Java Programming II

Lab6

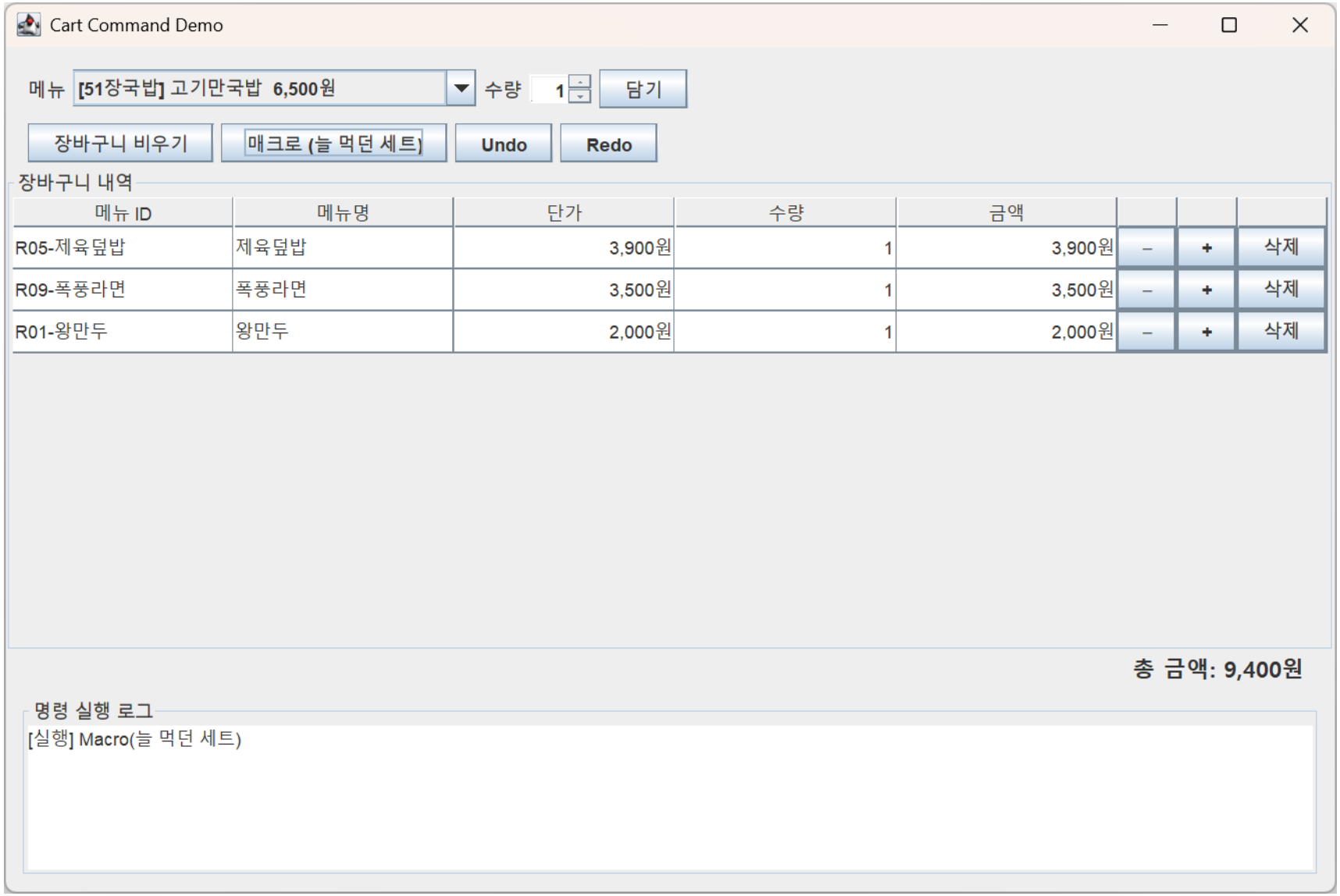
514770-1
Fall 2026
10/27/2026

Kyoung Shin Park
Computer Engineering
Dankook University

Lab6

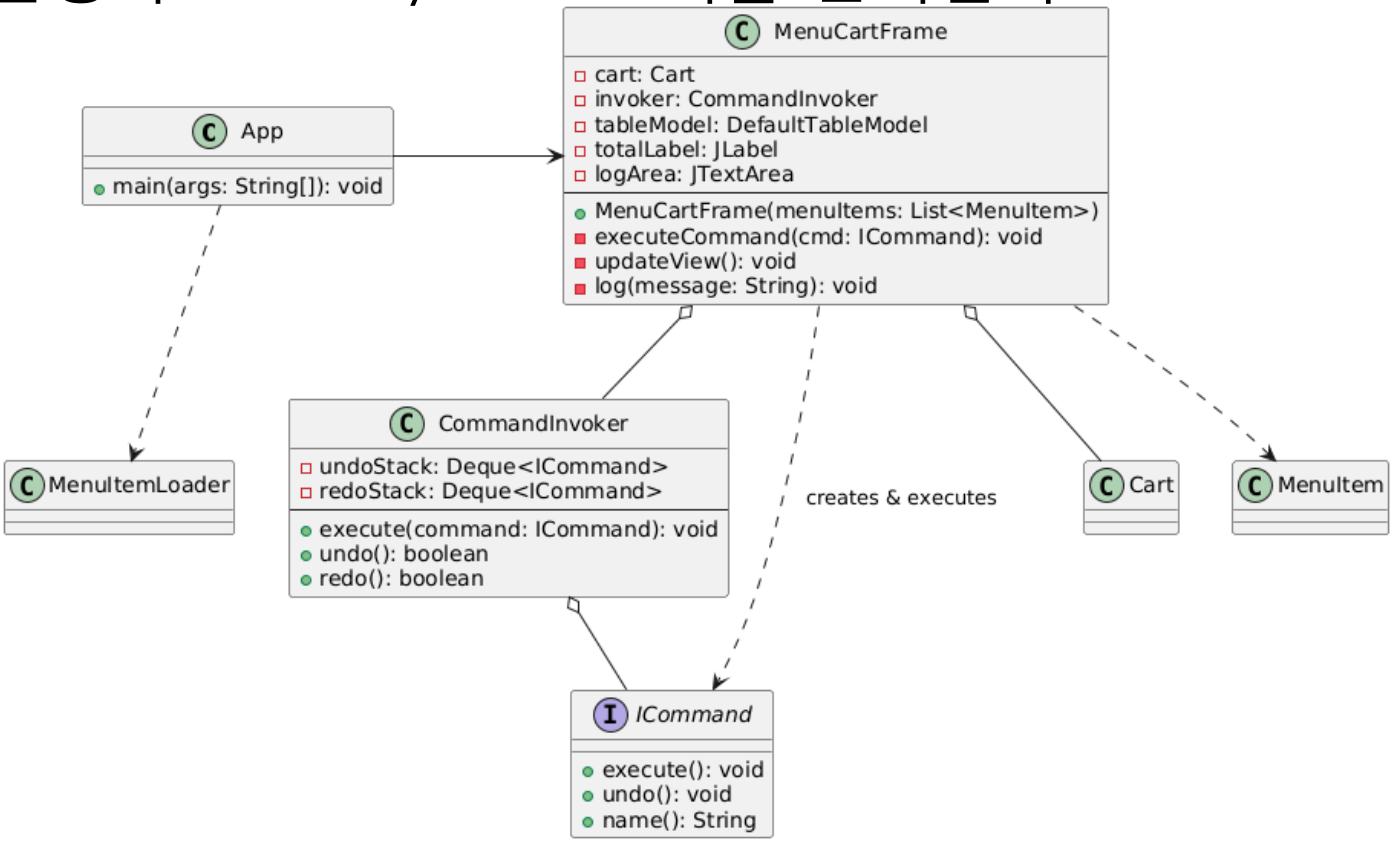
- **Command 패턴**을 사용하여 학생식당 메뉴 **장바구니** Swing 프로그램을 작성하라.
  - **ICommand** 인터페이스는 **execute(), undo(), String name()** 메서드를 가진다
  - **AddItemCommand** (담기) **ChangeQuantityCommand** (수량 변경) **RemoveItemCommand** (삭제) **ClearCartCommand** (비우기)는 ICommand를 구현한다.
  - 이 프로그램은 **Undo / Redo / Macro(세트 주문)** 기능을 갖춘다.
  - 장바구니의 모든 변경은 **반드시 Command를 통해서만** 이루어진다.
    - Cart를 직접 수정하면 Undo 히스토리와 실제 장바구니 상태가 어긋난다.
  - Undo는 수량뿐 아니라 **담긴 순서(위치)까지** 원래대로 되돌려야 한다.
  - 매크로는 **전부 또는 전무(all-or-nothing)**로 실행된다.
    - 중간에 실패하면 이미 실행한 명령을 역순으로 undo하여 실행 전 상태로 되돌린다.

