Задание.

Создать в оболочке NetBeans IDE проект из двух классов – базовый(родительский) класс и от него класс потомок. Надо сделать строго по методичке иначе преподаватель не принимает.

Таблица 11.2 – Варианты индивидуальных заданий.

|  |  |  |  |
| --- | --- | --- | --- |
| 17 | Вектор на плоскости: координаты вектора на плоскости (x1, y1, x2, y2) | Вычислить длину вектора | Изобразить линию вектора на форме (Image) с толщиной линии 2 пикселя |

Это базовый класс.

Таблица 11.5 – Варианты индивидуальных заданий.

|  |  |  |  |
| --- | --- | --- | --- |
| 17 | Вектор на плоскости: координаты  вектора на плоскости (x1, y1, x2, y2) | Вектор и точка на плоскости: координаты вектора ( x1, y1,x2, y2); координаты точки–x3, y3 | Определить площадь треугольника, образованного вектором и точкой. |

Это класс потомок.

**МЕТОДИЧЕСКИЕ УКАЗАНИЯ СТУДЕНТАМ**

по выполнению самостоятельной работы

В качестве примера разберем задачу № 30 из темы 11 (Элементы объектно-ориентированного программирования). Задача состоит из двух частей Таблица 11.2 (11.1. Cоздание данных типа «класс» - Средний уровень), № вар.30 и Таблица 11.5 (11.2. Наследование классов,- Средний уровень), № вар.30.

## Условие задания

Разберем условие индивидуального задания №30. Общие условия задачи для темы «Cоздание данных типа «класс»» для среднего уровня в задачнике в задачнике формируются так:

Задание: создать класс с полями, указанными в индивидуальном задании (табл. 11.2, столб 2). Реализовать в классе методы:

* конструктор по умолчанию;
* конструктор перезагрузки с параметрами;
* деструктор для освобождения памяти (с сообщением об уничтожении объекта);
* функции обработки данных (1 и 2), указанные в индивидуальном задании

Сразу отметим, что конструктор перезагрузки и деструктор нам не понадобиться, так же мы не будем выводить результаты в компонент Memo.

Для темы «Наследование классов» задание формулируется так:

Для класса, созданного в предыдущем задании (по вариантам табл. 11.2) создать класс-потомок с полями, указанными в индивидуальном задании (табл. 11.5, столб 2). Реализовать в классе-потомке методы:

* - конструктор;
* - функцию обработки данных, указанную в индивидуальном задании (табл.11.5, столб 3);
* - функцию формирования строки информации об объекте.

Создать проект для демонстрации работы: ввод и вывод информации об объектах: классе-родителе и классе-потомке. Строку информации об объекте, мы тоже делать не будем (не очень понятно, что имел в виду автор задачника).

Теперь приведем формулировку варианта индивидуального задания №30 из таблицы 11.2:

|  |  |  |
| --- | --- | --- |
| **Класс-родитель и его поля** | **Функция-метод 1 обработки данных** | **Функция-метод 2 обработки данных** |
| Работник: фамилия, должность, оклад | Увеличить оклад на15% (каждому работнику) | Работникам, у которых фамилия начинается с сочетания букв «Иван»,  присвоить должность «инженер» |

И из таблицы 11.5:

|  |  |  |
| --- | --- | --- |
| **Класс-родитель и его поля (из табл. 11.2)** | **Класс-потомок и его поля (поля класса-родителя выделены курсивом)** | **Функция-метод обработки данных объекта класса-потомка** |
| Работник: фамилия, должность, оклад | Работники предприятия:  фамилия, должность, оклад, *рейтинг* (в 100-бальной системе) | Увеличить оклад работников на 20%, если их рейтинг от 60 до 75 балов, на 40%, если их рейтинг  от 75 до 90 балов, на60%, если их рейтинг от 90 до 100 балов. |

Нам нужно будет написать два класса (базовый класс и его потомок) , для представления понятия «Работник». Кроме этого нам потребуется класс, представляющий форму с элементами управления. При помощи этой формы мы будем демонстрировать работу двух наших написанных классов.

Обратите внимание, что в задании говориться о работниках во множественном числе, значит, нам потребуется иметь дело со списком экземпляров нашего класса (объектами «Работник»).

## Создание проекта

Поскольку разработка Java в среде IDE всегда выполняется в рамках проектов, сначала необходимо создать новый проект , в котором будут сохраняться исходные файлы и другие файлы проекта. Проект среды IDE представляет собой группу исходных файлов Java и связанные с ними метаданные, включая файлы свойств проекта, сценарий сборки Ant, управляющий параметрами сборки и выполнения, а также файл project.xml.

