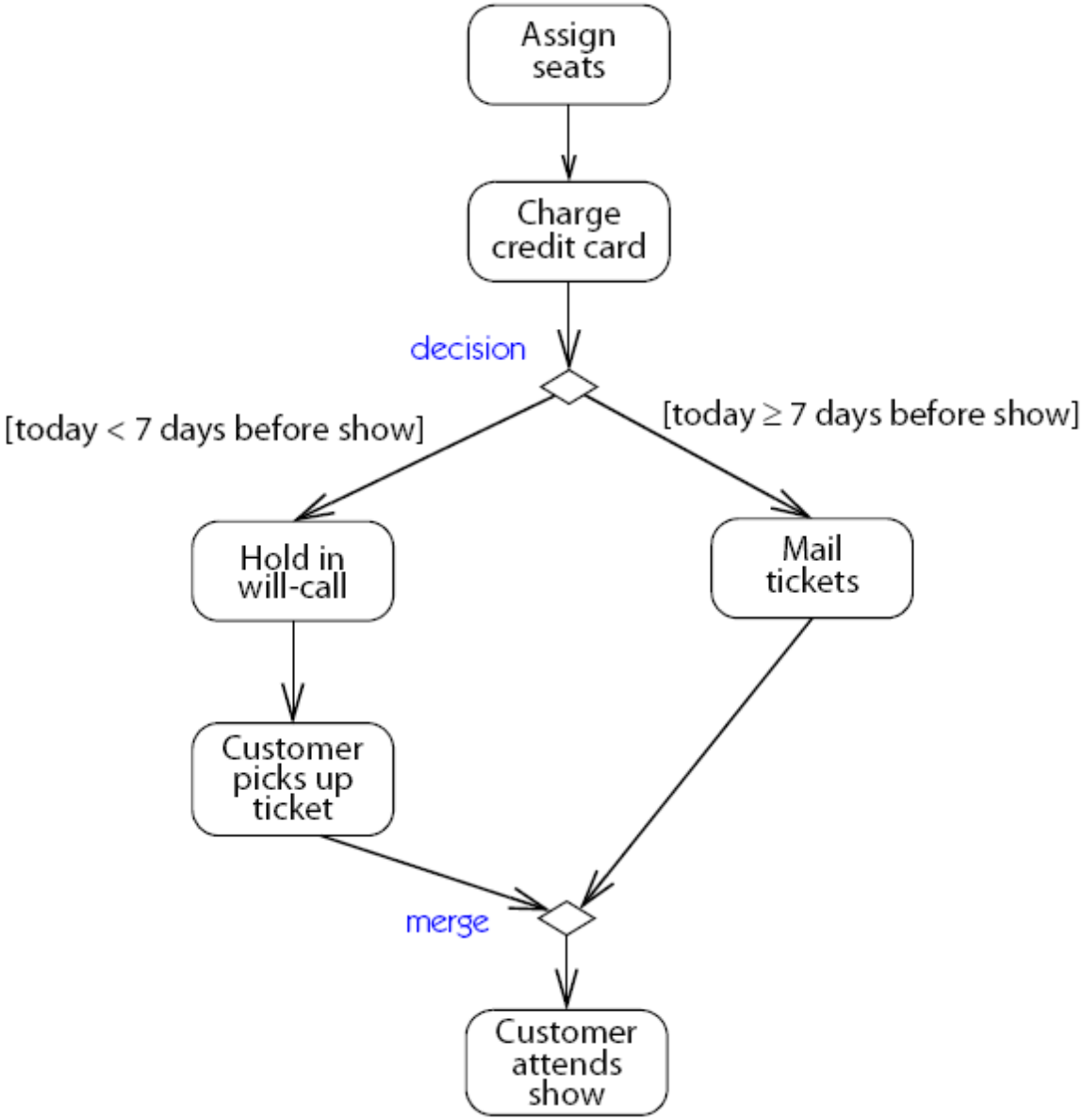


Decision + Merge

- Decision node – has one input edge and multiple output edges (with guards):
  - Only one output edge is enabled even if more than one guard is satisfied
  - One output may be labeled with „else”
  - If no guard is true, the model is ill formed
- Merge – a place at which two or more alternate path of control come together
  - When any input edge is enabled the output edge is enabled

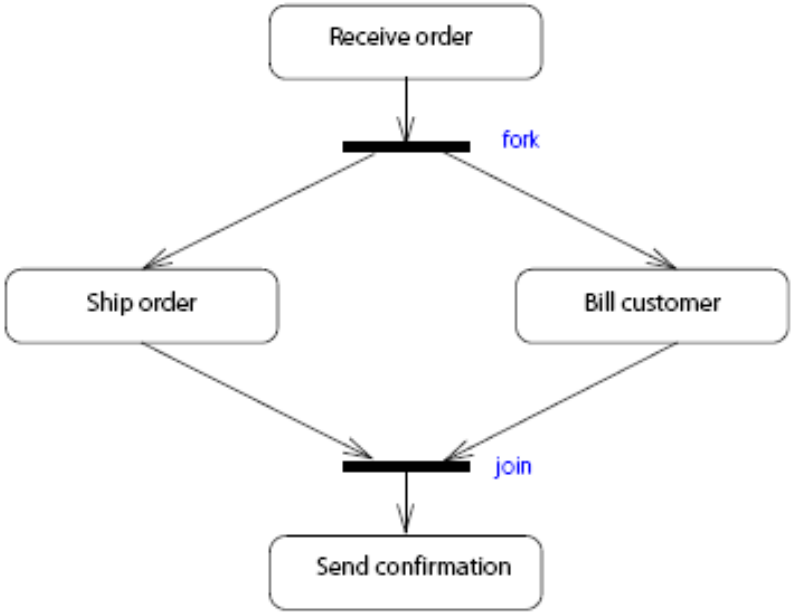


Example


Assign seats
Charge credit card
decision
[today < 7 days before show]
[today ≥ 7 days before show]
Hold in will-call
Mail tickets
Customer picks up ticket
merge
Customer attends show

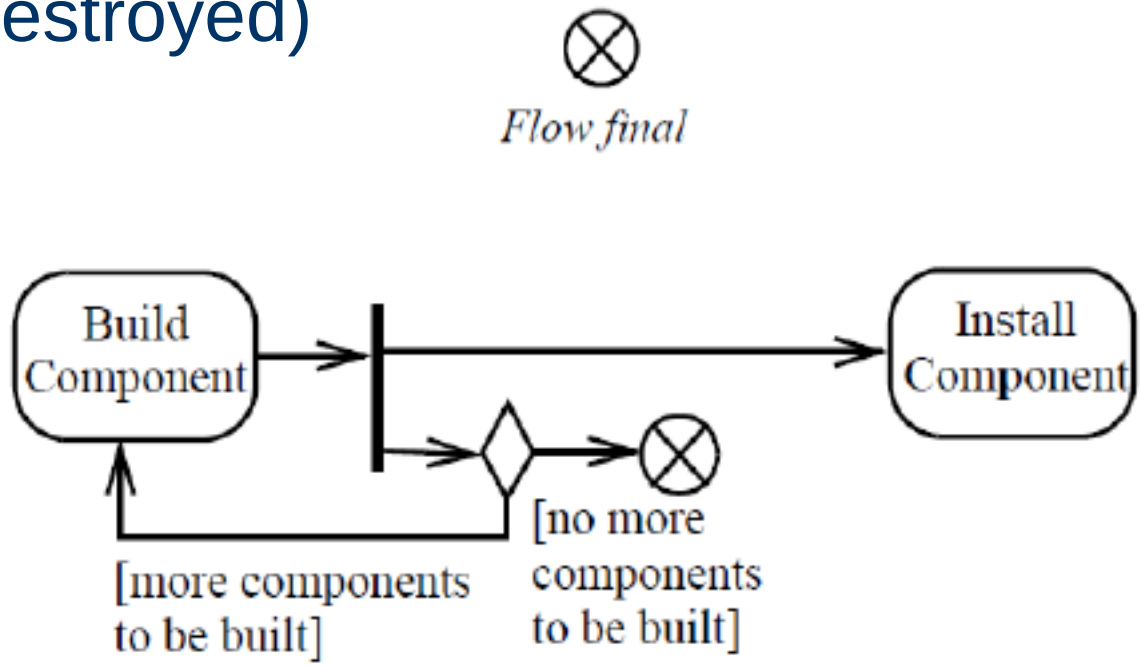
Fork + Join

- Fork node – has one input edge and multiple output edges:
  - When the input edge is enabled, all of the output edges become enabled
- Join node – has many input edges and one output edges:
  - When all of the input edges are enabled, the output edge becomes enabled



