4.4 Interfaces and polymorphism

Name: _____ Class: _____ Date: _____ **Total: 24 marks**

KEY WORDS polymorphism • dynamic dispatch

Objective

Program against an interface, trace dynamic dispatch, and keep substitutes honest.

1 Key knowledge and worked examples

■ The contract

A `Notifier` protocol names one method, `send`. Any object with that method fits – duck typing; the protocol writes the contract down ✓. Substitution: preconditions only widen, guarantees only grow ✓.

■ Worked example: one loop, three senders

`team = [EmailNotify(), SmsNotify(), QueueNotify()]`; one loop prints `email: loan due`, `sms: loan due`, `queue: loan due` ✓. A fourth sender joins with no change to the loop ✓.

2 Practice

2.1 A variable holds an `SmsNotify` and the call `send` runs. State which method executes and what chose it. [3]

2.2 A fourth sender class is added. State what changes in the existing loop body. [3]

2.3 Classify each move as widening or narrowing: accepting an empty string; raising on an empty string. [3]

3 Exam-style questions

3.1 A reminder system holds a list of notifiers and runs one loop: `for n in team: print(n.send('loan due'))`. The list holds an email sender returning `email: loan due`, an SMS sender returning `sms: loan due`, and a queue sender returning `queue: loan due`.

Write the three lines the loop prints. Explain, in order, what the variable names, what the object carries, and what dispatch picks – naming the rule that decides. State what edits the call site needs when a fourth sender joins the list. [8]

3.2 A substitute `SilentNotify.send` raises when the text is empty, while the interface promises that send never raises and always returns a string.

Name the rule the substitute breaks and the direction preconditions may legally move. State what a caller relying on the promise now suffers. Write the contract fix – what the substitute must accept and still return – and name what an `isinstance` chain should be replaced with. [7]