Lab6


Cart Command Demo
메뉴
[51장국밥] 고기만국밥 6,500원
수량
1
담기
장바구니 비우기
매크로 (늘 먹던 세트)
Undo
Redo
장바구니 내역
메뉴 ID
메뉴명
단가
수량
금액
R05-제육덮밥
제육덮밥
3,900원
1
3,900원
삭제
R09-폭풍라면
폭풍라면
3,500원
1
3,500원
삭제
R01-왕만두
왕만두
2,000원
1
2,000원
삭제
총 금액: 9,400원
명령 실행 로그
[실행] Macro(늘 먹던 세트)

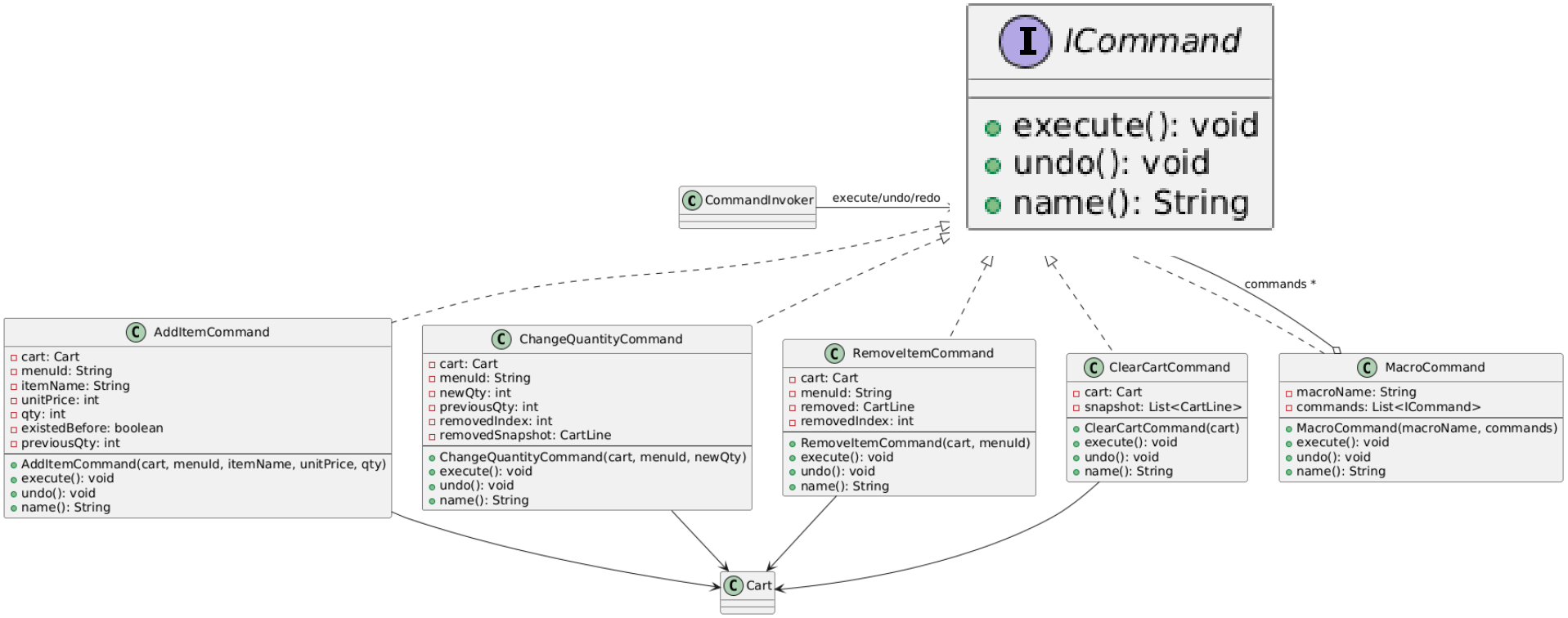
Lab6

- App은 메뉴 목록을 읽어 MenuCartFrame을 호출하고, MenuCartFrame은 Command 객체를 생성하여 CommandInvoker에 전달한다. CommandInvoker는 명령을 실행하고 undo/redo 스택을 관리한다.