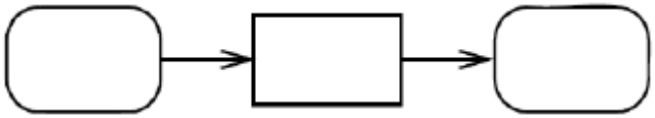
Flow final

- Flow final node – terminates a flow (tokens are destroyed)

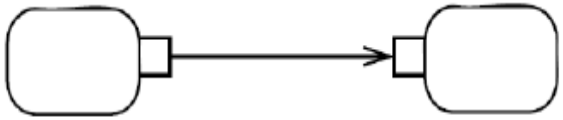


Object nodes and object flows

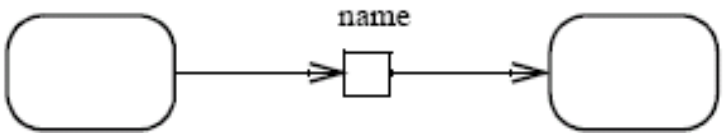
- Object node – a kind of activity node that represents an object that is the input or output of an activity; An object node has a type
- Object flow – a kind of activity edge that represents the flow of values between two action nodes
- If there is only an object flow outgoing from an activity it also transfers control token
- A small triangle near the pin tells it serves for transmitting an exception



Two object flow arrows linking object nodes and actions



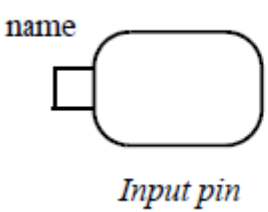
An object flow arrow linking two object node pins.

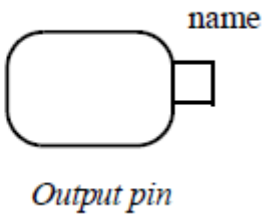


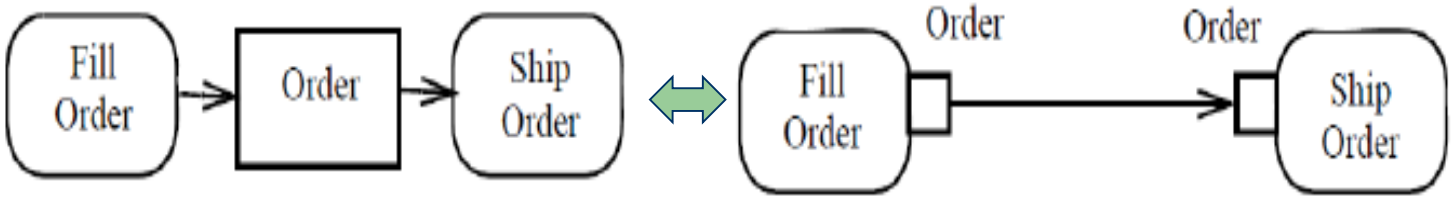
Standalone pin notations:
the output pin and the input pin have the same name and same type

Pins (alternatives for object nodes)

- Pin – is an object node that represents a connection point on an action for input and/or output values; pin can have name and type

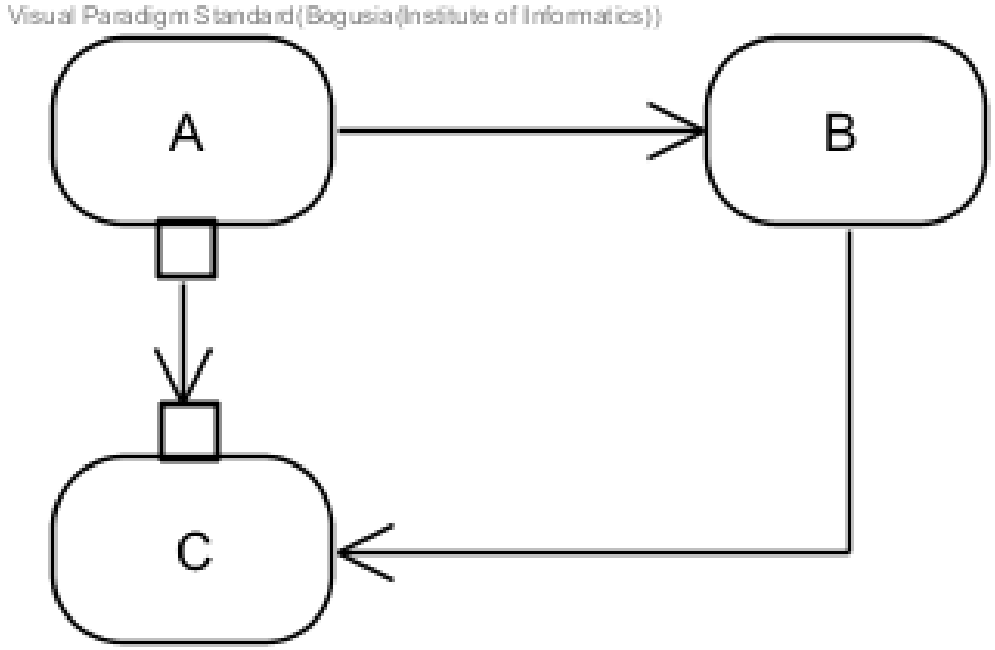






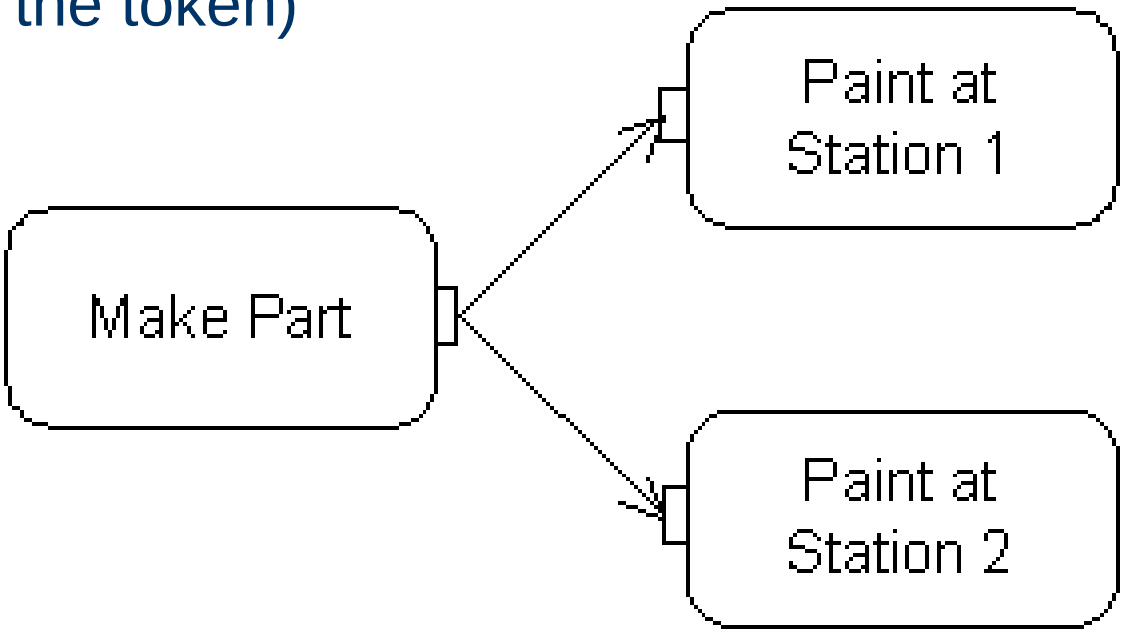
Combination of object and control flows

- The ordering of actions: A, B, C; output data produced by A are the input for C



