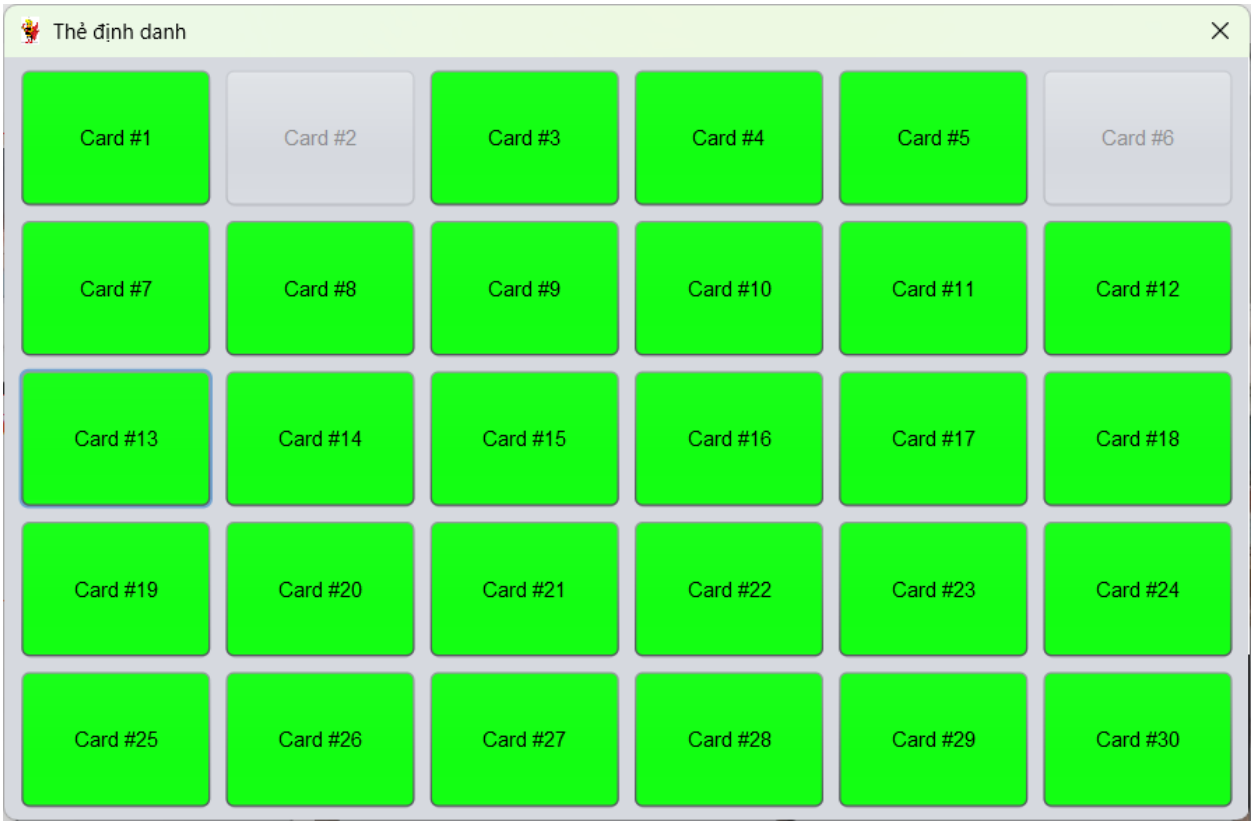
BÀI 1: SALSEJDIALOG

Xây dựng cửa sổ chức năng bán hàng hiển thị các thẻ định vị để nhân viên bán hàng quản lý phiếu bán hàng đang phục vụ theo thẻ có giao diện như sau:

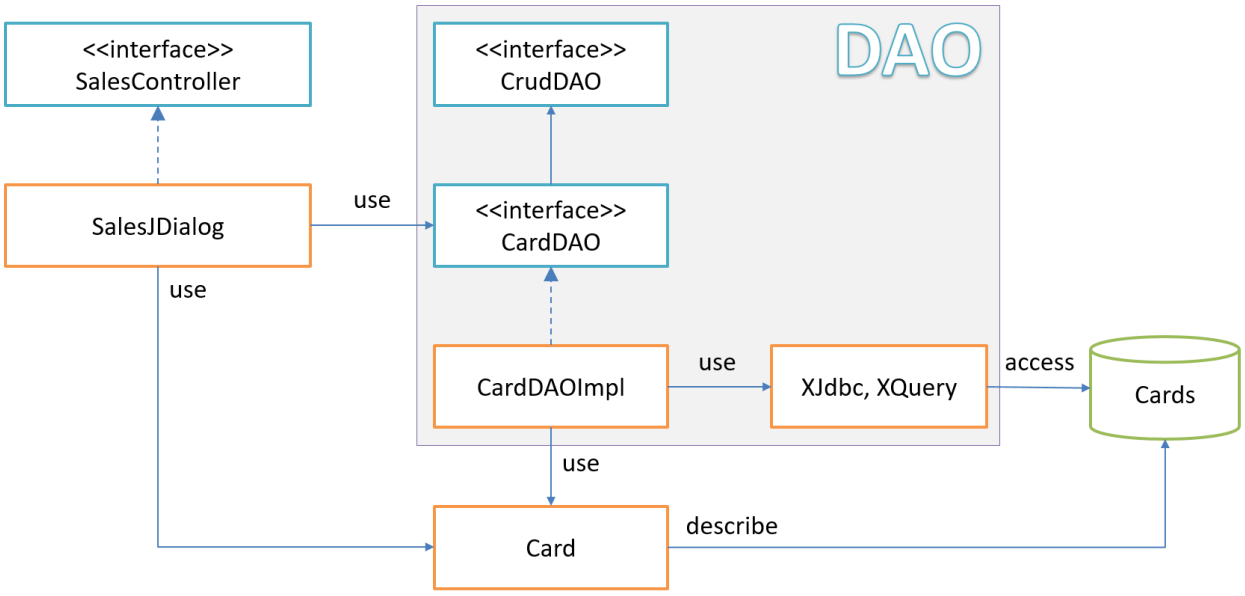


- Mở cửa sổ
  - Tải và hiển thị tất cả các thẻ định vị lên giao diện dưới dạng các JButton
  - Vô hiệu hóa các thẻ có trạng thái hư hỏng (error) hoặc thất lạc (lose)
  - Thiết lập mã thẻ định vị cho ActionCommand của mỗi nút
- Nhấp vào nút
  - Tải bill đang phục vụ gắn với thẻ (tạo mới nếu chưa có)
  - Hiển thị cửa sổ BillJDialog (chưa cần thiết kế giao diện)

CÁC THÀNH PHẦN CẦN THIẾT

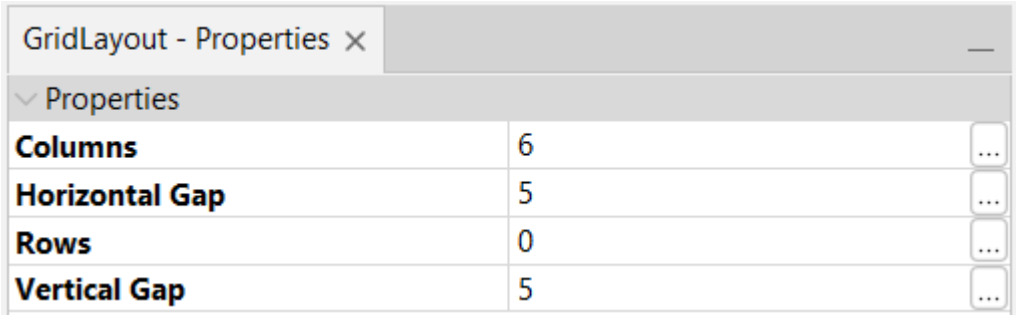
- GUI: SalesJDialog
- Controller: SalesController
- Tables: Cards
- Entities: Card
- DAO:
  - CrudDAO, CardDAO, CardDAOImpl, BillDAO, BillDAOImpl
  - XJdbc, XQuery

SƠ ĐỒ TỔ CHỨC



THIẾT KẾ GIAO DIỆN

- Sử dụng JPanel với Layout là GridLayout, đặt tên là pnlCards



HIỆU CHỈNH DAO

Bổ sung phương thức ***findServicingByCardId***() vào BillDAO và cài đặt mã để truy vấn Bill đang phục vụ theo thẻ định vị

```
public interface BillDAO extends CrudDAO<Bill, Long>{
        …
        public Bill findServicingByCardId(Integer cardId);
}
```

```
public class BillDAOImpl implements BillDAO{
        …
        @Override
        public Bill findServicingByCardId(Integer cardId) {
           String sql = "SELECT * FROM Bills WHERE CardId=? AND Status=0";
           Bill bill = XQuery.getSingleBean(Bill.class, sql, cardId);
           if (bill == null) { // không tìm thấy -> tạo mới
              Bill newBill = new Bill();
```

```
                newBill.setCardId(cardId);
                newBill.setCheckin(new Date());
                newBill.setStatus(0); // đang phục vụ
                newBill.setUsername(XAuth.user.getUsername());
                bill = this.create(newBill); // insert
            }
            return bill;
        }
}
```

