National University of Computer and Emerging Sciences
Lahore Campus

0151

Formal Methods (SE2003)

Final Exam

Date: 19th May 2025

Course Instructor

Dr. Wafa Basit

Total Time (Hrs): 3
Total Marks: 70
Total Questions: 3

Name: Hasan Yahya

22L-7971
Roll No

BSE-6C
Section

Student Signature

Do not write below this line

1) Attempt all the questions on the question paper 2) Do not use lead pencil 3) Draw neat diagrams 4) Make assumptions where necessary

CLO # 3. Apply the knowledge appropriate to the discipline, particularly in the field of formal models

Q1: Choose only one option. Selecting multiple options is not allowed. [10 marks]

1. When a method is overridden in a subclass, what must be true of the precondition?
   a) It must be stronger b) It must be identical (c) It must be weaker or equal
2. Which technique is used to address duplicated code in sibling classes?
   a) Extract Method (b) Pull Up Method c) Push Down Method
3. Which refactoring type is performed frequently and mixed with other tasks?
   a) Root Canal Refactoring (b) Floss Refactoring c) Composite Refactoring
4. ----------are the safety net for the refactoring process.
   a) Assertions b) Contracts (c) Unit tests d) Regression tests
5. Static analysis is extremely useful for finding *specific* types of bugs, such as
   (a) Syntactic Errors b) Logical Errors c) Arithmetic overflows
6. If a method seems more interested in a class other than the class it actually is in – move it to where it belongs.
   a) Divergent Change (b) Feature Envy c) Shotgun Surgery
7. Which one is not an input to the program verifier?
   a) Verification Condition b) code (c) Specification
8. ----------defined preconditons for 23 primitive refactorings.
   (a) Opdyke b)Roberts c)Fowler
9. OCL expressions do not have side effects
   a) Always (b) Never c. Sometimes
10. Petri nets are----------. If more than 1 transition could fire, one is selected arbitrarily.
    a) Mutually Exclusive (b) Nondeterministic c) Deterministic

*CLO # 2: Recognize the nature of **software system, suitable model. [C2 – Analysis]*** [5+5 Marks]

Q2: Design a **Z schema** for a simple **library management system** that tracks books and borrowers. The schema should include:

7+14.5+30.5+5.5 = 57.5/70 Excellent

National University of Computer and Emerging Sciences

Lahore Campus

[BOOK, BORROWER]

Library

books: ℙ BOOK
borrowers: ℙ BORROWER
loans: BOOK ⇸ BORROWER

dom loans ⊆ books
ran loans ⊆ borrowers
∀ b: BOOK • b ∈ dom loans ⇒ loans(b) ∈ borrowers

14.5/36

1. **Borrow book operation requires that the borrower is registered, the book is not already loaned and it is in the library. Also, no borrower can borrow more than three books at a time.**
Hint: |loans ▷ {br?}| would return the number of books borrowed by any borrower, br?.

BorrowBook

ΔLibrary
bk?: BOOK
br?: BORROWER

loans' = loans ∪ {bk? ↦ br?}
br? ∈ borrowers
bk? ∉ dom loans
bk? ∈ books
dom loans ⊆ books
ran loans ⊆ borrowers

∀ b: Book • b ∈ dom loans ⇒ loans(b) ∈ borrowers.
|loans ▷ {br?}| < 3
|loans ▷ {br?}|' = |loans ▷ {br?}| + 1

2. **Return book operation requires that the borrower is registered, the book is already loaned by the returning borrower and it is not in the library. Write all post-conditions as well.**