Race conditions

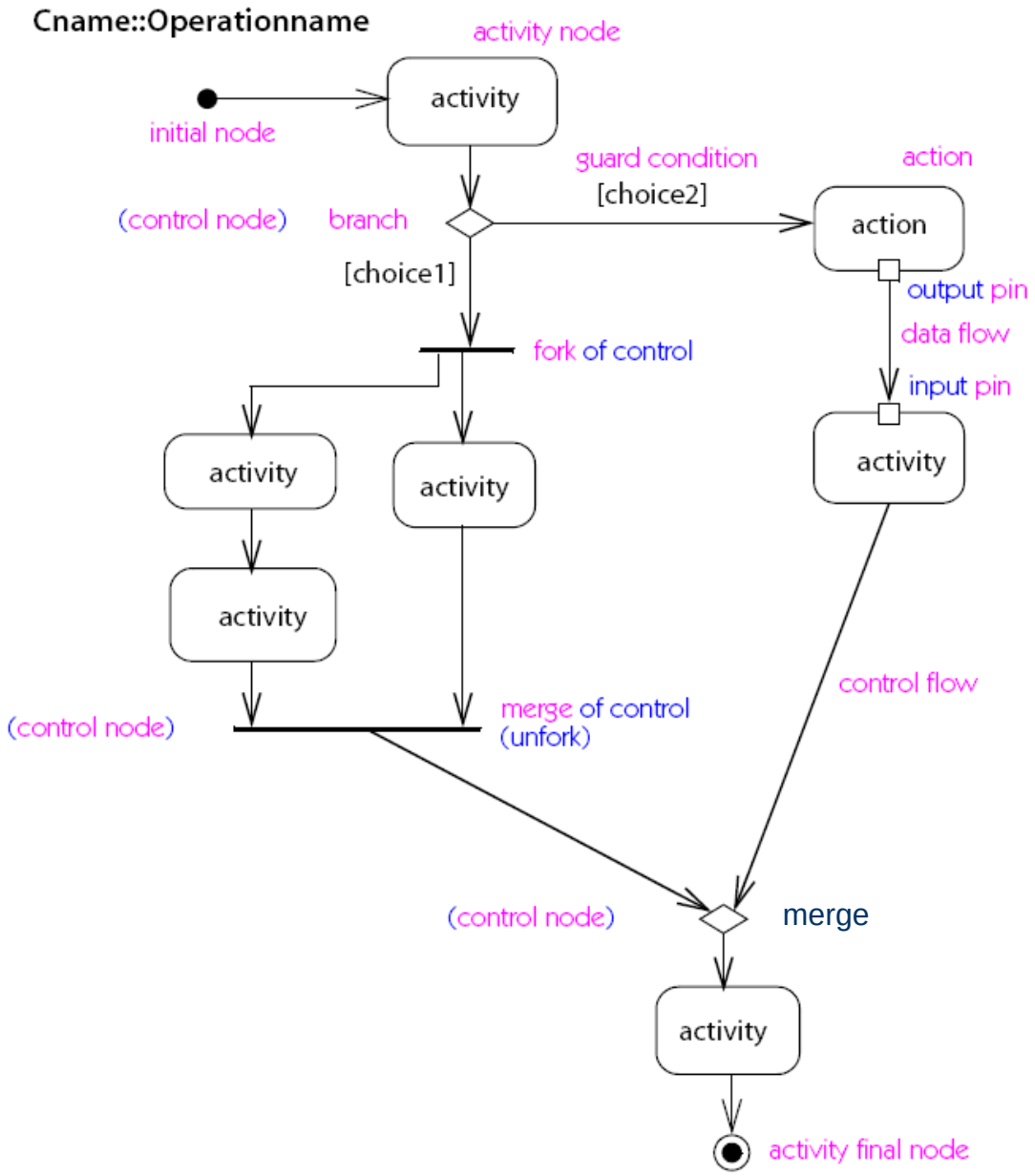
- Object nodes do not duplicate tokens – in consequence a token will be transmitted either to Station 1 or Station 2 (the station must be able to consume the token)



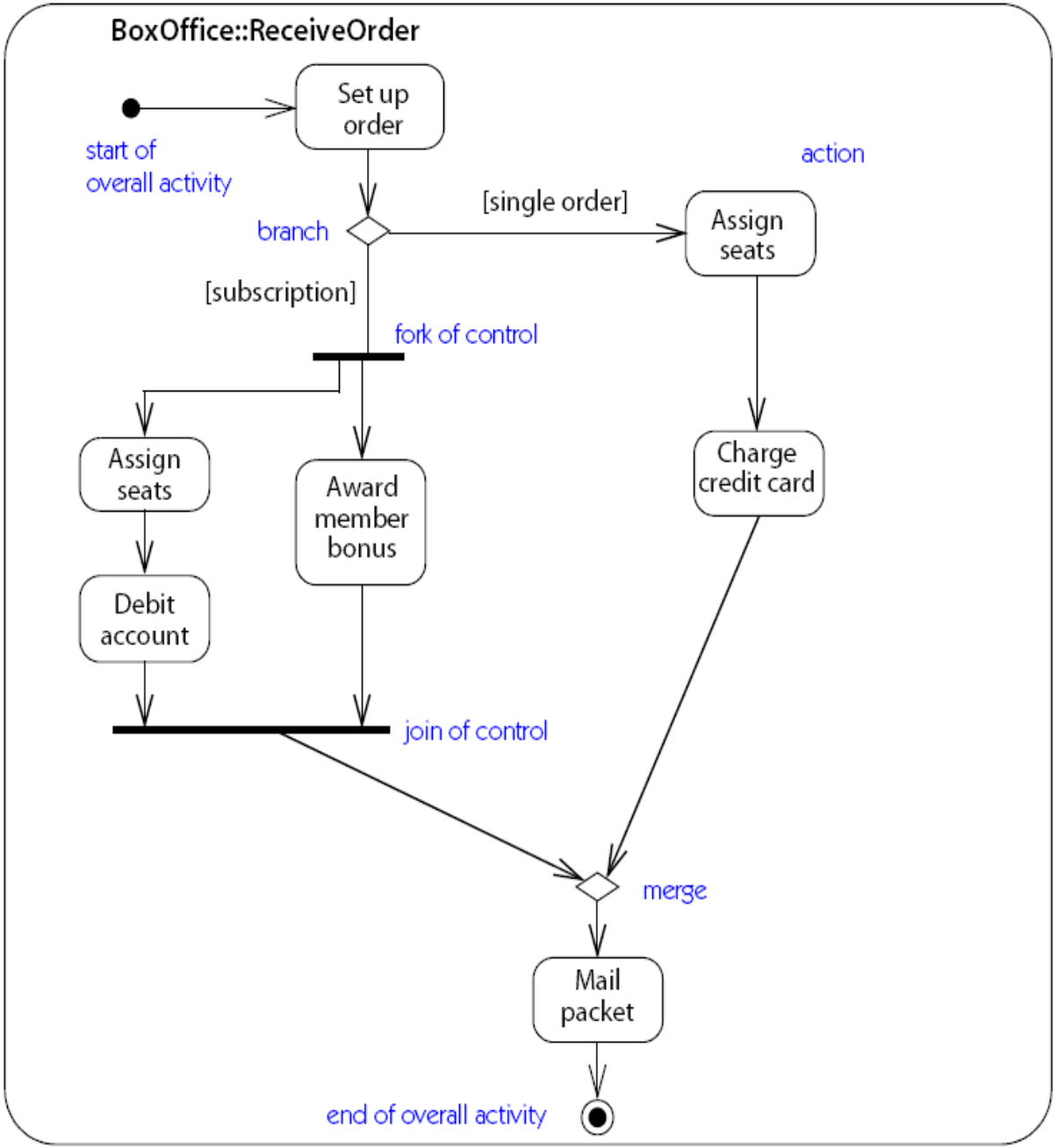
Weighted transitions

- The weight property dictates the minimum number of tokens that must traverse the edge at the same time (default 1)

