Графический интерфейс

- Для работы с созданием GUI используется Swing
- Далее - описание библиотеки swing
- Карскас библиотеки - компененты JavaBean
- Компоненты JavaBean и модель библиотеки Swing позволяют генерировать код gj vtht lj,fdktybz yjds[rjvgjytynjd
- Даже при ручном построении программы полчается удобочитаемый код
- Swing содержит все для создания современного строительного интерфейса
- В компоненты встроено добавление с клавиатуры
- Возможно динамическое изменение GUI

- Аплеты - програмы способные пересылаться по интернету для выполнения в браузере
- Не оправдали ожиданий, тк не у всех утсановлена jre
- Основы swing
- большинство приложений Swing - строится на основе простейшего объекта JFrame
- представляет из себя окно, заголовок в конструкторе, размер и проч через методы
- моно установить дефолтную операцию
- setSize - размер окна в пикселях

- Взаимодействия с компонентами GUI из метода main нежелательны
- Swing создает отдельный программный поток для получения
- событий пользовательского интерфейса и обновления экрана

- При изменении экрана из других потокв возможны конфликты с взаимными блокировками
- Потоки (main) - должны передавать выполняемые задачи потоку диспетчеризации событий Swing

- Нет необходимости програмно прорисовывать компоненты
- Перехват событий
- Необходим дополнительный код обрабатывающий действия по нажатию и проч
- Присутствует четкое отделение интерфейса от реализации
- Каждый компонент библиотеки Swing способен сообщить обо всех событиях, которые могут с ним произойти
- сли какое-то событие не нужнопросто не определяете его
- Для регистрации события необходимо добавить класс addActionListener с методом actionPerformed()
- Для присоединения кода обработки главного события нужно реализовать ActionListener в своем классе

- Вспомогательный класс (например SwingConsole)
- Содержит статический метод run
- Чтобы использовать его приложение должно выполняться в окне

- Создание кнопки
- JButton и текст и возможно изображение
- Имплементируем ActionListener и создаем на его основе класс в ктором описываем действия
- При создании кнопки вызываем метод addActionListener добавляем в аргументы объект сохданного класса
- Объект ActionEvent - объект, реагирующий на события

- Текстовые области
- JTextArea похожа на однострочное поле, позволяет редактировать несколько строк текста, имеет
- расширенные возможности
- append - метод, позволяющий переводить стандартный вывод в текстовое поле, выведенные сообщения не
- исчезают
- Управление расположением компонентов

- Нет ресурсов управляющих расположением компонентов. Расположение компонентов на форме определяется
- менеджером объектов, который решает, где будут находиться компоненты - зависит от времени добавления с помощью add
- Графические компоненты представленны классом Component. метод setLayout - позволяет выбрать
- произвольный менеджер компоновки. - по умолчанию BorderLayout - помещение компонента в центре, растягивая границы к сторонам панели
- можно перегрузить и использовать константу для выравнивания по границам
- В случае размещения по центру компонент растягивается в обоих направлениях, в попытке полностью занять контейнер
- FlowLayout - последовательно выкладывает компоненты на форму, пока не заполнит всю строку
- и переходит к следующей. Все компонены сжимаются до наименьшего из возможных размеров
- GirdLayout - менеджер расположения позволяющий построить таблицу компонентов (необходимо указывать количество строк
- и столбцов) с одинаковыми размерами нет оптимизации расположения
- GridBagLayout - позволяет подробно указать как должны располагаться области окна
- Основное предназначение - генерироваие кода
- для более понятного использования - GridBagLayout
- Абсолютное позиционирование
- Для используемого вами контейнера вместо менеджера расположения передать методу setLayout ссылку null
- Для каждого компонента вызвать setBounds или reshape передать прямогульник компонента в пикселях
- Можно сделать в конструкторе или методе paint
- BoxLayout - создан для использования GridBagLayout без его сложностей. Позволяет размещать
- объекты вертикально и горизонтально, позволяет управлять пространством между компонентами.
- Вместо установки граф инт с помощью кода можно использовать визуальный инструмент

- Модель событий библиотеки Swing
- Любой графический компонент может инициализировать событие. Каждое событие представлено определенным классом.
- Событие принимается слушателями, которые его обрабатывают
- Слушатель каждого события - объект класса, реализующий определенный интерфейс, характерный
- для всех слушателей события
- регистрация - add*типсобытия*Listener(). JavaBeans также использует addXXXlistener для
- определения какие события поддерживает компонент JavaBean
- Вся логика обработки события будет сосредоточена в классе слушателя. При создании класса слушателя
- необходимое условие - реализация подходящего интерфейса. удобно использовать
- внутренние классы, тк группировка слушателей в опр метсе программы, обладают ссылкой на внешние классы, что позволяет
- им легко преодолевать границы классов и подсистем программы
- Каждый компонент поддреживает определенный набор событий
- Обработка событий
- 1.Узнать название событие и реализовать интерфейс Listener
- 2.Написать метод для событий, которые вы хотите обработать.