Итак, расписываю по шагам:

1. Запускаем оболочку NetBeans IDE;
2. Выбираем команду "***Файл***" > "***Новый проект***". Также можно щелкнуть значок "***Новый проект***" на панели инструментов среды IDE.
3. Введите  в поле "***Имя проекта***" название приложения, которое вы желаете (я задал название по номеру варианта Example\_11\_30) и укажите местоположение проекта.
4. Выбираем категорию "***Java***" и вид проекта "***Приложение Java*** ", нажимаем "***Далее***"
5. Не устанавливайте флажок "***Использовать отдельную папку для хранения библиотек***".
6. Очистите флажок "***Создать главный класс***".
7. Нажмите кнопку "***Готово***".

Среда IDE создаст в системе папку с заданным именем по указанному пути. Эта папка содержит все файлы, связанные с проектом, включая сценарий Ant, папки для хранения исходных файлов и тестов, а также папку с метаданными проекта. Для просмотра структуры проекта используйте окно "***Файлы***" в среде IDE.

## Создание базового класса.

Сначала создадим в проекте базовый класс для представления понятия «Работник». Чтобы создать новый класс и задать его содержимое в соответствии с заданием (см. таблицу 11.2) выполните действия, описанные ниже:

1. В окне 'Проекты' щелкните правой кнопкой мыши узел с созданным проектом и выберите ***"Новый"*** >"***Класс Java*** "
2. Задайте имя класса (я задал имя Class1)
3. Нажмите кнопку "***Готово***". Будет создан новый класс с заданным именем;
4. Теперь редактируем текст созданного класса. В соответствии с заданием нам нужно создать его поля и два метода обработки данных. Двойной щелчок по имени класса в дереве проектов приводит к тому, что в окне редактора исходного кода появляется закладка со сгенерированным исходным кодом нашего класса. Для осуществления классом какой-либо деятельности следует дописать свой собственный код.
5. Редактируя исходный код , создаем поля класса:

* два поля строкового типа (фамилия, должность)
* одно поле целого типа (оклад)

этот код будет выглядеть так:

public class Class1 {

String name, position;

int salary;

1. Создаем метод 1 обработки данных.

void IncSalary() {

salary = (int) (salary\*1.15);

};

Имя метода – IncSalary, что означает «увеличить оклад». Слово void означает, что наш метод ничего не будет возвращать. Метод, как и требуется в задании, увеличивает оклад на 15%.

1. Создаем метод 2 обработки данных.

void Position2Engineer() {

position = "Инженер";

};

Имя метода – Position2Engieer, что означает присвоить должность «инженер» (должность в инженер). Метод крайне примитивен и просто присваивает члену класса position значение «Инженер» (проверку фамилии на «иван» будем делать в главной форме).

## Создание класса потомка.

Теперь должен быть создан класс, расширяющий понятие «Работник» (см. таблицу 11.5).

1. Создаем новый класс, т.е. повторяем пункты 1-3 из предыдущего раздела. Имя класса задается как Class2. Редактируем сгенерированный исходный код класса Class2.
2. Делаем его потомком класса Class1, для этого в описании класса добавляем указание на предка

public class **Class2** extends Class1 {

1. Создаем член класса рейтинг

int rating;

1. Создаем метод обработки данных класса-потомка. Он будет называться – ChngSalary - Изменить оклад . Оклад изменяем в зависимости от рейтинга работника.

void **ChngSalary**() {

if (rating >= 60 && rating <75 ) {

salary = (int) (salary\*1.2);

}

else if (rating >= 75 && rating <90 ) {

salary = (int) (salary\*1.4);

}

else if (rating >= 90 && rating <=100 ) {

salary = (int) (salary\*1.6);

}

Теперь, когда классы созданы, нужно создать форму с элементами управления, при помощи которой мы будем демонстрировать работу двух наших написанных классов.

## Создание класса главной формы.

Для разработки формы с элементами управления можно воспользоваться классами, находящимися в пакете swing. В этом случае вам придется написать довольно большой объем кода, при помощи которого вы будете размещать элементы управления на форме и задавать реакцию на действия с этими элементами (например, действие при нажатии на кнопку). Есть возможность упростить разработку если часть работы по написанию кода возложить на специальный инструмент для визуальной разработки интерфейса. Это средство GUI Builder включенное в среду NetBeans IDE позволяет разработчикам проектировать и составлять интерфейсы к программам посредством перетаскивания элементов в рабочую область т.е. освобождает разработчиков от необходимости изучения особенностей использования Swing.