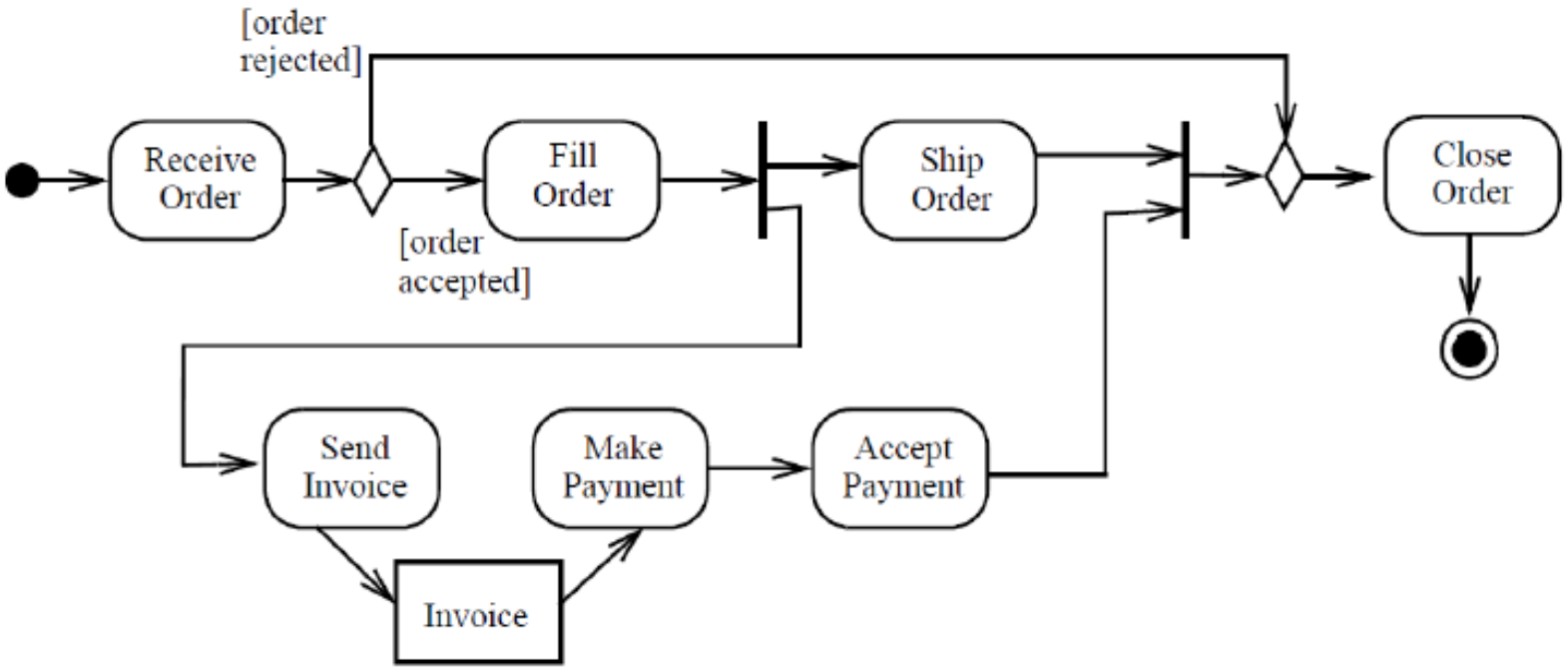
Basic notation summary



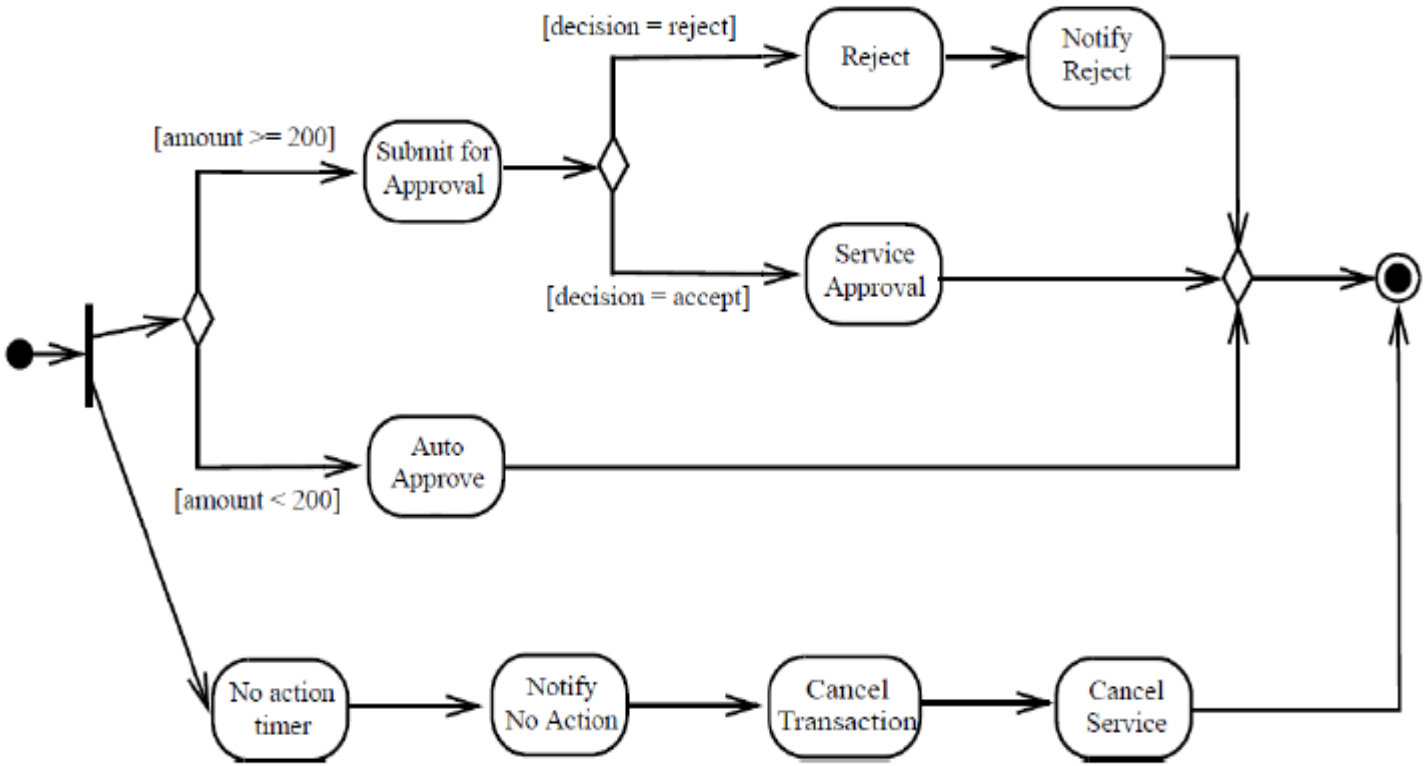
Example


BoxOffice::ReceiveOrder
Set up order
start of overall activity
action
branch
[single order]
Assign seats
[subscription]
fork of control
Assign seats
Award member bonus
Charge credit card
Debit account
join of control
merge
Mail packet
end of overall activity

Example

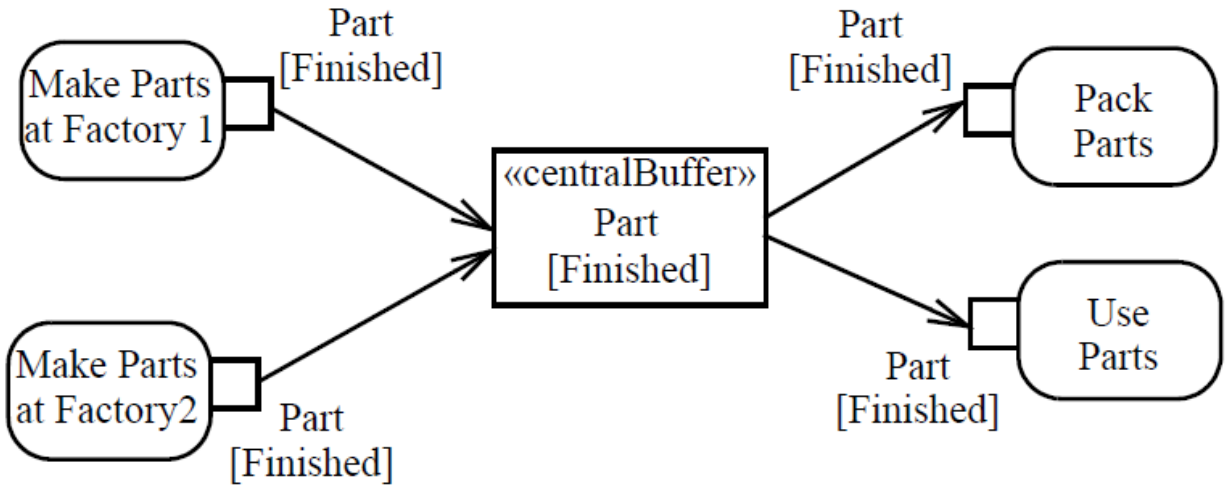

[order rejected]
Receive Order
Fill Order
[order accepted]
Ship Order
Close Order
Send Invoice
Make Payment
Accept Payment
Invoice