- Адаптеры слушателей - используются при реализации интрефейсов, содержащих больше методов,
- чем может понадобиться
- Некоторый интерфейсы слушателей снабжены адаптерами
- Слушатель событий для мыши - MouseAdapter
- При наследовании от адаптера переопр только методы, которые вы хотите изменить
- Адаптер класс - остальные методы уже определены

- Отслеживание нескольких событий
- Проследим за несколькими событиями кнопки Jbutton

- Значки
- можно использовать значок Icon внутри JLabel или в компонентах, унаследованных от базовой кнопки AbstractButton
- можно использовать любые файлы в формате GIF. Вызвать объект ImageIcon и передать конструктору имя
- файла с изображением . Значки можно установить как в конструкторе так и после создания кнопки

- Подсказки
- Все классы, унаследованные от JComponent имеют метод setTooTipText(String)
- Когда пользователь задержит курсор у компонента появиться подсказка

- Текстовые поля
- Можно добавлять убирать возможность редактирования, проверки выделение текста, вносить собственную строку

- Рамки
- метод setBorder() - позволяет разместить на компонентах различные окантовки

- Мини-редактор
- JTextPane - возможности по редактированию текстов - можно получать текст, добавлять текст, переходить на следующую сироку
- и проч

- Флажки
- Флажок позволяет произвести выбор из двух возможных состояний (вкл.выкл)
- JCheckBox - создается конструктором, которому передается текст надписи
- Когда пользователь изменяет сотояние флажка, происходит соотвествтующее событие

- Переключатели
- Радиокнопки - только одина работает из серии
- Чтобы создать надо объединить кнопки в группу ButtonGroup

- Раскрывающиеся списки
- Ракрывающиеся списки предназначены для того, чтобы пользователь смог выбрать один элемент из группы значений
- Чтобы получить список такого типо необходимо вызвать метод setEditable() у JComboBox

- Списки
- JList. В отличие от ComboBox виден целиком сразу. Если нужны результыты выбора у объекта вызывается метод
- getSelectedValues(). Список элементов можно создать автоматически передав передав массив конструктору класса JList
- В списках JList отсутствует автоматическая прямая поддержка прокрутки

- Панель вкладок
- Позволяет создать диалоговое окно с набором вкладок, у которого к одной из сторон прижат набор корешков вкладок
- Чтобы переключится к содержимому другого диалогового окна, нужно нажать на корешке нужной вкладки

- Окна сообщений
- Оконные среды часто содерат стандартный набор окон сообщений, который позволяет информировать пользователей или
- получать от него информацию
- Окна сообщений (и окна подтверждения выбора и проч) создаются классом JOptionPane
- + showConfirm, showOption, showInput

- Меню
- Все компоненты способные отражать меню - JApplet, JFrame, JDialog и их потомки содержат
- метод setJMenuBar(JMenuBar-в количестве одно штуки). В Java и Swing меню формируется в коде программы
- JMenuBar - можно добавлять меню JMenuItem - производное от AbstractButton. Есть разные варианты

- Бибилотека Swing поддерживает клавиатурные сокращения. Можно выбрать любой компонент, унаследованный от базовой кнопки
- AbstractButton без использование мыши. В JMenuItem существует перегруженный конструктор, во втором аргуметне передается индентификатор клавиши
- Можно использовать общий способ устанвки мнемоники - вызов метода setMnemonic()

- Можно использовать setActionCommand()

- Всплывающее меню
- Всплывающее меню обычно реализуется следующим образом
- создается внутренний класс, объект внутреннего класса добавляется ко всем компонентам

- Диалоговое окно - окно, создающеяся другим окном
- Для создания нужно использовать наследование от класса Jdialog который представляет
- собой еще один вид окна Window. Размещение определяется менеджером расположения. После создания
- следует вызвать метод setVisible(true), который выводит его на экран и активизирует (
- можно добавить это действие как результат onAction)

- Диалоговые окна выбора файлов
- JFileChooser. Можно запрашивать информацию о выбраном файе и его директории

- Все компоненты могут получать данные в формате HTML. Текст должен начинаться с тэга
- <html>, за которым могут следовать другие