Lab6

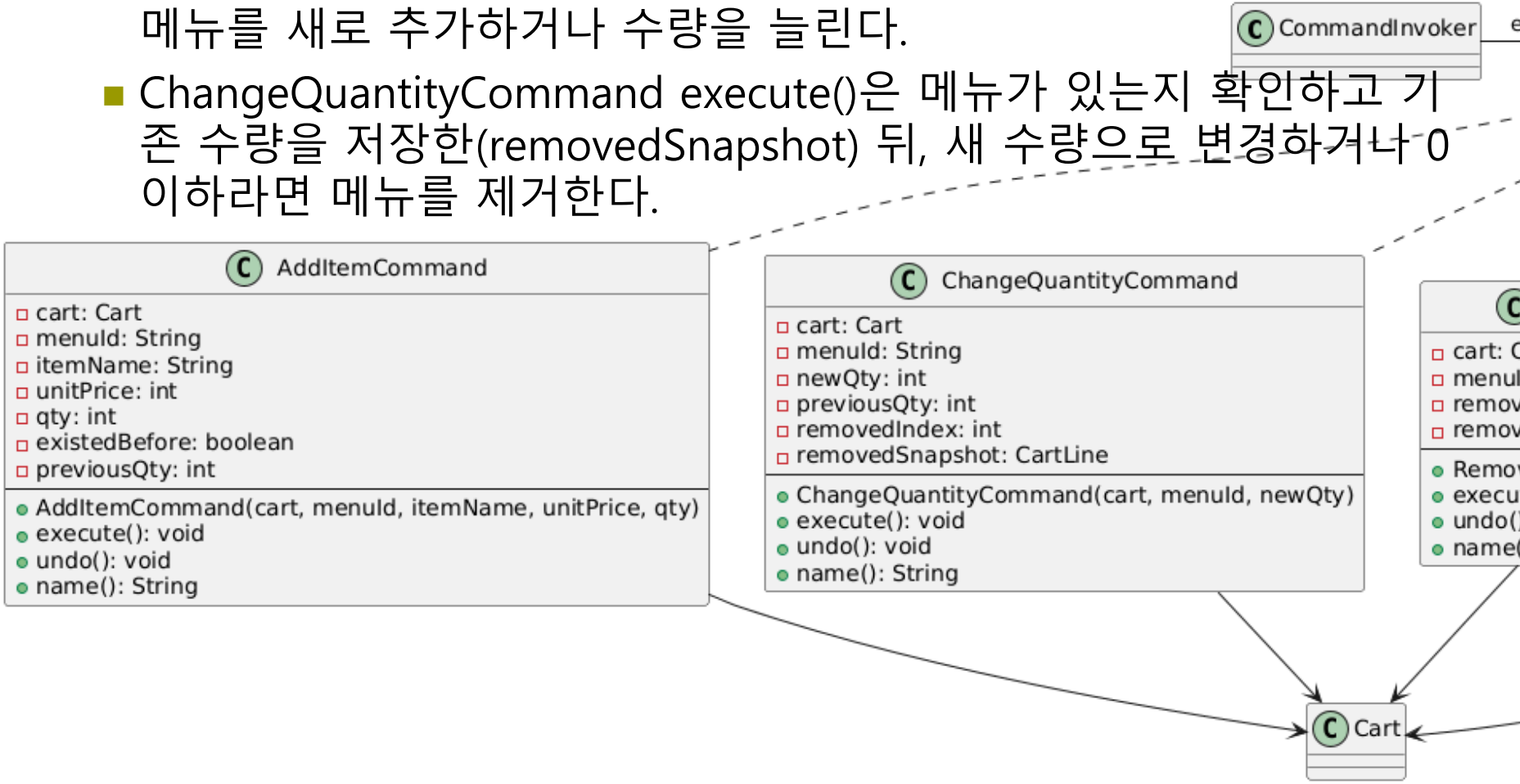
ICommand



Lab6

AddItemCommand, ChangeQuantityCommand

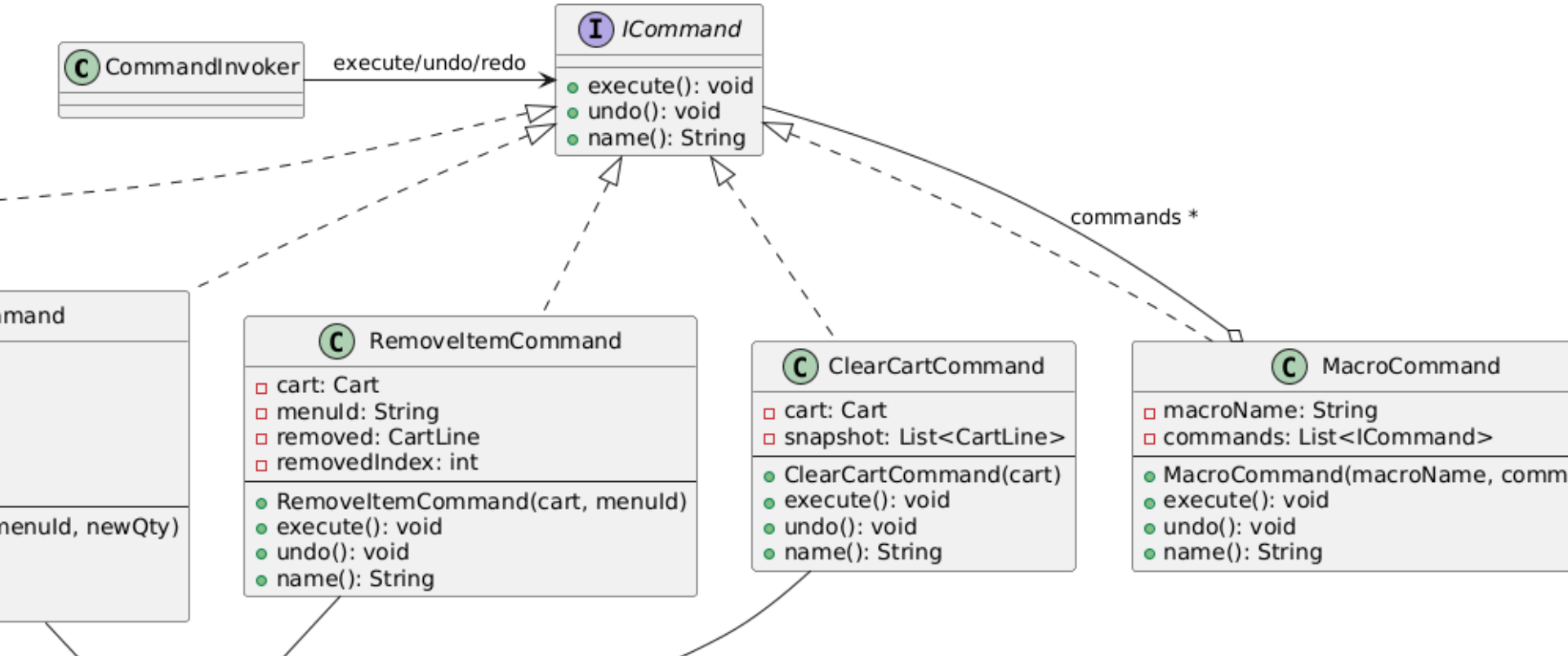
- AddItemCommand execute()은 기존 메뉴 여부와 수량을 기록한 뒤, 메뉴를 새로 추가하거나 수량을 늘린다.
- ChangeQuantityCommand execute()은 메뉴가 있는지 확인하고 기존 수량을 저장한(removedSnapshot) 뒤, 새 수량으로 변경하거나 0 이하라면 메뉴를 제거한다.