Example (racing)



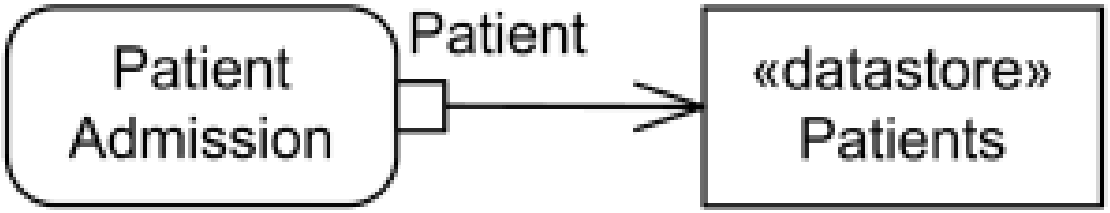
Central buffer

- A **central buffer** node is an **object node** for managing flows from multiple sources and destinations. A central buffer node accepts tokens from multiple object in flows, buffers those and passes them along to out flow (after that object tokens are removed from the buffer)



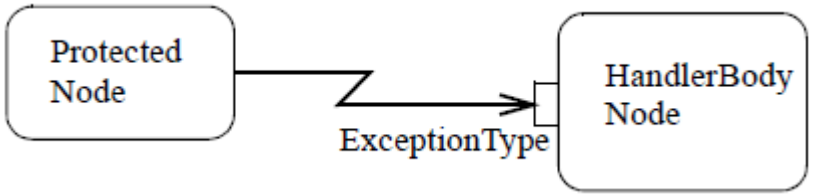
Data store

- A **data store** is a **central buffer** node for non-transient information. The original value is kept in datastore, and its copies are transferred when it is required



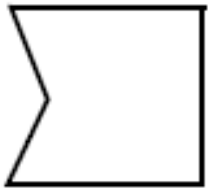
Exception and exception handlers

- Action can raise an exception
- Exception is represented by a „lightning” arrow
- Exception type is placed at the pin of exception handler action (option)



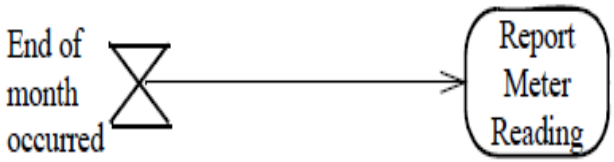
Accept event action

- Accept event actions enable to model how to handle event occurrences detected by the object owning the behavior



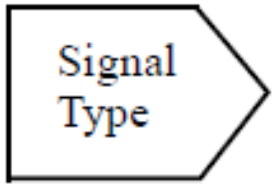
Accept event action

Accept time event action

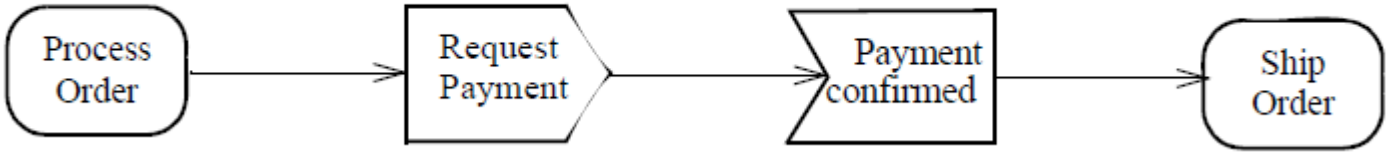


Send signal action

- Send signal action – allows to communicate with another activity diagram

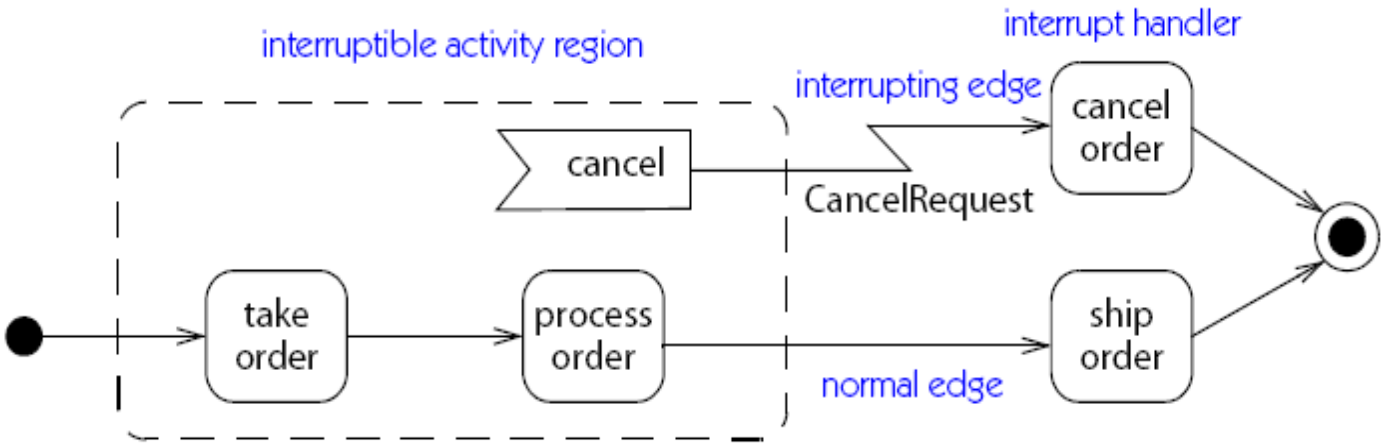


Send signal action



Interruptible activity region

- Interruptible activity region – a set of action nodes and edges within which activity is terminated if a designated event occurs