CONTROLLER

Tạo interface SalesController và cài đặt mã để điều khiển các tương tác

```
public interface SalesController {
   void open(); // tải và hiển thị thẻ lên cửa sổ bán hàng
   void showBillJDialog(int cardId); // hiển thị cửa sổ chứa phiếu bán hàng hàng của 1 thẻ
}
```

```
public class SalesJDialog extends JDialog implements SalesController {
        ...
        @Override
        public void open() {
            this.setLocationRelativeTo(null);
            this.loadCards(); // tải và hiển thị các thẻ lên cửa sổ bán hàng
        }
        @Override
        public void showBillJDialog(int cardId) { // Hiển thị cửa số phiếu bán hàng của thẻ
            BillDAO dao = new BillDAOImpl();
            Bill bill = dao.findServicingByCardId(cardId); // tải bill đang phục vụ của thẻ
            BillJDialog dialog = new BillJDialog((Frame) this.getOwner(), true);
            dialog.setBill(bill); // Cần khai báo vào BillJDialog @Setter Bill bill
            dialog.setVisible(true);
        }
        private void loadCards() {// tải và hiển thị các thẻ lên cửa sổ bán hàng
            CardDAO dao = new CardDAOImpl();
            List<Card> cards = dao.findAll();
            pnlCard.removeAll();
            cards.forEach(card -> pnlCard.add(this.createButton(card)));
        }
        private JButton createButton(Card card) { // tạo Jbutton cho thẻ
            JButton btnCard = new JButton();
            btnCard.setText(String.format("Card #%d", card.getId()));
            btnCard.setPreferredSize(new Dimension(0, 80));
            btnCard.setEnabled(card.getStatus() == 0);
            btnCard.setBackground(btnCard.isEnabled() ? Color.GREEN : Color.GRAY);
            btnCard.setActionCommand(String.valueOf(card.getId()));
```

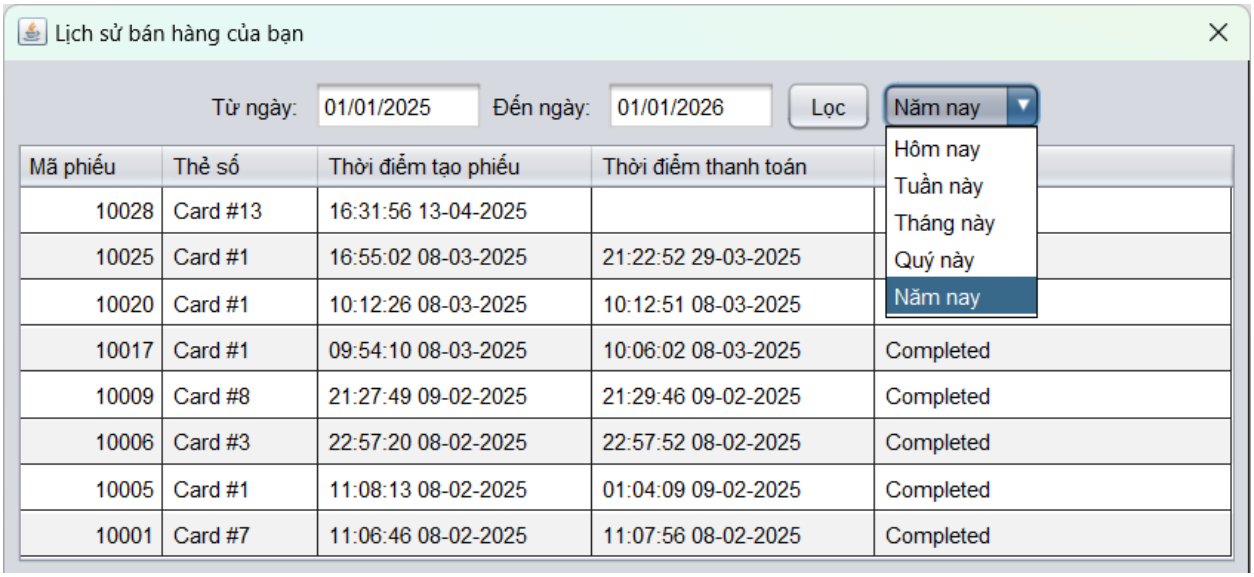
```
            btnCard.addActionListener((ActionEvent e) -> {
                int cardId = Integer.parseInt(e.getActionCommand());
                SalesJDialog.this.showBillJDialog(cardId);
            });
            return btnCard;
        }
}
```