ReturnBook

ΔLibrary
bk?: BOOK
br? BORROWER

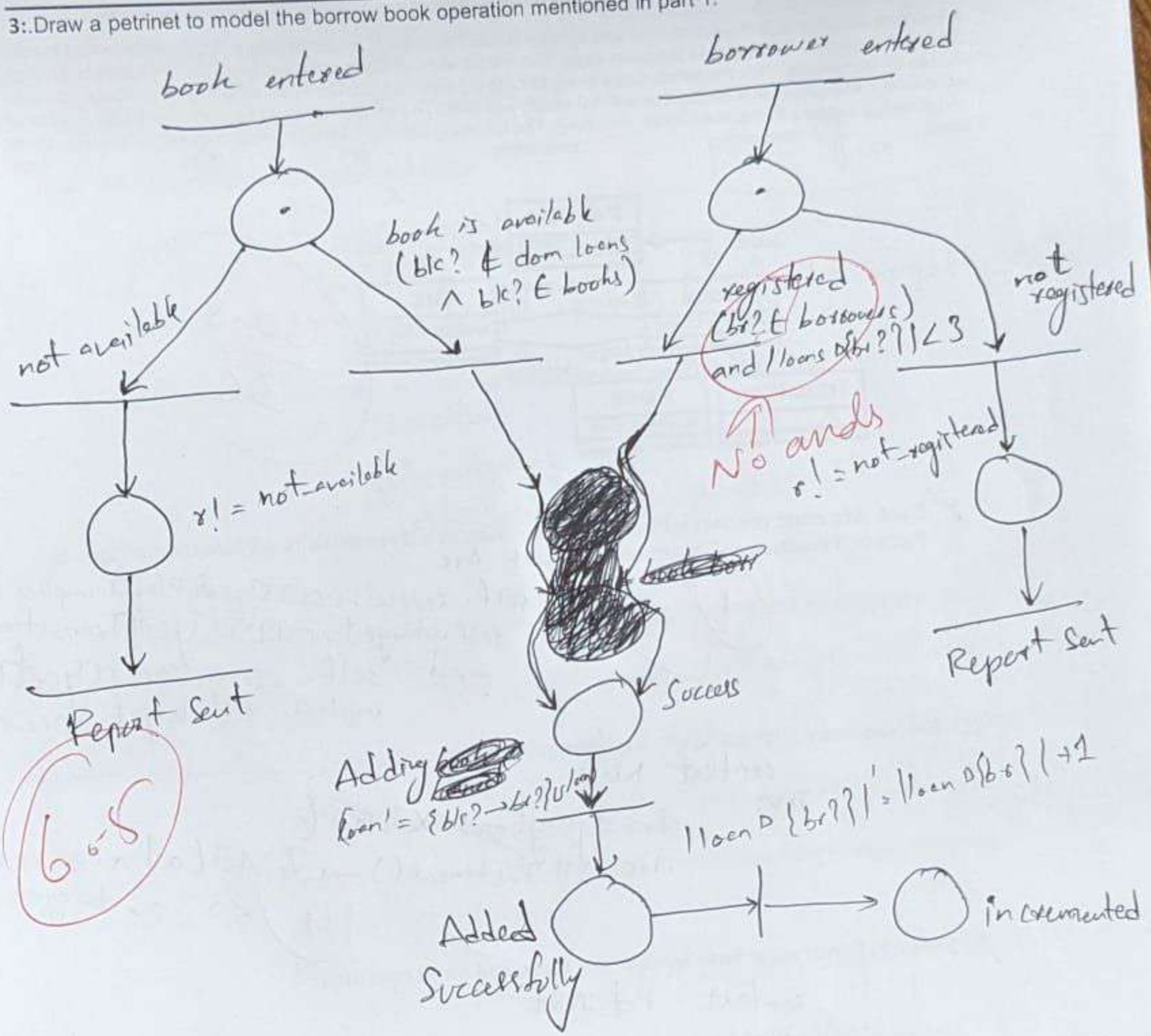
loans' = loans ⩤ {bk?}
br? ∈ borrowers
bk? ∈ dom loans
bk? ∉ books
dom loans ⊆ books
ran loans ⊆ borrowers

∀ b: Book • b ∈ dom loans ⇒ loans(b) ∈ borrowers
loans(bk?) = br?
|loans ▷ {br?}|' = |loans ▷ {br?}| − 1
|loans ▷ {br?}| < 3

Report! = not_available | not_registered | okay

CLO # 3. Apply the knowledge appropriate to the discipline, particularly in the field of formal models. [C3 – Application) ..... [8 marks]

3:.Draw a petrinet to model the borrow book operation mentioned in part 1.



Assuming for Errors

Report ::= not_available | not_registered | okay

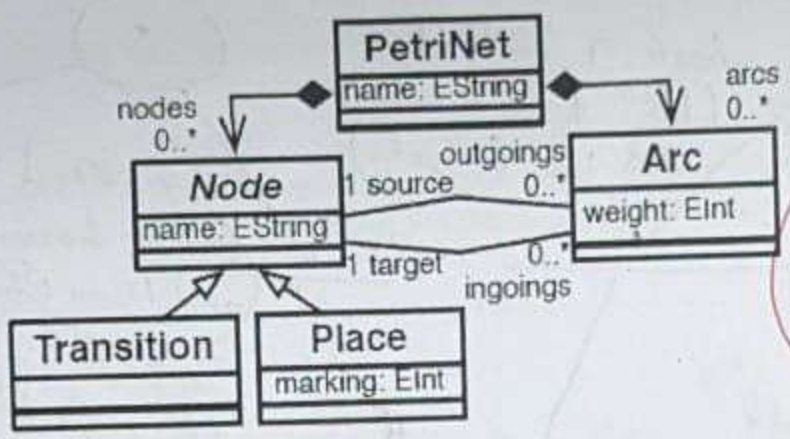
If we have no existe Report then we will write Error = sending & Error instead of report