Lab6

RemoveItemCommand, ClearCartCommand

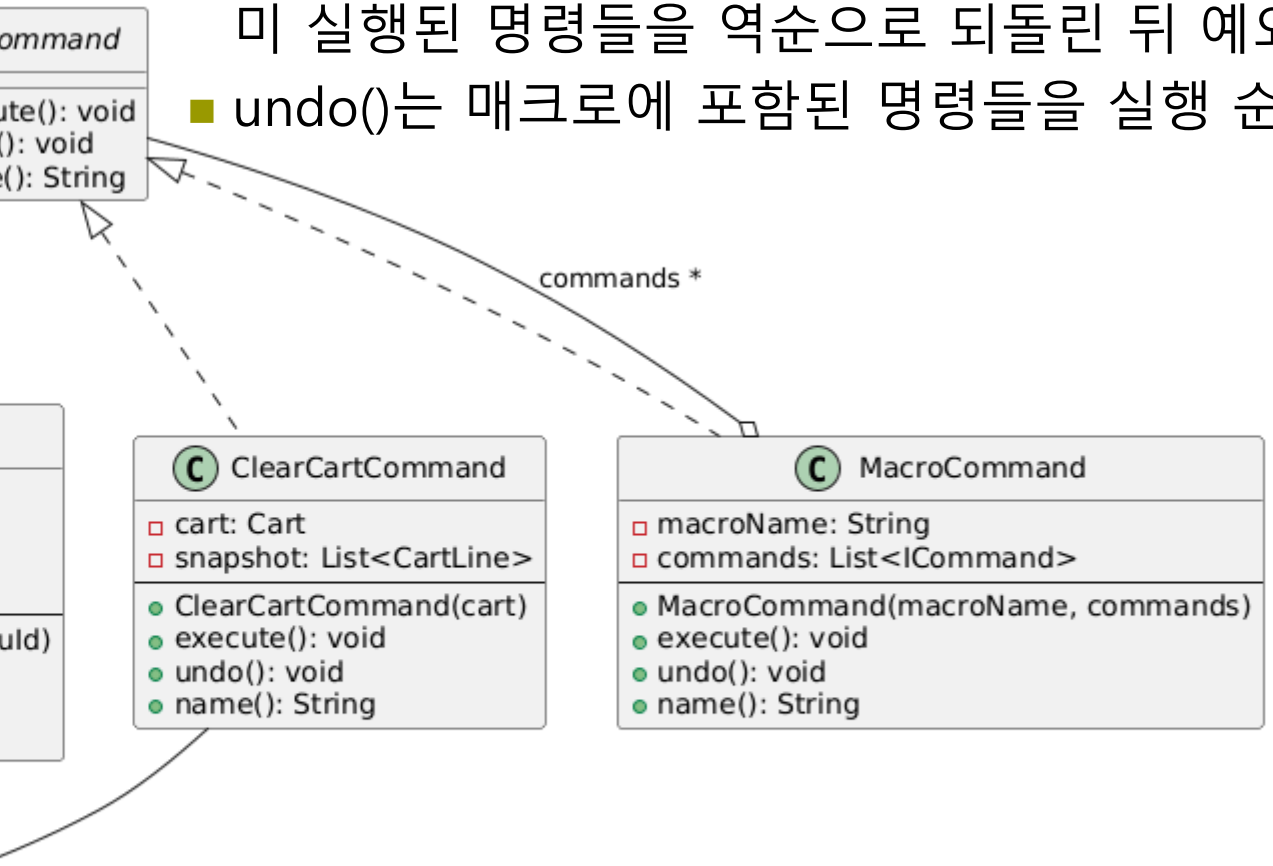
- RemoveItemCommand execute()은 메뉴의 현재 위치(removedIndex)를 기억한 뒤, 해당 메뉴를 장바구니에서 제거한다.
- ClearCartCommand execute()은 장바구니 상태를 저장(snapshot)한 뒤, 장바구니를 비운다. 그래야 undo()시 다시 되돌림.



