Лабораторна робота №10

Создание проекта

Первым действием является создание проекта среды IDE для разрабатываемого приложения. Дадим проекту имя NumberAddition.

1. Выберите Файл -> Создать проект. Также можно щелкнуть значок New Project на панели инструментов среды IDE.
2. В области Categories выберите узел "Java". В области "Projects" выберите "Java Application". Нажмите кнопку "Далее".
3. Введите NumberAddition в поле Project Name ("Имя проекта") и укажите путь, например, в вашем основном каталоге, как местоположение проекта.
4. Установите флажок "Использовать отдельную папку для хранения библиотек" и укажите местоположение папки библиотек (необязательно). Для получения дополнительных сведений об этой возможности ознакомьтесь с разделом Совместное использование библиотек проекта.
5. Удалите флажок "Create Main Class", если он установлен.
6. Нажмите кнопку "Завершить".

Упражнение 2: Создание внешнего интерфейса

Для продолжения процесса создания интерфейса необходимо создать контейнер Java, в который будут помещены другие требуемые элементы графического интерфейса. В этом действии контейнер будет создан с помощью элемента JFrame. Контейнер будет помещен в новый пакет, который будет отображаться в узле "Source Packages".

Создание контейнера JFrame

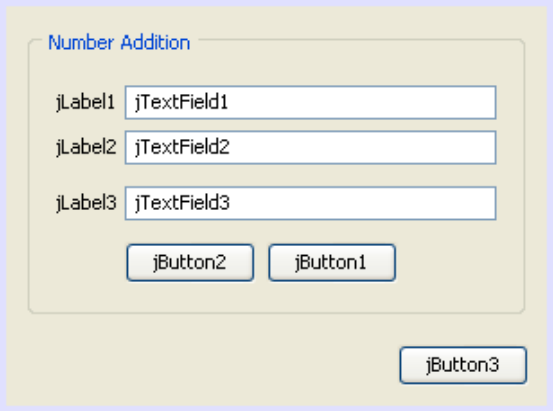
1. В окне 'Проекты' щелкните правой кнопкой мыши узел NumberAddition и выберите Создать > Другие.
2. В диалоговом окне создания файла выберите категорию Swing GUI Forms и тип файла JFrame Form. Нажмите кнопку "Далее".
3. Введите NumberAdditionUI в качестве имени класса.
4. Выберите пакет my.numberaddition.
5. Нажмите кнопку "Завершить".

Среда IDE создает форму NumberAdditionUI и класс NumberAdditionUI в приложении NumberAddition и открывает форму NumberAdditionUI в GUI Builder. Пакет my.NumberAddition заменяет собой пакет по умолчанию.

Добавление элементов: создание внешнего интерфейса

Далее с помощью окна "Palette" внешний интерфейс приложения заполняется панелью JPanel. После этого добавляются три элемента JLabel (текстовые подписи), три элемента JTextField (текстовые поля) и три элемента JButton (кнопки). Если до этого работа с конструктором графического интерфейса пользователя не выполнялась сведения о размещения компонентов см. в разделе Разработка графического пользовательского интерфейса Swing в IDE NetBeans.

После перетаскивания и размещения указанных выше элементов элемент JFrame должен выглядеть так, как показано на рисунке ниже.



Если в правом верхнем углу среды IDE отсутствует окно Palette ("Палитра"), выберите Window ("Окно") > Palette ("Палитра").