Для разработки формы с использованием GUI Builder выполните следующие действия:

1. В окне "***Проекты***" щелкните правой кнопкой мыши узел нашего проекта и выберите "***Новый***" > "***Форма JFrame***".
2. Введите придуманное вами название формы в поле имени класса.
3. Нажмите кнопку "***Готово***". Среда IDE создаст новую форму с заданным вами именем и класс с таким же именем. В редакторе открывается закладка с этим именем. Теперь можно приступить к разработке пользовательского интерфейса приложения.
4. Для визуального проектирования нажимаем кнопку "***Проект***" и переходим в визуальный редактор.
5. Устанавливаем желаемые размеры формы
6. При помощи окна "С***войства***" устанавливаем заголовок окна. В нашем случае свойство title устанавливаем как «Задание 11 30»

Созданный нами класс должен содержать элементы интерфейса для демонстрации работы и методы для описания поведения этих элементов. Кроме этого нам потребуется создать список объектов (работников) и методы для работы с этим списком.

Перечислим элементы интерфейса, которые нам понадобятся для демонстрации работы двух написанных нами классов. Итак, нам потребуются:

* Поле (JtextField) для ввода/вывода фамилии работника (Class1.name)
* Поле (JtextField) для ввода/вывода должности работника (Class1.position)
* Поле (JtextField) для ввода/вывода оклада работника (Class1. Salary)
* Поле (JtextField) для ввода/вывода рейтинга работника (Class2.rating)
* Кнопка (Jbuttton) для создания нового объекта (экземпляра работника)
* Кнопка (Jbuttton) для удаления объекта
* Кнопка (Jbuttton) для выполнения метода 1 обработки данных класса-родителя (Class1.IncSalary)
* Кнопка (Jbuttton) для выполнения метода 2 обработки данных класса-родителя – (Class1. Position2Engieer)
* Кнопка (Jbuttton) для выполнения метода обработки данных класса-потомка – (Class2. ChngSalary)
* компонент выбора числа или счетчик JSpinner предназначенный для выбора текущего объекта из списка.
* Поле для отображения количества элементов списке.

Кроме этого на форме присутствуют элементы для улучшения внешнего вида, такие как поясняющие надписи (JLabel) и разделяющая линия jSeparator

1. Размещаем на форме все элементы интерфейса.
   1. В окне "***Палитра***" находим компонент Метка и перетаскиваем его на форму. В окне "***Свойства***" заполняем text – Фамилия
   2. Компонент «Текстовое поле» размещаем рядом с меткой «Фамилия». Переименуем это поле, В окне "***Свойства***"переключимся на закладку «Код» и устанавливаем «Имя переменной» JtextFieldName.
   3. Компонент Метка перетаскиваем на форму и задаем text – Должность
   4. Компонент «Текстовое поле» размещаем рядом с меткой «Должность», переименуем его как JtextFieldPosition.
   5. Компонент Метка перетаскиваем на форму и задаем text –Оклад
   6. Компонент «Текстовое поле» размещаем рядом с меткой «Оклад», переименуем его как JtextFieldSalary.
   7. Размещаем на форме компонент JSpinner, переименуем его как JSpinnerCrn (от current - текущий)
   8. Размещаем компонент «Текстовое поле», переименуем его в JTextFieldCount.
   9. Находим компонент Кнопка и перетаскиваем его на форму. Устанавливаем текст на кнопке "***Новый***" и меняем имя на JButtonNewPers (NewPers имеется в виду Новый человек)
   10. Компонент Кнопка перетаскиваем на форму и задаем text –Удалить, Имя переменной – JbuttonDelPers
   11. Кнопку перетаскиваем на форму и задаем text –« Увеличить оклад», Имя переменной - JButtonIncSalary
   12. Кнопку перетаскиваем на форму и задаем text –«Иван в инженеры», Имя переменной - JButtonPosition2Engieer
   13. Помещаем на форму разделяющую линию jSeparator (Она зрительно отделит то что относится к классу-родителю от того что относится к классу-потомку)
   14. Перетаскиваем на форму метку, задаем text –Рейтинг
   15. Компонент «Текстовое поле» перетаскиваем на форму, переименуем его как JtextFieldRating
   16. Компонент Кнопка перетаскиваем на форму и задаем text –«Увеличить оклад по рейтингу», Имя переменной – JbuttonChngRating

## Создание класса главной формы.

В результате размещения элементов на форме мы имеем пустой каркас приложения, т.е. внешний вид полностью соответствует тому, что мы хотели, но при нажатии на кнопки ничего не происходит. Теперь требуется написать код, который будет делать все что нам требуется.

