**Report describing the series of refactorings I applied to each of the classes:**

**Class: AddRecordDialog.java**

1. **Line 90: New method - addComponents(JPanel)**

**Before:**

// initialize dialog container

public Container dialogPane() {

JPanel empDetails, buttonPanel;

empDetails = new JPanel(new MigLayout());

buttonPanel = new JPanel();

JTextField field;

empDetails.setBorder(BorderFactory.createTitledBorder("Employee Details"));

empDetails.add(new JLabel("ID:"), "growx, pushx");

empDetails.add(idField = new JTextField(20), "growx, pushx, wrap");

idField.setEditable(false);

empDetails.add(new JLabel("PPS Number:"), "growx, pushx");

empDetails.add(ppsField = new JTextField(20), "growx, pushx, wrap");

**// etc – add surname, firsname, gender, dept, salary, fulltime**

buttonPanel.add(save = new JButton("Save"));

save.addActionListener(this);

save.requestFocus();

buttonPanel.add(cancel = new JButton("Cancel"));

cancel.addActionListener(this);

empDetails.add(buttonPanel, "span 2,growx, pushx,wrap");

// loop through all panel components and add fonts and listeners

for (int i = 0; i < empDetails.getComponentCount(); i++) {

empDetails.getComponent(i).setFont(this.parent.font1);

if (empDetails.getComponent(i) instanceof JComboBox) {

empDetails.getComponent(i).setBackground(Color.WHITE);

}// end if

else if(empDetails.getComponent(i) instanceof JTextField){

field = (JTextField) empDetails.getComponent(i);

if(field == ppsField)

field.setDocument(new JTextFieldLimit(9));

else

field.setDocument(new JTextFieldLimit(20));

}// end else if

}// end for

idField.setText(Integer.toString(this.parent.getNextFreeId()));

return empDetails;

}

**After:**

**I extracted the empDetails.add.. etc, made a new method:**

private void addComponents(JPanel empDetails) {

empDetails.add(new JLabel("ID:"), "growx, pushx");

empDetails.add(idField = new JTextField(20), "growx, pushx, wrap");

idField.setEditable(false);

empDetails.add(new JLabel("PPS Number:"), "growx, pushx");

empDetails.add(ppsField = new JTextField(20), "growx, pushx, wrap");

empDetails.add(new JLabel("Surname:"), "growx, pushx");

empDetails.add(surnameField = new JTextField(20), "growx, pushx, wrap");

empDetails.add(new JLabel("First Name:"), "growx, pushx");

empDetails.add(firstNameField = new JTextField(20), "growx, pushx, wrap");

empDetails.add(new JLabel("Gender:"), "growx, pushx");

empDetails.add(genderCombo = new JComboBox<String>(this.parent.gender), "growx, pushx, wrap");

empDetails.add(new JLabel("Department:"), "growx, pushx");

empDetails.add(departmentCombo = new JComboBox<String>(this.parent.department), "growx, pushx, wrap");

empDetails.add(new JLabel("Salary:"), "growx, pushx");

empDetails.add(salaryField = new JTextField(20), "growx, pushx, wrap");

empDetails.add(new JLabel("Full Time:"), "growx, pushx");

empDetails.add(fullTimeCombo = new JComboBox<String>(this.parent.fullTime), "growx, pushx, wrap");

}

Then in the original method when I initialized the container I passed the JPanel to my new method:

addComponents(empDetails);

1. **Line 134: isValid(Component comp)**

* I extracted the code from the method ‘checkInput()’ to set the background if the entry is valid.

**Before:**

// check for input in text fields

public boolean checkInput() {

boolean valid = true;

// if any of inputs are in wrong format, colour text field and display message

if (ppsField.getText().equals("")) {

ppsField.setBackground(new Color(255, 150, 150));

valid = false;

}// end if

if (this.parent.correctPps(this.ppsField.getText().trim(), -1)) {

ppsField.setBackground(new Color(255, 150, 150));

valid = false;

}// end if

if (surnameField.getText().isEmpty()) {

surnameField.setBackground(new Color(255, 150, 150));

valid = false;

}// end if

if (firstNameField.getText().isEmpty()) {

firstNameField.setBackground(new Color(255, 150, 150));

valid = false;

**// etc – gender, dept, salary, fulltime**

**After:**

1. **New method:**

public void isValid(Component comp){

comp.setBackground(new Color(255,150,150));

}

1. **Update method ‘checkInput()’:**

// check for input in text fields

public boolean checkInput() {

boolean valid = true;

// if any of inputs are in wrong format, colour text field and display message

if (ppsField.getText().equals("")) {

isValid(ppsField);

valid = false;

}// end if

if (this.parent.correctPps(this.ppsField.getText().trim(), -1)) {

isValid(ppsField);

valid = false;

}// end if

if (surnameField.getText().isEmpty()) {

isValid(surnameField);

valid = false;