015

CLO #2: Recognize the nature of software system, suitable model. [C2 – Analysis]

Q3 (a): Define OCL constraints for the following UML Class Diagram [36 marks, 4 Marks each]

A PetriNet is composed of several Arcs and several Nodes. Arcs have a source and a target Node, while Nodes can have several in-coming and outgoing Arcs. The model distinguishes between two different types of Nodes: Places or Transitions. This model captures every necessary concept to build Petri nets. However, there can also exist valid instances of this model that are not valid Petri nets. The model needs to be enhanced with additional properties to capture the domain more precisely. The following well-formedness rules, expressed in OCL, show some mandatory properties of Petri nets.



30-5 / 36

1. **Each Arc must connect a Place to a Transition or a Transition to a Place** (never Place-to-Place or Transition-to-Transition): context Arc
inv: self.source.OCLIsTypeOf(Place) implies self.target.OCLIsTypeOf(Transition) and self.source.OCLIsTypeOf(Transition) implies self.target.OCLIsTypeOf(Place)

2. **No Node may have an Arc to itself (no self-loops):**
context Node
inv: ~~self.outgoings->exclude(~~ Arc.allInstances() -> forAll(a | a.source <> a.target)

3. **Every PetriNet must have at least one Place and one Transition:**
context PetriNet
inv: self.nodes->select(n | n.OCLIsTypeOf(Transition))->size() >= 1 and self.nodes->select(n | n.OCLIsTypeOf(Place))->size() >= 1

~~and self.nodes->size()~~

2 Recovery if at minimum 1 place you have 1 transition 2 nodes. you will have

4. **All Places must have non-negative markings:**

Both are ok

context Place
inv: Place.allInstances() → forAll(p | p.marking ≥ 0)

context Place
inv: self.marking ≥ 0

4

assuming 0 is not considered positive

5. **All Arcs must have a positive weight:**

Both are ok

context Arc
inv: self.weight ≥ 1

context Arc
inv: Arc.allInstances() → forAll(a | a.weight ≥ 1)

4

6. **Each Transition must have at least one incoming and one outgoing Arc:**

context Transition
inv:
self.outgoings → size() ≥ 1 ∧
self.ingoings → size() ≥ 1

4

attributes

7. **A PetriNet cannot have multiple Arcs between the same source and target nodes (no duplicates):**

Assuming ~~Multi-directional~~ arcs allowed between A → B is not same as B → A

context Petrinet
inv: self.arcs → ISUnique(self.arcs.source.name)
and self.arcs → ISUnique(self.arcs.target.name).

Not an

0.5

8. **All nodes in a PetriNet must be connected (i.e., have at least one incoming or outgoing Arc):**

context Petrinet
inv:
self.nodes → forAll(n | n.ingoing → size() ≥ 1
∨ n.outgoings → size() ≥ 1)

2

∨ or are both used for or.

0151

8. **A Transition cannot directly connect to another Transition (enforced at the PetriNet level):**

context ~~Transition~~ Petrinet
inv:
~~self.arcs->forAll(a | a.oclIsTypeOf(Transition).source.oclIsTypeOf(Transition) = false~~
self.arcs → forAll(a | a.oclIsTypeOf(Transition) implies a.target.oclIsTypeOf(Transition) = false and a.source.oclIsTypeOf(Transition) = false)

4

Q3 (b): "If a node is a `Place`, and its marking is 0, then it should not have any outgoing arcs." Following is the OCL specification of the above mentioned constraint identify the bad smell (2 marks) and remove it (4 points)

```
context Node
inv PlaceWithZeroMarkingShouldNotHaveOutgoings:
  self.oclIsTypeOf(Place) implies
    self.oclAsType(Place).marking = 0 implies
      Arc.allInstances()->forAll(a | a.source <> self)
```

Bad Smell
= Type Based Conditionals
↳ Also more then one implies suggests implies chain.

1.5

After removing bad smell;

context ~~Node~~ Place
inv: PlaceWithZeroMarking ShouldNotHaveOutgoings:
self.marking = 0 implies
Arc.allInstances() → forAll(a | a.source <> self)

4

5/6

~~context Petrinet~~
~~inv: self.arcs->forAll(a | a.~~
~~source.oclIsTypeOf(Transition)~~
~~implies a.target.source.~~

------Good Luck----