Lab6

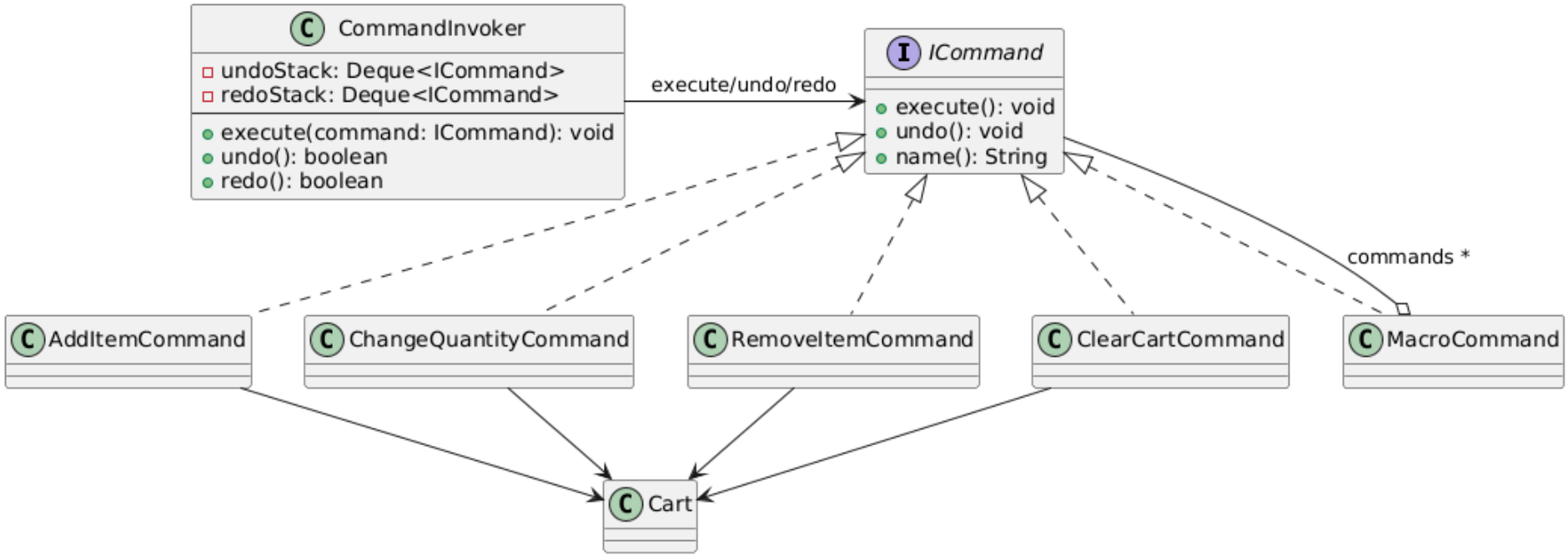
MacroCommand

- execute()는 List of ICommand을 순서대로 실행. 중간에 실패하면 이미 실행된 명령들을 역순으로 되돌린 뒤 예외를 던진다.
- undo()는 매크로에 포함된 명령들을 실행 순서의 역순으로 되돌린다.