}// end if

if (firstNameField.getText().isEmpty()) {

isValid(firstNameField);

valid = false;

**// etc – gender, dept, salary, fulltime**

**Class: Employee.java**

1. Line 116: Refactored the Employee toString() method’s return statement.

**Before:**

return "Employee ID: " + this.employeeId + "\nPPS Number: " + this.pps + "\nSurname: " + this.surname

+ "\nFirst Name: " + this.firstName + "\nGender: " + this.gender + "\nDepartment: " + this.department + "\nSalary: " + this.salary

+ "\nFull Time: " + bool;

**After:**

return "Employee ID: " + this.getEmployeeId() + "\nPPS Number: " + this.getPps() + "\nSurname: " + this.getSurname()

+ "\nFirst Name: " + this.getFirstName() + "\nGender: " + this.getGender() + "\nDepartment: " + this.getDepartment() + "\nSalary: " + this.getSalary()

+ "\nFull Time: " + bool;

**Class: EmployeeDetails.java**

1. Line 645: extracted code from the method isSomeoneToDisplay() and added a new method clearFields() so that it could be reused elsewhere in the code.

**Before:**

…

…

if (!someoneToDisplay) {

currentEmployee = null;

idField.setText("");

ppsField.setText("");

surnameField.setText("");

firstNameField.setText("");

salaryField.setText("");

genderCombo.setSelectedIndex(0);

departmentCombo.setSelectedIndex(0);

fullTimeCombo.setSelectedIndex(0);

JOptionPane.showMessageDialog(null, "No Employees registered!");

}

return someoneToDisplay;

}

**After:**

1. **New method:**

private void clearFields() {

currentEmployee = null;

idField.setText("");

ppsField.setText("");

surnameField.setText("");

firstNameField.setText("");

salaryField.setText("");

genderCombo.setSelectedIndex(0);

departmentCombo.setSelectedIndex(0);

fullTimeCombo.setSelectedIndex(0);

}

1. **Update method isSomeoneToDisplay()**

…

…

if (!someoneToDisplay) {

clearFields();

JOptionPane.showMessageDialog(null, "No Employees registered!");

}

return someoneToDisplay;

}// end isSomeoneToDisplay

1. Line 712: isValid(Component comp) – similar to AddRecordDialog.java I extracted the code for setting the background in the mehod checkInput() if it’s valid.

**Before:**

// check for input in text fields

private boolean checkInput() {

boolean valid = true;

// if any of inputs are in wrong format, colour text field and display

// message

if (ppsField.isEditable() && ppsField.getText().trim().isEmpty()) {

ppsField.setBackground(new Color(255, 150, 150));

valid = false;

} // end if

if (ppsField.isEditable() && correctPps(ppsField.getText().trim(), currentByteStart)) {

ppsField.setBackground(new Color(255, 150, 150));

valid = false;

} // end if

if (surnameField.isEditable() && surnameField.getText().trim().isEmpty()) {

surnameField.setBackground(new Color(255, 150, 150));

valid = false;

} // end if

**// etc – firsname, gender, dept, salary, fulltime**

**After:**

1. **New method:**

public void isValid(Component comp){

comp.setBackground(new Color(255,150,150));

}

1. **Updated checkInput()**

// check for input in text fields

public boolean checkInput() {

boolean valid = true;

// if any of inputs are in wrong format, colour text field and display message

if (ppsField.getText().equals("")) {

isValid(ppsField);

valid = false;

}// end if

if (this.parent.correctPps(this.ppsField.getText().trim(), -1)) {

isValid(ppsField);

valid = false;

}// end if

if (surnameField.getText().isEmpty()) {

isValid(surnameField);

valid = false;

}// end if

if (firstNameField.getText().isEmpty()) {

isValid(firstNameField);

valid = false;

**// etc – gender, dept, salary, fulltime**

**Class: EmployeeSummaryDialog.java**

1. Line 72: ConstructTable(String[]) – I extracted the code to construct the table so that it could be reused and so the method was more readable.

**Before:**

public Container summaryPane() {

…

…

// construnct table and choose table model for each column

tableModel = new DefaultTableModel(this.allEmployees, header) {

public Class getColumnClass(int c) {

switch (c) {

case 0:

return Integer.class;

case 4:

return Character.class;

case 6:

return Double.class;

case 7:

return Boolean.class;

default:

return String.class;

}// end switch

}// end getColumnClass

};

**After:**

1. **New method**

private DefaultTableModel constructTable(Vector<String> header) {

return new DefaultTableModel(this.allEmployees, header) {

public Class getColumnClass(int c) {

switch (c) {

case 0:

return Integer.class;

case 4:

return Character.class;

case 6:

return Double.class;

case 7:

return Boolean.class;

default:

return String.class;

}// end switch

}// end getColumnClass

};

}

1. **Updated public Container summaryPane()**

// construnct table and choose table model for each column

tableModel = constructTable(header);

employeeTable = new JTable(tableModel);