KẾT NỐI CONTROLLER VỚI EVENT HANDLER

```
    private void formWindowOpened(java.awt.event.WindowEvent evt) {
        // TODO add your handling code here:
        this.open();
    }
```

BÀI 2: HISTORYJDIALOG

Xây dựng cửa sổ chức năng hiển thị các phiếu bán hàng của nhân viên đang làm việc theo khoảng thời gian lọc có giao diện minh họa như sau:

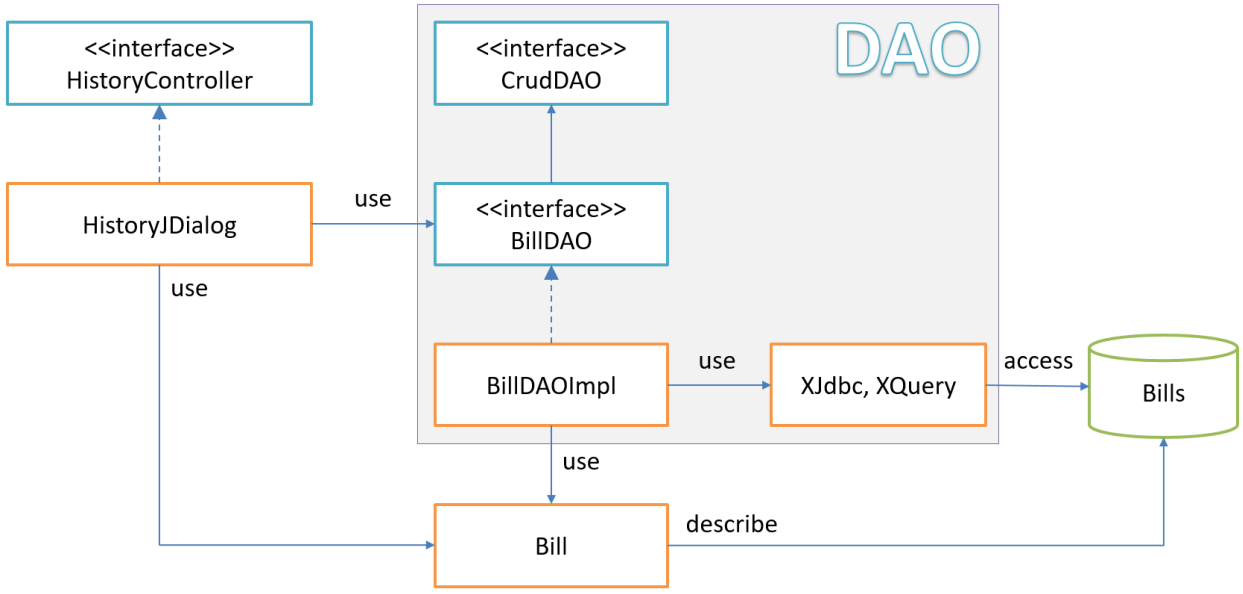


- Mở cửa sổ (HistoryJDialog)
  - Thiết lập khoảng thời gian lọc là Hôm nay
  - Tải và hiển thị bill theo thời gian lọc
- Click nút Lọc
  - Tải và hiển thị bill theo thời gian lọc
- Chọn khoảng thời gian
  - Thiết lập khoảng thời gian lọc theo khoảng thời gian chọn
  - Tải và hiển thị bill theo thời gian lọc
- Click đúp vào bill
  - Tải phiếu bán hàng được chọn
  - Hiển thị cửa sổ BillJDialog (chưa cần thiết kế giao diện)

CÁC THÀNH PHẦN CẦN THIẾT

- GUI: HistoryJDialog
- Controller: HistoryController
- Tables: Bills
- Entities: Bill
- DAO:
  - CrudDAO, BillDAO, BillDAOImpl
  - XJdbc, XQuery