Interruptible activity region, example

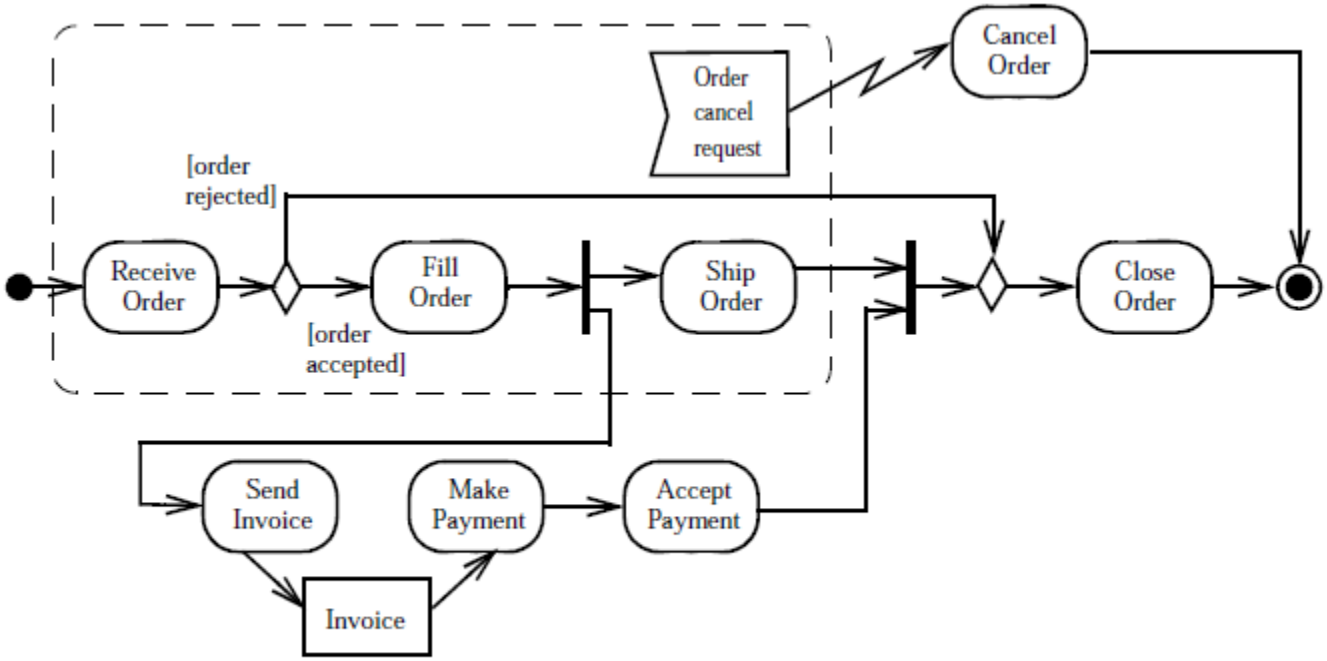
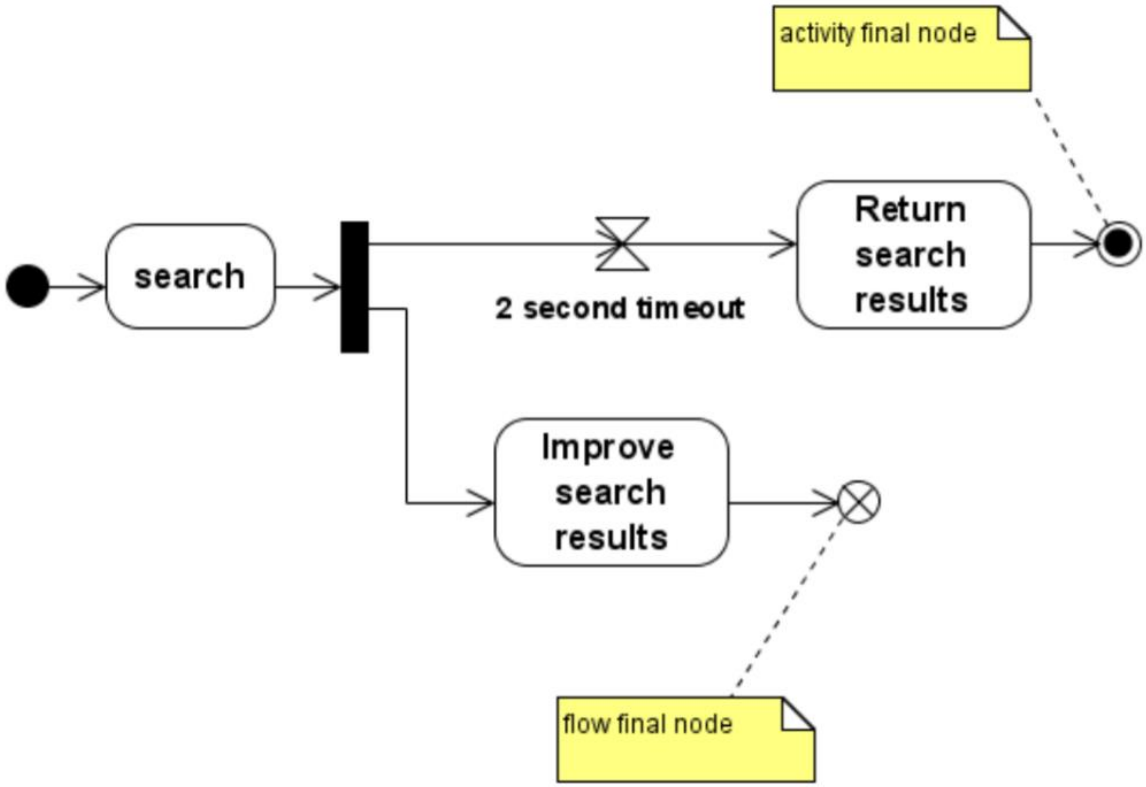
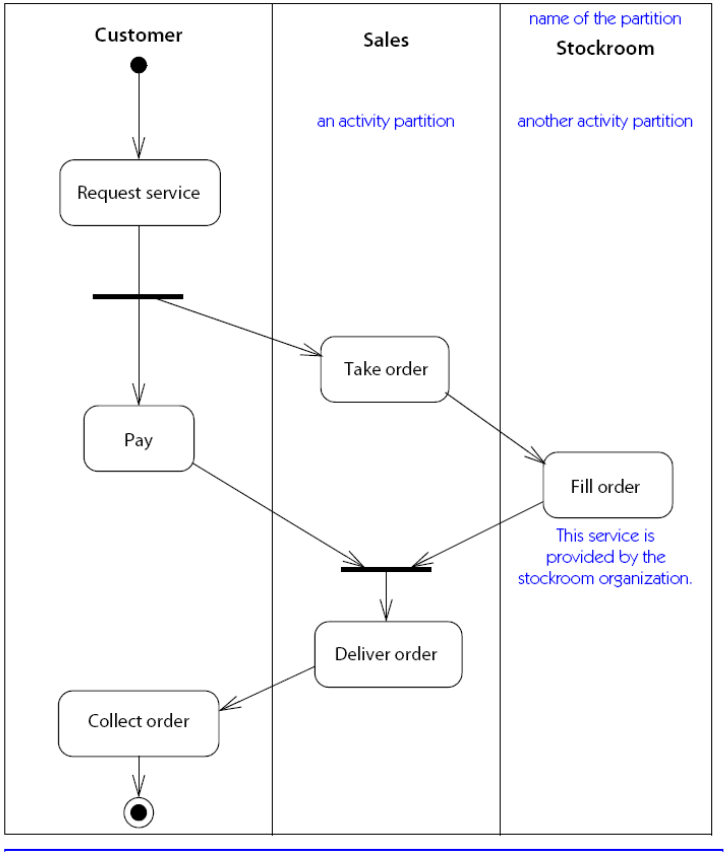