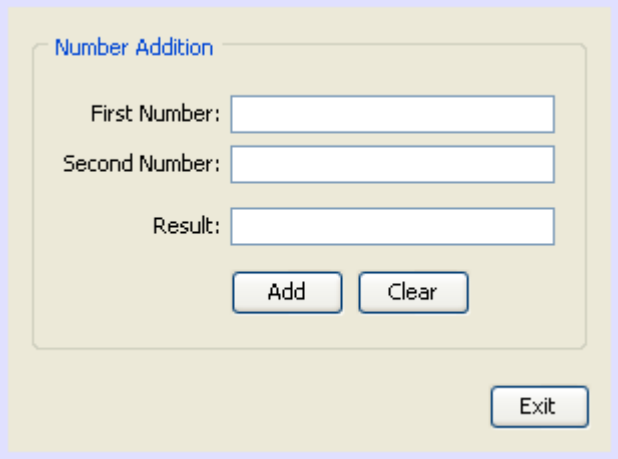
1. Для начала выберите панель из категории Swing Containers ("Контейнеры Swing") в палитре и перетащите ее на JFrame.
2. Панель JPanel будет выделена. Перейдите к окну "Properties" и нажмите кнопку с многоточием (...) рядом с полем "Border" для выбора стиля границы.
3. В диалоговом окне "Border" выберите "TitledBorder" из списка и введите Number Addition в поле "Title". Для сохранения изменений и закрытия диалогового окна нажмите кнопку "OK".
4. Теперь на экране должен отображаться пустой элемент "JFrame" с заголовком "Number Addition", как показано на рисунке. Согласно рисунку добавьте к нему три метки JLabel, три текстовых поля JTextField и три кнопки JButton.

Переименование элементов

На этом этапе будет выполнено переименование элементов, которые были добавлены к элементу JFrame.

1. Дважды щелкните jLabel1 и измените ntrcn (свойство "text") на First Number.
2. Дважды щелкните jLabel2 и измените текст на Second Number.
3. Дважды щелкните jLabel3 и измените текст на Result.
4. Удалите стандартный текст из jTextField1. Отображаемый текст можно преобразовать в редактируемый. Для этого щелкните правой кнопкой мыши текстовое поле и выберите 'Редактировать текст' во всплывающем меню. При этом может потребоваться восстановить первоначальный размер полеjTextField1. Повторите это действие для полей jTextField2 и jTextField3.
5. Измените отображаемый текст jButton1 на Clear. (Для изменения текста кнопки щелкните кнопку правой кнопкой мыши и выберите "Edit Text". В качестве альтернативы можно щелкнуть кнопку, выдержать паузу и щелкнуть еще раз.)
6. Измените отображаемый текст jButton2 на Add.
7. Измените отображаемый текст jButton3 на Exit.

Теперь готовый графический интерфейс должен выглядеть так, как показано на рисунке ниже:



Упражнение 3: Добавление функциональности

В этом упражнении будет добавлена необходимая функциональность к кнопкам "Add", "Clear" и "Exit". Поля jTextField1 и jTextField2 будут использоваться для ввода значений пользователем, а jTextField3 – для вывода результата работы программы. Создаваемая программа представляет собой простейший калькулятор. Итак, приступим!

Добавление функциональности к кнопке "Exit"

Для того чтобы кнопки стали функциональными, каждой из них необходимо присвоить обработчик событий, который будет отвечать за реагирование на события. В нашем случае требуется идентифицировать событие нажатия кнопки – путем щелчка мышью или с помощью клавиатуры. Поэтому будет использоваться интерфейс "ActionListener", предназначенный для обработки событий "ActionEvent".

1. Щелкните правой кнопкой мыши кнопку "Exit". Во всплывающем меню выберите Events ("События") > Action ("Действие") > actionPerformed. Учтите, что меню содержит множество других событий, на которые может реагировать программа! При выборе события actionPerformed среда IDE автоматически добавит прослушиватель ActionListener к кнопке Exit ("Выход") и создаст метод обработчика для обработки метода прослушивателя actionPerformed.
2. В среде IDE автоматически открывается окно "Source Code", где отображается место вставки действия, которое должно выполняться кнопкой при ее нажатии (с помощью мыши или клавиатуры). Окно "Source Code" должно содержать следующие строки:

```
private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {
    //TODO add your handling code here:
}
```

3. Теперь добавим код действия, которое должна выполнять кнопка "Exit". Замените строку TODO на System.exit(0);. Готовый код кнопки "Exit" должен выглядеть следующим образом:

```
4.      private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {
5.          System.exit(0);
        }
```

Добавление функциональности к кнопке "Clear"

1. Щелкните вкладку "Design" в верхней части рабочей области для возврата к экрану "Form Design".
2. Щелкните правой кнопкой мыши кнопку "Clear" (jButton1). В появившемся меню выберите "Events > Action > actionPerformed".