Обсудим общую идею нашего приложения.

Все объекты, соответствующие работникам будут создаваться при нажатии на кнопку "***Новый***" (JButtonNewPers) и помещаться в список. Количество элементов в списке отображается в поле JTextFieldCount. Поля текущего объекта будут отображаться в полях формы JtextFieldName, JtextFieldPosition, JtextFieldSalary и JtextFieldRating. При помощи этих же элементов мы будем присваивать значение членам класса. Т.е. все объекты хранятся в списке и при помощи текстовых полей формы мы получаем двусторонний доступ к членам текущего объекта. Текущий объект выбирается из списка при помощи счетчика JSpinnerCrn. Удаляется текущий объект из списка при нажатии на кнопку "***Удалить***" (JbuttonDelPers).

Все методы классов (Class1, Class2) выполняются при нажатии на кнопки "***Увеличить оклад***" (JButtonIncSalary), "***Иван в инженеры***" , (JButtonPosition2Engieer) и "***Увеличить оклад по рейтингу***", (JButtonChngRating). Все три метода выполняются для каждого объекта из списка.

Нажимаем на кнопку "***Источник***" в редакторе, т.е. переключаемся в режим редактирования кода. Область кода, которая была автоматически создана, будет выделена серыми областями, которые называются защитными блоками. Код в защищенных областях изменить невозможно . Редактировать код класса можно только для кода на белом фоне окна редактора. Теперь начинаем программировать:

1. Импортировать имя класса ArrayList из пакета java.util, это значит, что мы получаем возможность обращаться к классу ArrayList напрямую. Класс ArrayList это класс представляющий список.

import java.util.ArrayList;

1. Объявляем член класса типа список.

ArrayList<Class2> personList;

Наш список будет иметь название personList. Название класса Class2 в угловых скобках означает, что список будет содержать объекты типа Class2 (Класс-потомок).

Кроме этого объявим переменную currentInd, которая будет хранить что номер текущего объекта из списка personList (отметим, что нумерация объектов в списке начинается с 0).

Программируем конструктор класса, т.е. проектируем действия, которые наша форма совершает в момент своего создания. Наш конструктор должен создать экземпляр списка personList

public **Example11\_30JFrame**() {

initComponents();

personList = new ArrayList <Class2>();

currentInd = -1;

}

Первая строка конструктора была создана автоматически, в этой строке вызывается функция, которая создает все элементы, расположенные на форме. Мы дописываем две строки. В первой создается экземпляр списка personList, во второй мы указываем, что номера текущего объекта из списка еще нет (ведь список еще пустой).

## Методы, обеспечивающие интерфейс класса «Работник» и главной формы

Теперь нам потребуются два метода, которые будут осуществлять копирование данных из текущего объекта списка в поля формы и обратно. Сначала покажем метод копирования данных из объекта в поля формы, т.е. показа текущего экземпляра класса. Этот метод назван GetPerson.

protected void **GetPerson**(){

if (currentInd >= 0) {

jTextFieldName.setText(personList.get(currentInd).name);

jTextFieldPosit.setText(personList.get(currentInd).position);

jFormattedTextFieldSalary.setText("" + personList.get(currentInd).salary);

jFormattedTextFieldRating.setText("" + personList.get(currentInd).rating);

}

}

Копирование осуществляется, если в списке существует текущий объект, т.е. номер текущего объекта (currentInd) больше или равен нулю. Текущий элемент списка мы каждый раз находим при помощи строки personList.get(currentInd), т.е. при помощи метода get класса personList. Текст в текстовом поле устанавливаем при помощи метода setTex. Так как оклад и рейтинг имеют тип int, то необходимо сделать преобразование к строковому типу. Это осуществляется тем, что сначала указываем пустую строку ( ""), а потом добавляем к ней значение поля объекта в этом случае Java выполнит преобразование автоматически.