SƠ ĐỒ TỔ CHỨC



THIẾT KẾ GIAO DIỆN

- tblBills: bảng chứa các phiếu bán hàng
- txtBegin: ô nhập Từ ngày
- txtEnd: ô nhập Đến ngày
- btnFilter: nút Lọc
- cboTimeRanges: Combo box chứa các khoảng thời gian

HIỆU CHỈNH DAO

Khai báo bổ sung phương thức ***findByUserAndTimeRange***() vào BillDAO và cài đặt mã để truy vấn bill theo username và khoảng thời gian lọc:

```
public interface BillDAO extends CrudDAO<Bill, Long>{
   ...
   List<Bill> findByUserAndTimeRange(String username, Date begin, Date end);
}
```

```
public class BillDAOImpl implements BillDAO{
...
      @Override
```

```
    public List<Bill> findByUserAndTimeRange(String username, Date begin, Date end) {
        String sql = "SELECT * FROM Bills " +
                     " WHERE Username=? AND Checkin BETWEEN ? AND ?";
        return XQuery.getBeanList(Bill.class, sql, username, begin, end);
    }
}
```

CONTROLLER

Tạo interface SalesController và cài đặt mã để điều khiển các tương tác

```
public interface HistoryController {
   void open(); // hiển thị bill theo khoảng thời gian Hôm nay
   void fillBills() ; // tải và hiển thị bill theo khoảng thời gian lọc
   void showBillJDialog(); // mở cửa số phiếu bán hàng
   void selectTimeRange(); // chọn khoảng thời gian
}
```

```
public class SalesJDialog extends JDialog implements HistoryController {
        ...
        BillDAO billDao = new BillDAOImpl();
        List<Bill> bills = List.of();

        @Override
        public void open() {
           this.setLocationRelativeTo(null);
           this.selectTimeRange();
        }
        @Override
        public void selectTimeRange() {
           TimeRange range = TimeRange.today();
           switch(cboTimeRanges.getSelectedIndex()){
              case 0 -> range = TimeRange.today();
              case 1 -> range = TimeRange.thisWeek();
              case 2 -> range = TimeRange.thisMonth();
              case 3 -> range = TimeRange.thisQuarter();
              case 4 -> range = TimeRange.thisYear();
           }
           txtBegin.setText(XDate.format(range.getBegin()));
           txtEnd.setText(XDate.format(range.getEnd()));
           this.fillBills();
        }
        @Override
        public void fillBills() {
           String username = XAuth.user.getUsername();
           Date begin = XDate.parse(txtBegin.getText(), "MM/dd/yyyy");
           Date end = XDate.parse(txtEnd.getText(), "MM/dd/yyyy");
```

```
            bills = billDao.findByUserAndTimeRange(username, begin, end);
            DefaultTableModel model = (DefaultTableModel) tblBills.getModel();
            model.setRowCount(0);
            String[] statuses = {"Servicing", "Completed", "Canceled"};
            bills.forEach(b -> {
            Object[] row = {
                    b.getId(),
                    "Card #" + b.getCardId(),
                    XDate.format(b.getCheckin(), "HH:mm:ss dd-MM-yyyy"),
                    XDate.format(b.getCheckout(), "HH:mm:ss dd-MM-yyyy"),
                    statuses[b.getStatus()]
                };
                model.addRow(row);
            });
        }
        @Override
        public void showBillJDialog() {
            Bill bill = bills.get(tblBills.getSelectedRow());
            BillJDialog dialog = new BillJDialog((Frame) this.getOwner(), true);
            dialog.setBill(bill); // truyền bill vào cửa sổ BillJDialog
            dialog.setVisible(true);
            dialog.addWindowListener(new java.awt.event.WindowAdapter() {
                @Override
                public void windowClosed(java.awt.event.WindowEvent e) {
                    HistoryJDialog.this.fillBills();
                }
            });
        }
}
```

KẾT NỐI CONTROLLER VỚI EVENT HANDLER

```
    private void tblBillsMouseClicked(java.awt.event.MouseEvent evt) {
        // TODO add your handling code here:
        if (evt.getClickCount() == 2) {
            this.showBillJDialog();
        }
    }

    private void formWindowOpened(java.awt.event.WindowEvent evt) {
        // TODO add your handling code here:
        this.open();
    }
```

```
private void cboTimeRangesActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    this.selectTimeRange();
}

private void btnFilterActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    this.fillBills();
}
```