Figure 12.100 - InterruptibleActivityRegion example

Flow final node example

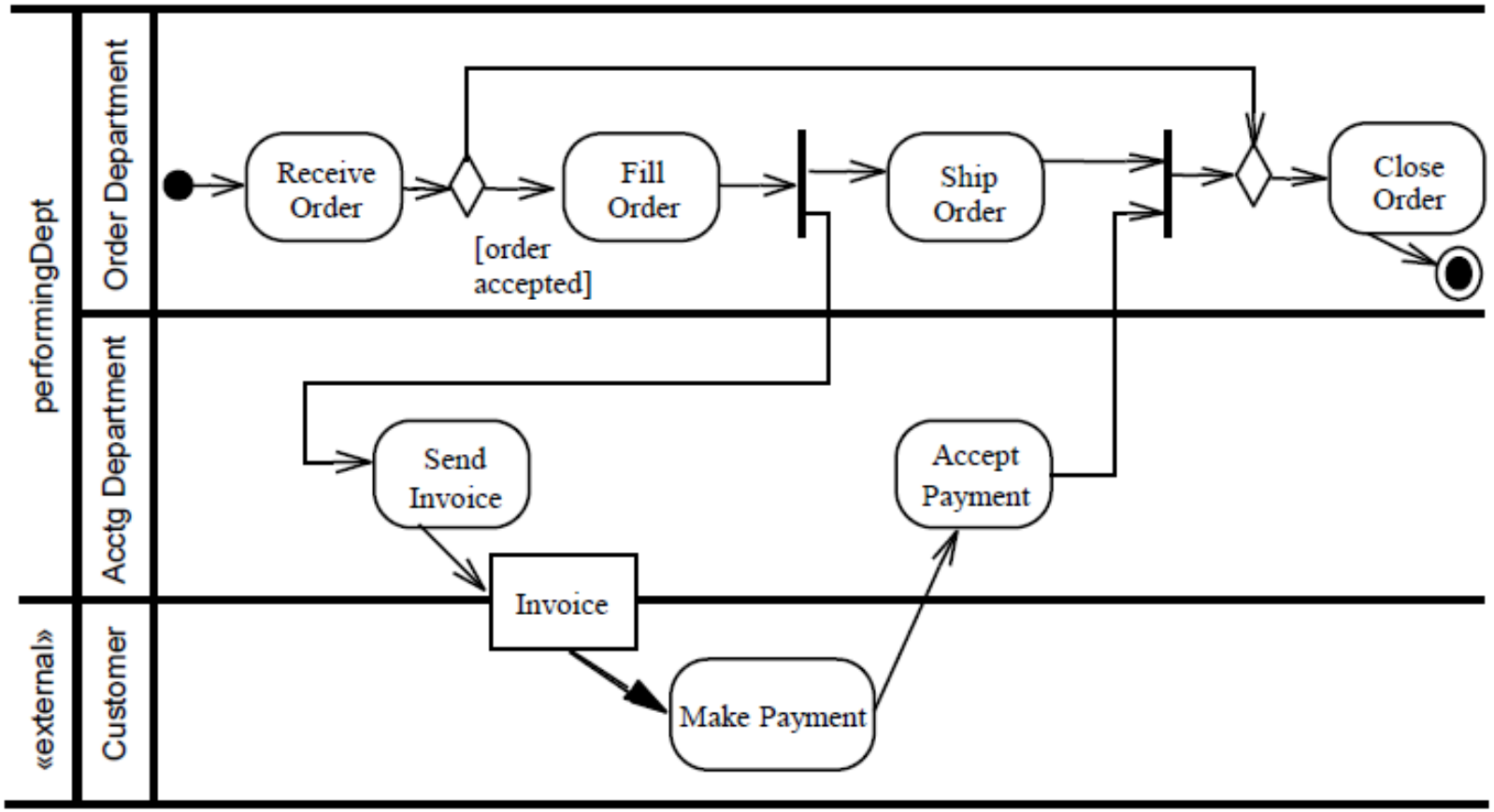


Partitions (swimlanes)



- Partition – element used for organizing the activities in a model according to responsibility

Example


performingDept
Order Department
Acctg Department
«external»
Customer
Receive Order
[order accepted]
Fill Order
Ship Order
Close Order
Send Invoice
Invoice
Make Payment
Accept Payment

Revision

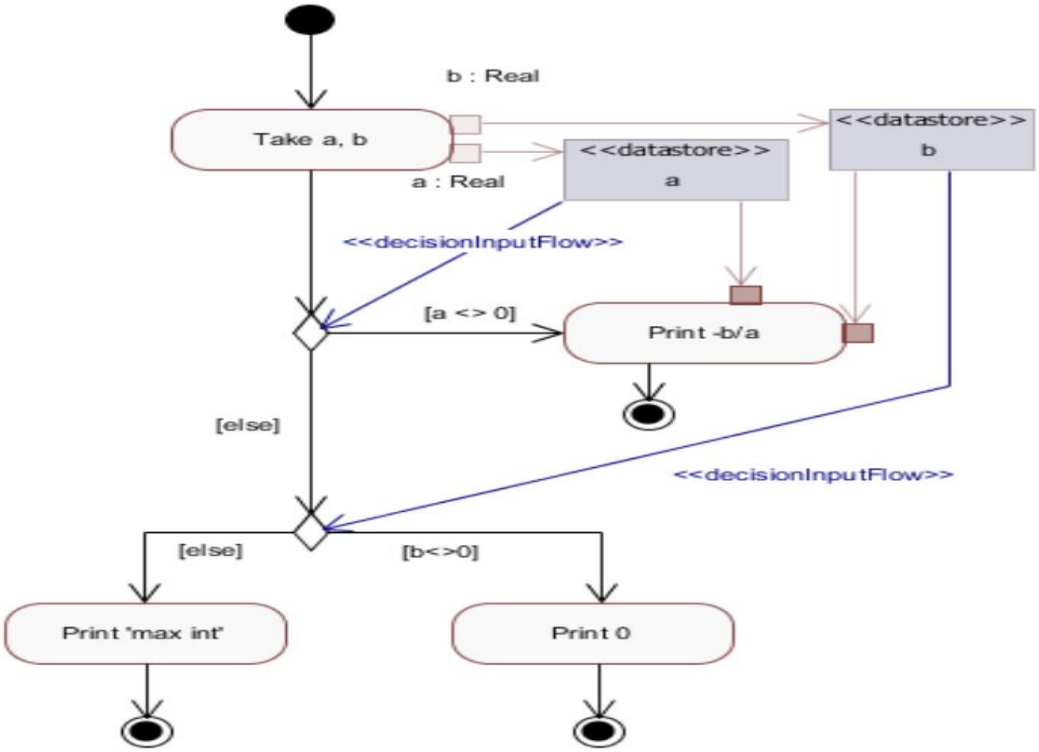
- What can activity diagram be used for?
- What are the basic elements of activity diagram?
- What kind of nodes can be modeled on activity diagrams?

Exercise 1

- Model with activity diagram a procedure of
  - Calculating solution of a linear equation

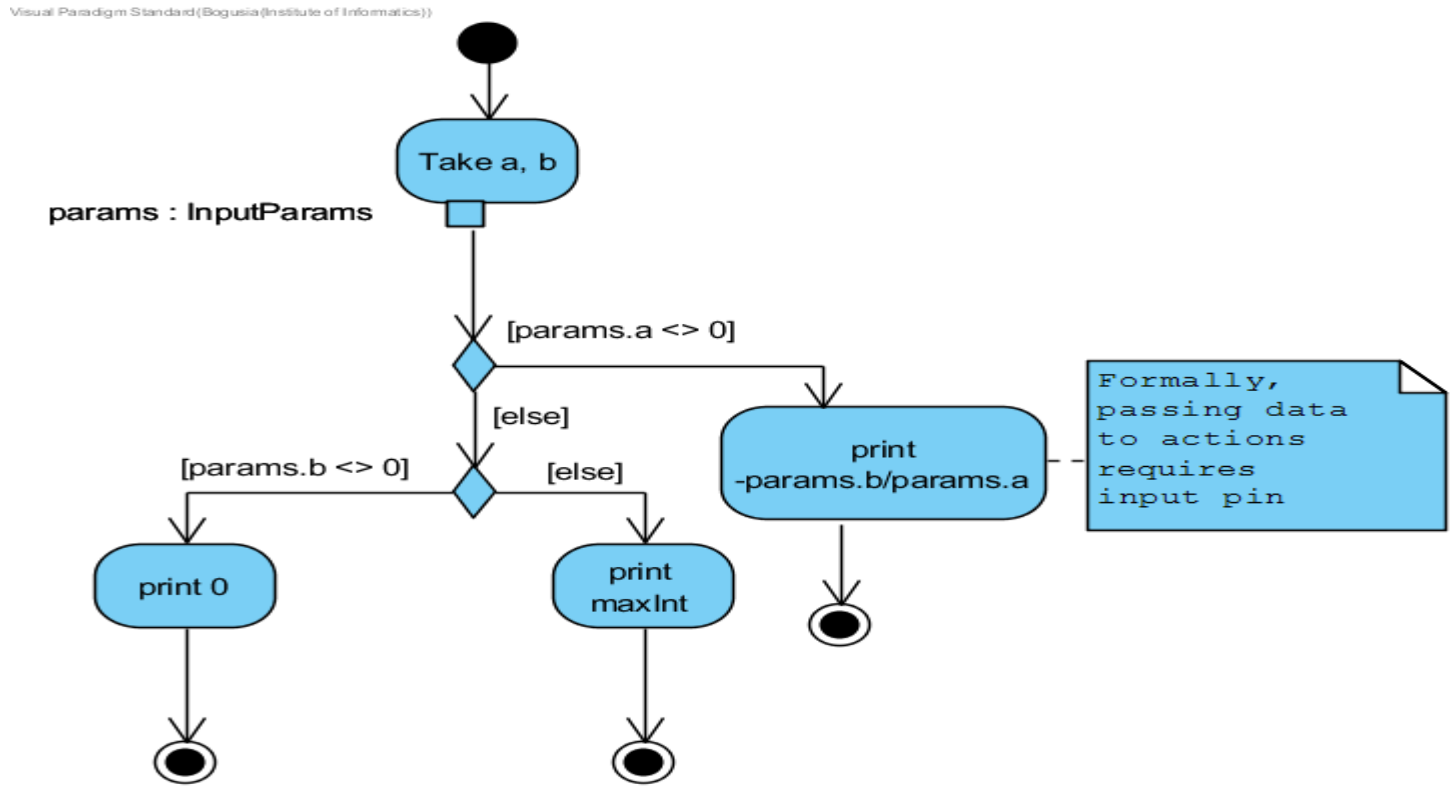
Exercise 1, cont.