Метод, который решает обратную задачу, т.е. записывает в поля текущего объекта данные из текстовых полей формы, называется **SetPerson:**

protected void **SetPerson**(){

if ((currentInd >= 0) && (jTextFieldName.getText() != null)) {

personList.get(currentInd).name = jTextFieldName.getText();

personList.get(currentInd).position = jTextFieldPosit.getText();

try {

personList.get(currentInd).salary = new Integer(jFormattedTextFieldSalary.getText());

} catch (Exception e) {

personList.get(currentInd).salary = 0;

jFormattedTextFieldSalary.setText(null);

}

try {

personList.get(currentInd).rating = new Integer(jFormattedTextFieldRating.getText());

} catch (Exception e) {

personList.get(currentInd).rating = 0;

jFormattedTextFieldRating.setText(null);

}

}

}

Копирование в текущий объект происходит, если этот объект существует, т.е. номер текущего объекта (currentInd) больше или равен нулю и если в поле «Фамилия» на форме что-то записано (jTextFieldName.getText() != null))

Преобразование строки из текстового поля в целочисленные члены класса происходит с использованием конструктора, так называемого оболочечного класса (мы используем оболочечный класс Integer).

В примере была использована конструкция try-catch. Это необходимо для обработки ошибки, в случае, если строка содержит иные символы кроме чисел.

## Методы для работы со списком работников

Теперь запрограммируем реакцию на нажатие кнопок, связанных с созданием/удалением объектов в списке. Для того чтобы привязать к кнопке реакцию на ее нажатие нужно просто сделать на ней двойной щелчок (в окне "***Проекты***"). Будет сгенерирован метод класса jButtonNewPersActionPerformed (для кнопки jButtonNewPers). Кроме этого в защищенной области в приватном методе initComponents появится код, связывающий сгенерированный метод и саму кнопку. Также имя этого метода появится в окне свойств кнопки на закладке ***"События***" (событие ActionPerformed).

private void **jButtonNewPersActionPerformed**(java.awt.event.ActionEvent evt) {

}

Работает это так, когда пользователь нажмет кнопку, кнопка создаст объект класса ActionEvent и вызовет jButtonNewPersActionPerformed (event).

Сгенерированный метод пока пустой, чтобы приложение выполняло какое-нибудь действие в ответ на нажатие на кнопку необходимо дописать свой код. Для кнопки jButtonNewPers напишем следующее:

private void **jButtonNewPersActionPerformed**(java.awt.event.ActionEvent evt) {

SetPerson();

personList.add(new Class2());

currentInd = personList.size()-1;

GetPerson();

jTextFieldCount.setText(""+(currentInd+1));

jSpinnerCrn.setValue(currentInd);

}

Здесь выполняется следующее:

* Выполняется метод SetPerson(), чтобы записать в поля текущего объекта списка данные из текстовых полей формы.
* В список personList добавляется вновь созданный экземпляр класса Class2
* Присвоим значение переменной currentInd, которая хранит номер текущего объекта, значение делаем равным номеру последнего объекта списка.
* Копируем поля из созданного объекта на форму. Текстовые поля будут очищены, а в численных будет записан 0.
* Выводим количество объектов списке
* В счетчике jSpinnerCrn показываем номер текущего объекта в списке.

Аналогично создаем обработчик для кнопки jButtonDelPers

private void **jButtonDelPersActionPerformed**(java.awt.event.ActionEvent evt) {

personList.remove(currentInd);

currentInd = personList.size()-1;

if (currentInd == personList.size()) {

currentInd--;

jSpinnerCrn.setValue(currentInd);

} else {

GetPerson();

}

jTextFieldCount.setText("" + personList.size());

}

Процедура по разному ведет себя если удаляется не последний и последний элемент списка.

Для управления списком нужно также запрограммировать реакцию на изменение в счетчике JSpinnerCrn, счетчик указывает номер текущего элемента в списке. Нужно в окне свойств кнопки на закладке "***События***" щелкнуть на событие StateChanged. В сгенерированный шаблон добавляем свой код.

private void **jSpinnerCrnStateChanged**(javax.swing.event.ChangeEvent evt) {

SetPerson();

currentInd = (int) jSpinnerCrn.getValue();

if (currentInd >= personList.size()) {

currentInd = personList.size()-1;

jSpinnerCrn.setValue(currentInd);

} else if (currentInd < 0) {

currentInd = 0;

jSpinnerCrn.setValue(currentInd);

}

GetPerson();

}

Код работает следующим образом:

* Выполняется метод SetPerson(), чтобы записать в поля текущего объекта списка данные из текстовых полей формы.
* Присвоим новое значение переменной currentInd, которая хранит номер текущего объекта, значение. Значение берется из компонента jSpinnerCrn
* Проверяем значение переменной currentInd на ее допустимость:
  + номер текущего объекта не может быть больше или равен количеству элементов в списке
  + номер текущего объекта не может быть меньше нуля
* Выполняется метод GetPerson(), чтобы скопировать данных из текущего объекта списка в поля формы.

