```
DEFINE services AS DICTIONARY:
    1 → ("Dog Grooming", 49.99)
    2 → ("Cat Grooming", 39.99)
    3 → ("Pet Bathing", 29.99)
    4 → ("Pet Nail Clipping", 19.99)
    5 → ("Pet Sitting", 24.99)
    6 → ("Pet Walking", 14.99)
    7 → ("Pet Training", 59.99)
    8 → ("Pet Boarding", 89.99)
    9 → ("Pet Transportation", 39.99)
    10 → ("Pet Photography", 99.99)

FUNCTION display_services()
    PRINT "Available Services:"
    FOR EACH service_id, (name, price) IN services
        PRINT service_id + ". " + name + " - $" + price
END FUNCTION

FUNCTION get_order() RETURNS LIST
    INITIALIZE order AS EMPTY LIST

    LOOP UNTIL user inputs 'done'
        PROMPT "Enter service number (or type 'done'): " → input
        IF input IS 'done'
            BREAK
        TRY
            CONVERT input TO INTEGER → choice
            IF choice IN services
                ADD choice TO order
            PRINT "Added: " + services[choice].name
          ELSE
            PRINT "Invalid selection"
        CATCH error
          PRINT "Please enter a valid number or 'done'"
    END LOOP

    RETURN order
END FUNCTION

FUNCTION summarize_order(order)
    IF order IS EMPTY
        PRINT "No services selected."
        RETURN

    SET total TO 0
    PRINT "Order Summary:"

    FOR EACH service_id IN order
        GET name, price FROM services[service_id]
        PRINT "- " + name + ": $" + price
        ADD price TO total

    PRINT "Total: $" + total
END FUNCTION

CLASS Customer
    FUNCTION __init__(name)
        SET self.name TO name
        SET self.order TO EMPTY LIST
    END FUNCTION

    FUNCTION add_service(service_id)
        IF service_id IN services
            ADD service_id TO self.order
            PRINT "Added: " + services[service_id].name
        ELSE
            PRINT "Service ID not valid."
    END FUNCTION

    FUNCTION get_total() RETURNS FLOAT
        RETURN SUM OF services[id].price FOR EACH id IN self.order
    END FUNCTION
END CLASS

CLASS PremiumCustomer EXTENDS Customer
    FUNCTION apply_discount() RETURNS FLOAT
        RETURN self.get_total() * 0.9  // 10% discount applied
    END FUNCTION
END CLASS

CLASS OrderSummary
    FUNCTION __init__(customer)
        SET self.customer TO customer
    END FUNCTION

    FUNCTION display()
        PRINT "Order Summary for " + customer.name
        IF customer.order IS EMPTY
            PRINT "No services selected."
            RETURN

        SET total TO 0
        FOR EACH service_id IN customer.order
            GET name, price FROM services[service_id]
            PRINT "- " + name + ": $" + price
            ADD price TO total
        END FOR

        PRINT "Total: $" + total
    END FUNCTION
END CLASS

FUNCTION get_customer() RETURNS Customer
    PROMPT "Enter your name:" → name
    PROMPT "Are you a premium customer? (yes/no):" → type

    IF type IS 'yes'
        RETURN NEW PremiumCustomer(name)
    ELSE
        RETURN NEW Customer(name)
END FUNCTION

FUNCTION get_order(customer)
    LOOP UNTIL user inputs 'done'
        PROMPT "Enter service number (or type 'done'): " → input
        IF input IS 'done'
```

```
            BREAK
        TRY
            CONVERT input TO INTEGER → choice
            CALL customer.add_service(choice)
        CATCH error
            PRINT "Invalid input."
    END LOOP
END FUNCTION

FUNCTION main()
    PRINT "Welcome to the Mobile Pet Spa"

    CALL display_services()

    SET customer TO get_customer()

    CALL get_order(customer)

    SET summary TO NEW OrderSummary(customer)
    CALL summary.display()

    IF customer IS INSTANCE OF PremiumCustomer
        SET discounted_total TO customer.apply_discount()
        PRINT "Premium Discount Applied! New Total: $" + discounted_total
END FUNCTION
```