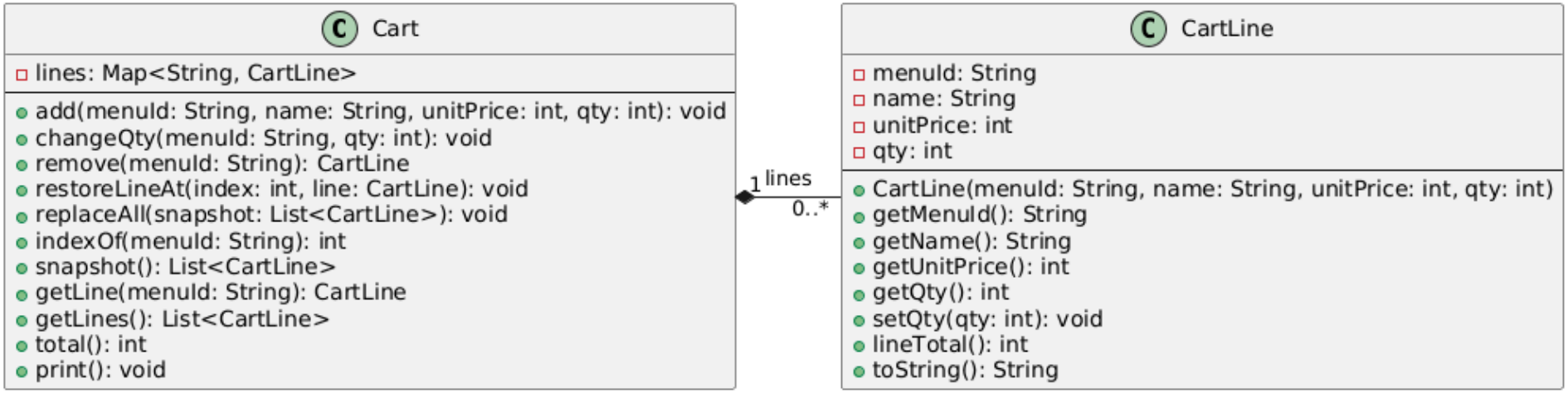
Lab6

- **CommandInvoker**는 명령을 **execute, undo, redo**
  - execute(ICommand c): **c.execute()** 후 push undoStack & clear redoStack
  - undo(): pop undoStack 후 **c.undo()** & push to redoStack
  - redo(): pop redoStack 후 **c.execute()** & push to undoStack



Lab6

- **Cart**는 **Receiver**로, 장바구니의 실제 상태를 보유하고 Command 요청을 처리한다.
  - lines는 LinkedHashMap<String, CartLine>이다. 키는 menuId이고, 담긴 순서가 유지된다.
- **CartLine**은 장바구니의 한 줄로, qty만 setQty()로 바꿀 수 있고, 나머지 값은 생성 후 바뀌지 않는다.
  - lineTotal()은 단가 * 수량을 반환한다.



Lab6

- **Cart** 메서드
  - **add(menuId, name, unitPrice, qty)** 이미 있는 메뉴면 수량을 더하고, 없으면 맨 뒤에 새 CartLine을 추가한다.
  - **changeQty(menuId, qty):** 수량을 변경. qty ≤ 0이면 줄을 삭제한다.
  - **Remove(menuId):** 줄을 삭제하고, 삭제한 CartLine을 반환한다.
  - **restoreLineAt(index, line):** 삭제했던 줄을 원래 위치에 되돌린다. Undo에서 사용한다.
  - **replaceAll(snapshot):** 전체 상태를 스냅샷으로 교체한다. 비우기와 비우기 undo에서 사용한다.
  - **indexOf(menuId)**: 현재 위치(0-based)를 반환하고, 없으면 -1을 반환. Undo 시 위치 복원을 위해 실행 전에 기억해 둔다.
  - **snapshot()**: 전체 줄을 깊은 복사한 목록을 반환한다. 비우기에서 사용
  - getLine(menuId)는 menuId로 CartLine을 찾아서 반환. getLines()는 장바구니 현재상태 읽기 전용 목록을 반환한다.
  - total() 장바구니 합계 반환, print() 장바구니 상태를 콘솔 출력한다.

```
public class AddItemCommand extends ICommand {
  private final Cart cart;
  private final String menuId;  private final String itemName;
  private final int unitPrice;  private final int qty;
  private boolean existedBefore;  private int previousQty;
…
  @Override
  public void execute(){
    CartLine existing = cart.getLine(menuId);
    existedBefore = existing != null;
    previousQty = existedBefore ? existing.getQty() : 0;
    cart.add(menuId, itemName, unitPrice, qty);
  }
  @Override
  public void undo(){
    if (existedBefore) cart.changeQty(menuId, previousQty);
    else cart.remove(menuId);
  }
  @Override
  public String name(){
    return "AddItem(" + itemName + " x" + qty + ")";
  }
}
```