Все механизмы по работе со списком экземпляров класса Class2 теперь готовы, осталось реализовать код, при помощи которого мы покажем выполнение методов этих экземпляров.

## Реализация выполнения методов обработки данных Class1/ Class2

Код, обеспечивающий выполнение функции-метода 1 обработки данных класса-родителя связываем с обработчиком события для кнопки "***Увеличить оклад***" ( JButtonIncSalary).

private void **jButtonIncSalaryActionPerformed**(java.awt.event.ActionEvent evt) {

SetPerson();

for (int i = 0; i < personList.size(); i++) {

personList.get(i).IncSalary();

}

GetPerson();

}

Здесь выполняется следующее:

* Выполняется метод SetPerson для копирования в поля текущего объекта данных из текстовых полей формы.
* В цикле для каждого элемента списка выполняем метод IncSalary (Увеличить оклад)
* Выполняется метод GetPerson, чтобы показать измененный оклад

Выполнение функции-метода 2 обработки данных класса-родителя связываем с обработчиком события для кнопки ***"Иван в инженеры***" (JButtonPosition2Engieer)

private void **jButtonPosition2EngineerActionPerformed**(java.awt.event.ActionEvent evt) {

for (int i = 0; i < personList.size(); i++) {

if (personList.get(i).name.toUpperCase().startsWith("ИВАН")) {

personList.get(i).Position2Engieer();

}

}

GetPerson();

}

Особенностью здесь является то, что вызов метода Position2Engieer выполняется только тогда, когда фамилия работника начинается с сочетания букв «иван». Чтобы результат не зависел от регистра написания фамилии, преобразуем поле name к верхнему регистру.

Функция-метод обработки данных объекта класса-потомка выполняется при помощи обработчика события для кнопки "***Увеличить оклад по рейтингу***" (JbuttonChngRating). В цикле для каждого элемента списка выполняем метод ChngSalary (Изменить оклад работника в зависимости от его рейтинга)

private void **jButtonChngRatingActionPerformed**(java.awt.event.ActionEvent evt) {

SetPerson();

for (int i = 0; i < personList.size(); i++) {

personList.get(i).ChngSalary();

}

GetPerson();

}

## Тестирование приложения

Приведем последовательность действий по тестированию спроектированного приложения.

E:\Univer\Программирование и основы алгоритмизации Java\Методичка к лабораторным работам\Самостоятельная работа Java\вып-ть.bmpЧтобы запустить приложение, нажмите по проекту правой кнопкой мыши и выберите "***Запустить проект*** " из контекстного меню или нажмите на кнопку

Создаем и помещаем в список последовательно три экземпляра «Работник» (Class2), т.е . нажимаем на кнопку "***Работник***", после чего заполняем все поля на форме. Поля были заполнены так:

1й работник

* Фамилия - Иванов
* Должность - Техник
* Оклад- 10000
* Рейтинг - 70

2й работник

* Фамилия - Петров
* Должность - Техник
* Оклад- 10000
* Рейтинг – 80

3й работник

* Фамилия - Сидоров
* Должность - Техник
* Оклад- 10000
* Рейтинг - 90

Просмотреть все поля всех объектов можно пролистывая список при помощи счетчика jSpinnerCrn.

Теперь протестируем методы спроектированного класса.

Метод IncSalary – нажимаем на кнопку " ***Увеличить оклад***" - сразу после этого оклад у всех работников увеличится на15%, т.е. мы увидим что оклад после применения метода IncSalary будет 11500.

Метод Position2Engieer– нажимаем на кнопку "***Иван в инженеры***". Метод Position2Engieer будет выполнен для 1-го работника с фамилией Иванов. После выполнения Position2Engieer в поле «Должность» будет записано «Инженер»

Метод ChngSalary – нажимаем на кнопку ***"Увеличить оклад по рейтингу".*** Метод ChngSalary будет выполнен для всех объектов из списка. Оклад будет изменен в зависимости от поля «Рейтинг». В результате объекты в списке будут иметь следующий вид.

1й работник

* Фамилия - Иванов
* Должность - Инженер
* Оклад- 13800
* Рейтинг - 70

2й работник

* Фамилия - Петров
* Должность - Техник
* Оклад- 16100
* Рейтинг – 80

3й работник

* Фамилия - Сидоров
* Должность - Техник
* Оклад- 18400
* Рейтинг - 90