3. Нажатие кнопки "Clear" должно приводить к удалению всего текста из всех текстовых полей "jTextField". Для этого следует добавить код, аналогичный приведенному выше. Готовый исходный код должен выглядеть следующим образом:

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt){
    jTextField1.setText("");
    jTextField2.setText("");
    jTextField3.setText("");
                }
```

Этот код удаляет текст из всех трех полей JTextField, оставляя их пустыми.

Добавление функциональности к кнопке "Add"

Кнопка "Add" должна выполнять три действия.

1. Сначала она принимает данные, введенные пользователем в полях jTextField1 и jTextField2, и преобразовывает их из типа "String" в тип "Float".
2. Затем она выполнит сложение двух чисел.
3. И, наконец, она преобразует сумму в тип String и поместит ее в jTextField3.

Начнем!

1. Щелкните вкладку "Design" в верхней части рабочей области для возврата к экрану "Form Design".
2. Щелкните правой кнопкой мыши кнопку "Add" (jButton2). Во всплывающем меню выберите Events ("События") > Action ("Действие") > actionPerformed.
3. Добавьте код действий, которые должна выполнять кнопка "Add". Готовый исходный код должен выглядеть следующим образом:

```
private void jButton2ActionPerformed(java.awt.event.ActionEvent evt){
    // First we define float variables.
    float num1, num2, result;
    // We have to parse the text to a type float.
    num1 = Float.parseFloat(jTextField1.getText());
    num2 = Float.parseFloat(jTextField2.getText());
    // Now we can perform the addition.
    result = num1+num2;
    // We will now pass the value of result to jTextField3.
    // At the same time, we are going to
    // change the value of result from a float to a string.
    jTextField3.setText(String.valueOf(result));
                }
```

Теперь программа полностью готова, и можно приступить к ее сборке и выполнению.

Упражнение 4: Выполнение программы

Для выполнения программы в среде IDE выполните следующие действия:

1. Выберите Run ("Запуск") > Run Main Project ("Запуск главного проекта") (как вариант, нажмите F6).

Примечание. При открытии окна с указанием того, что для Project NumberAddition не задан основной класс, следует выбрать my.NumberAddition.NumberAdditionUI в качестве основного класса в том же окне и нажать кнопку ОК.

Для запуска программы вне среды IDE выполните следующие действия:

1. Для сборки архива JAR приложения выберите "Run > Clean and Build Main Project" (Shift-F11).
2. При помощи проводника по файловой системе или диспетчера файлов перейдите в каталог NumberAddition/dist.

Примечание. Местоположение каталога проекта NumberAddition зависит от пути, указанного при создании проекта в шаге 3 в разделе

Упражнение 1. Создание проекта.

3. Дважды щелкните файл NumberAddition.jar.

Через несколько секунд приложение запустится.

Примечание. Если при двойном щелчке файла JAR не выполняется запуск приложения, дополнительные сведения о настройке связей файлов JAR в используемой операционной системе см эту статью.

Можно также запустить приложение из командной строки.

Для запуска приложения из командной строки выполните следующие действия:

1. Вызовите командную строку или окно терминала.
2. В командной строке измените текущий каталог на каталог NumberAddition/dist.
3. В командной строке введите следующий оператор:

```
java -jar  NumberAddition.jar
```

Примечание. Убедитесь, что my.NumberAddition.NumberAdditionUI задан как основной класс до запуска приложения. Для провери этого, щелкните правой кнопкой узел мыши узел проекта NumberAddition на панели 'Проекты', выберите 'Свойства' во всплывающем меню и выберите категорию 'Выполнить' в диалоговом окне 'Свойства проекта'. В поле 'Основной класс' должно отображатьсяmy.numberaddition.NumberAdditionUI.

Механизм обработки событий

В этом руководстве было рассмотрено реагирование на простое событие нажатия кнопки. Существует множество событий, на которые может реагировать приложение. Просмотреть в среде IDE список доступных событий, которые могут обрабатываться элементами графического интерфейса, можно следующим образом:

1. Вернитесь к файлу NumberAdditionUI.java в редакторе. Щелкните вкладку "Design" для просмотра структуры графического интерфейса в GUI Builder.
2. Щелкните правой кнопкой мыши любой элемент графического интерфейса и выберите "Events" в появившемся меню. Теперь можно просто изучить содержимое меню, не выбирая каких-либо пунктов.
3. В качестве альтернативы можно выбрать "Properties" в меню "Window". В окне "Properties" щелкните вкладку "Events". Вкладка "Events" позволяет просмотреть и изменить обработчики событий, связанные с текущим активным элементом графического интерфейса.
4. Приложение также может реагировать на нажатие клавиш, одинарный, двойной или тройной щелчок мышью, перемещение указателя мыши, изменение размера окна и перемещение фокуса ввода. Меню "Events" позволяет автоматически создать обработчики событий для всех этих событий. Наиболее распространенным из них является событие "Action". (Для получения дополнительных сведений см. практические рекомендации по обработке событий в руководстве Sun Java Events Tutorial.)

Как выполняется обработка событий? При каждом выборе события из меню событий среда IDE автоматически создает так называемый прослушиватель событий и связывает его с компонентом разработчика. Для более подробного ознакомления с процессом обработки событий выполните следующие действия.

1. Вернитесь к файлу NumberAdditionUI.java в редакторе. Щелкните вкладку "Source" для просмотра исходного кода графического интерфейса.

2. Выполните прокрутку вниз и просмотрите реализованные методы jButton1ActionPerformed(), jButton2ActionPerformed() иjButton3ActionPerformed(). Эти методы называются обработчиками событий.
3. Теперь перейдите к методу initComponents(). Если этот метод отсутствует, найдите строку Generated Code и щелкните знак + рядом с этой строкой для отображения скрытого метода initComponents().
4. Обратите внимание на синий блок, окружающий метод initComponents(). Этот код был автоматически создан средой IDE и не может быть изменен пользователем.
5. Теперь посмотрите на сам метод initComponents(). Помимо прочего, он содержит код, инициализирующий элементы графического интерфейса и помещающий их в форму. Этот код создается и обновляется автоматически при размещении и изменении элементов в режиме проектирования.
6. В методе initComponents() найдите следующий фрагмент:

```
jButton3.setText("Exit");
jButton3.addActionListener(new java.awt.event.ActionListener() {
   public void actionPerformed(java.awt.event.ActionEvent evt) {
        jButton3ActionPerformed(evt);
   }
       });
```

В этом месте к элементу графического интерфейса, в данном случае к jButton3, добавляется объект прослушивания событий "ActionListener". Интерфейс "ActionListener" имеет метод "actionPerformed" объекта "ActionEvent", который реализуется путем простого вызова обработчика событийjButton3ActionPerformed. Теперь эта кнопка реагирует на события действий. Каждый раз при нажатии кнопки создается событие "ActionEvent", которое передается в метод "actionPerformed" интерфейса прослушивания событий, исполняющий код, предусмотренный разработчиком для этого события в обработчике событий.
Как правило, для получения возможности реагирования каждый интерактивный элемент графического интерфейса должен быть зарегистрирован в каком-либо интерфейсе прослушивания событий и иметь связанный обработчик событий. Как наглядно показано, IDE NetBeans автоматически обрабатывает подключение прослушивателя событий, что обеспечивает для пользователей возможность сосредоточиться на реализации фактической бизнес-логики, которая должна инициироваться событием.