- Algorithm perspective



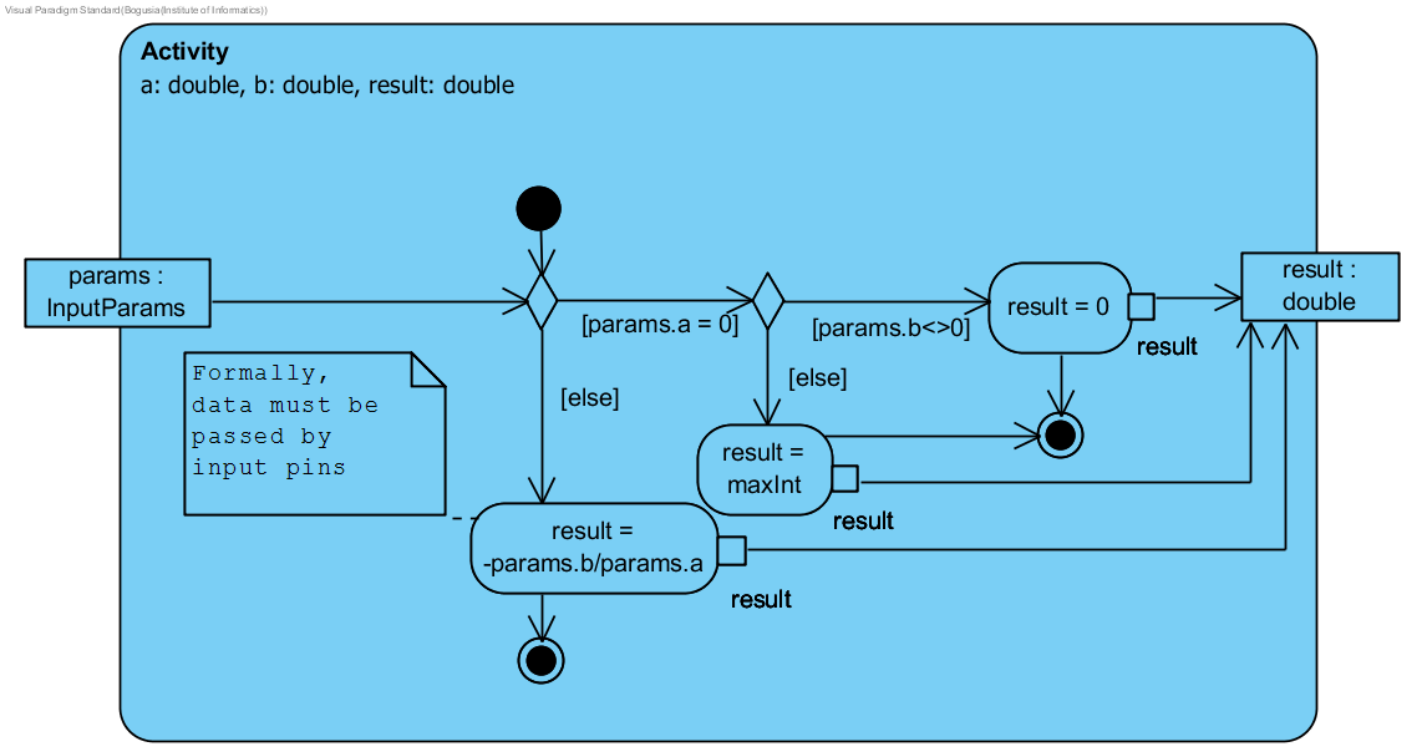
Exercise 1, cont.

- Algorithm perspective



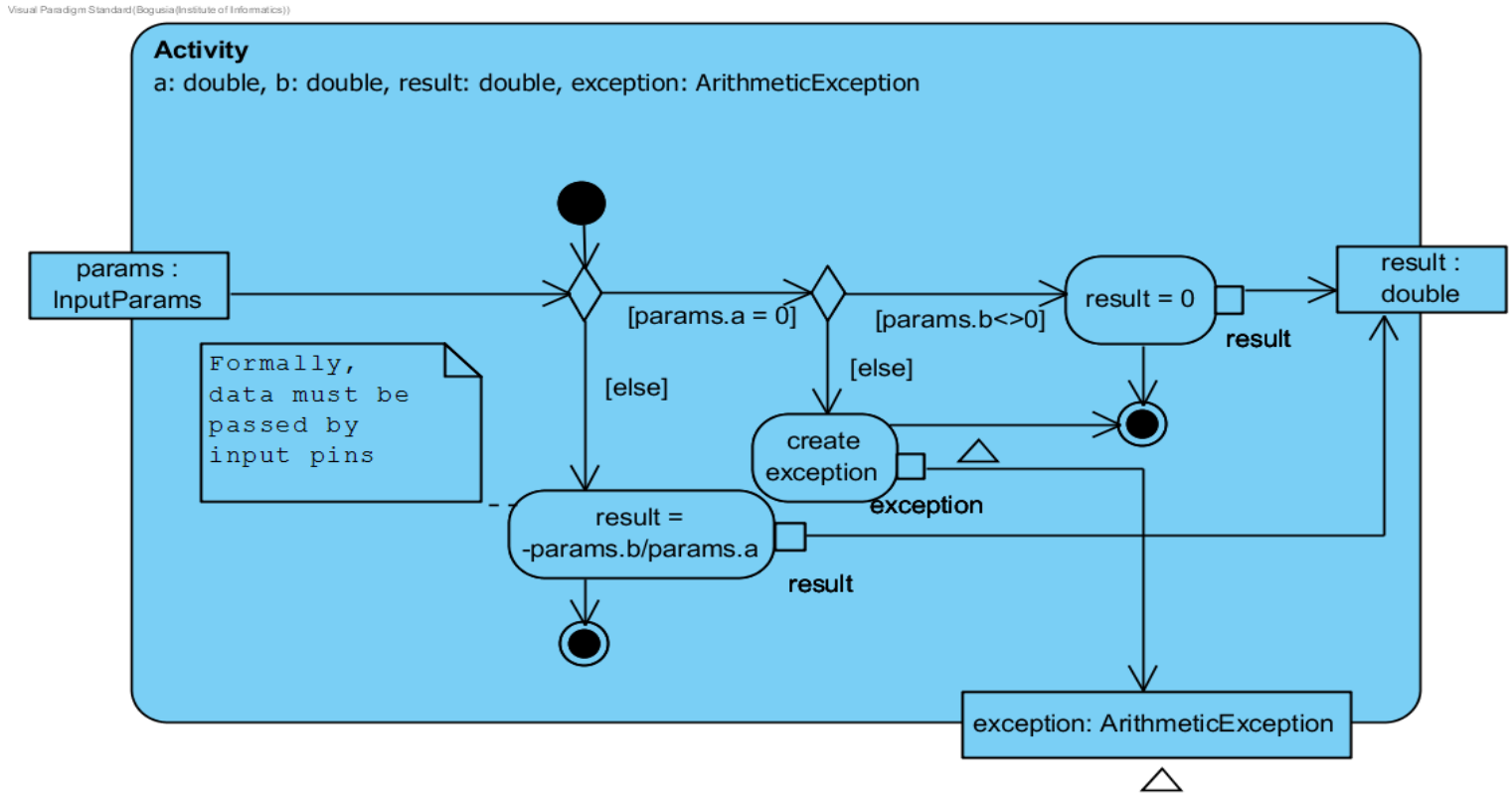
Exercise 2

Function perspective



Exercise 2, cont.

- Function with exception



Exercise 2

- Model the use-case specification with activity diagram. Use partitions and object flows when possible

- UC 001 – Register to the system
- Main flow of actions:
  1. User wants to register to the system.
  2. System asks about *personal data.*
  3.User provides *personal data*.
  4. User asks for registration.
  5. System validates personal data and creates new non-active user account.
  6. System prepares an e-mail and send it to the user by mail server.
  7. User reads the e-mail and activates the account.
  8. System validates that user account exists, activates it and enables login to the system

  Alternative flow 1 - Personal data are invalid

  5a. System validates that personal data are improper. System informs about that the user. Goto step 2.

  Exceptional flow – the non-active account doesn't exists

  8. System validates that account doesn't exists and informs about it the user.