Lab6

```
public class Cart {
  // key:menuId (예:"R05-제육덮밥"), value: 장바구니한줄 (담긴 순서 유지)
  private final Map<String, CartLine> lines = new LinkedHashMap<>();
  public void add(String menuId, String name, int unitPrice, int qty){
    CartLine existing = lines.get(menuId);
    if (existing != null) existing.setQty(existing.getQty() + qty);
    else lines.put(menuId, new CartLine(menuId, name, unitPrice, qty));
  }
  public void changeQty(String menuId, int qty) { …… }
  public CartLine remove(String menuId) { …… }  // 삭제한 줄 반환
  public int indexOf(String menuId) { …… }// 현재위치(undo 위치 복원용)
  public void restoreLineAt(int index, CartLine line) { …… } // 원래 위
치에 다시 넣기
  public List<CartLine> snapshot() { …… } // 깊은 복사 (비우기 undo용)
  public void replaceAll(List<CartLine> snapshot) { …… }  // 스냅샷으로
전체 교체
  public int total() { …… }          // 합계 금액
  ……
}
```

Lab6

```
public class MenuCartFrame extends JFrame {
...
    private final Cart cart = new Cart();
    private final CommandInvoker invoker = new CommandInvoker();
    // ... Swing UI 요소 ...

    public MenuCartFrame(List<MenuItem> menuItems) {
        super("Command Pattern + Swing Demo");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setSize(900, 600);
        setLocationRelativeTo(null);
        setResizable(false); // fixed size
.. Continue..
// 상단(버튼 패널)
// 1행: 메뉴 선택 + 수량 + 담기
// 2행: 비우기 + 매크로 + Undo/Redo
// 중앙(장바구니 목록) 메뉴ID, 메뉴명, 단가, 수량, 금액, [−]버튼, [+]버튼, [삭제] 버튼
// 하단(총 금액 + 로그)
```

Lab6

```
        // 매크로용 메뉴
          MenuItem jeoyuk = find(menuItems, "R05-제육덮밥");
          MenuItem ramyeon = find(menuItems, "R09-폭풍라면");
          MenuItem mandu = find(menuItems, "R01-왕만두");

          // 이벤트 모든 변경은 executeCommand()로 넘긴다
          btnAdd.addActionListener(e -> {
              MenuItem item = (MenuItem) menuCombo.getSelectedItem();
              int qty = (Integer) addQtySpinner.getValue();
              executeCommand(new AddItemCommand(cart, item.menuId(),
item.name(), item.price(), qty));
          });
          btnClear.addActionListener(e -> executeCommand(new
ClearCartCommand(cart)));
          btnMacro.addActionListener(e -> …
          btnUndo.addActionListener(e -> …
          btnRedo.addActionListener(e -> …
          updateView();
      }
```

Lab6

```
// Command 객체로 invoker.execute로 넘기고 로그를 남기고 화면 갱신
private void executeCommand(ICommand cmd) {
    try {
        invoker.execute(cmd);
        log("[실행] " + cmd.name());
    } catch (Exception ex) {
        log("[실패] " + ex.getMessage());
    }
    updateView();
}
// log 출력
private void log(String message) {
    logArea.append(message + "\n");
    logArea.setCaretPosition(logArea.getDocument().getLength());
}
```

Lab6

```
    // 화면 갱신: 장바구니 내용과 총 금액을 JTable과 JLabel에 반영한다.
    private void updateView() {
        tableModel.setRowCount(0);
        for (CartLine line : cart.getLines()) {
            tableModel.addRow(new Object[]{
                    line.getMenuId(),line.getName(),
                    String.format("%,d원", line.getUnitPrice()),
                    line.getQty(), String.format("%,d원",
line.lineTotal())
            });
        }
        totalLabel.setText(String.format("총 금액: %,d원  ",
cart.total()));
    }
}
```

Submit to e-learning

- Lab6 과제에 yourcode를 추가 (yourcode 없을시 10점에서 -1점 감점)
- Submit the Lab6 assignment (JAVA26-2-HW6-YourID-YourName.zip including the report) to e-learning (due by 11/2).