- Регуляторы и индикаторы выполнения
- JSlider - позволяет пользователю вводить данные, перемещая ползунок по шкале.
- JProgressBar - показывает данные в сравнении с эталоном(помогает оценить состояние выполнения)
- Для того, чтобы оценить два компонента, достаточно использоваь одну и туже модель поведения

- Выбор внешнего вида и поведение программы
- Модульный интерфейс пользователя позволяет вашей программе эмулировать внешний вид и поведение
- различных операционных систем
- Можно динамически изменять визуал программы
- Есть два пути - оставить платформно-независимый интерфейс, либо наследовать интерфейс системы, использующей
- эту программу.
- Если наследовать интерфейс ОС, то - try{UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName())
- JNLP и JavaWebStart
- Аплет можно снабдить цифровой подписью по соображением безопасности, а можно использовать
- Протокол запуска Java по сети.
- JNLP решает проблему, не лишая примуществ апплетов
- Приложение способно динамически получать данные из Интернета и автоматически проверять версию
- JNLP получает доступ к ресурсам клиентской системы даже в случае неподписанного jar файла
- Для успешного запуска программы необходимо изменить url сервера так, чтобы он обозначал соответсвующий каталош
- на вашей href - должен указывать имя файла запуска

- Параллельное выполнение и Swing - есть поток диспетчеризации событий
- Случайное использование потока диспетчеризации событий для выполнения продолжительной задачи
- Объект Executor - автоматически ставит оидающие задачи в очередь
- Объект TaskManager - ArrayList с объектами TaskItem

- Чтобы предоставить пользователю визуальные признаки того, что задача продолжает выполняться
- используют progressBar или ProgressMonitor

- Визуальные потоки
- Рассмотрены на примере Jpanel с сеткой. Получает из командной строки значения размера сетки
- цветов и продолжительность задержки между изменениями
- В ходе работы можно обнаружить особенности реализации многопоточности
- В методе run - бесконечный цикл, который присваивает cColor новый случайный цвет, затем
- вызывает repaint() для отображения
- Программа демонстрирует радикальные различия в производительности и поведении между разными реализациями JVM
- на разных платформах

- Визуальное программирование и компоненты JavaBean
- Класс является еденицей многократного использования
- При создании приложения нужны компоненты, которые делают то, что нужно

- Характеристики компонента - свойства
- Действия визуального компонента - события
- Построитель приложений использует механизм отражения, чтобы динамически исследовать
- компонент и определить, какими свойствами он обладает и какие события поддреживает
- Среда разработки выполняет значительную часть работы

- Компонент JavaBean
- Технология JavaBeans, где визуальный компонент Bean - простой класс.
- Чтобы получить готовый компонент Bean необходимо изменить
- способ именования методов своих классов
- Соглашение о записи имен
- Для свойства с именем xxx создаются методы - get set
- Для boolean вместо get можно исп is
- Обычные метода должны быть объявлены как открытые
- Для создания библиотеки используются Swing

- Все поля класса - закрытые и изменяются с помощью методов и свойств
- Объект Bean - обыкновенный класс
- Компонент Bean позволяет обработывать события ActionEvent и KeyEvent -
- для этого были добавлены методы add и remove

- Получение информации о компоненте Bean - инструмент Introspector
- Один из важных моментов механики визуал компонентов - выборка цвета и помещение на форму
- Построитель приложений должен создать экземпляр выбранного
- компонента а затем извлечь всю необходимую информацию о списке свойств и событий
- Introspector инструмент, упрощает применение bean и предоставляет
- механизм создания более сложных компонентов

- Важный элемент - метод getBeanInfo() - возвращает объект BeanInfo из которого можно
- получить информацию о свойствах методах и событиях

- Второй аргумент в getBeanInfo сообщает в каком месте иерархии объекта надо остановиться
- Имена методов, которые вы здесь видите на самом деле получены из объектов Method
- Выдается информация о слушателе, о методах для добавления и удаления слушателя

- Пример более сложного компонента Bean - Панель JPanel - рисует небольшую окружность вокруг
- курсора мыши при ее передвижениях
- При нажатии кнопки в центре надпись
- Настраиваемые свойства - диаметр окружности
- Цвет текста
- Размер шрифта
- Сам текст
- Также определены слушатели, что похволяет обработать нажатие кнопки мыши
- Класс BangBean реализует интерфейс Serializable
- Возбуждение исключения TooManyListener говорит о том, что метод подерживает одноадресных
- слушателей
- Обычно создаются многоадресные события, но тогда нужно учитывать вопрос многозадачнос

- При создании компонента Bean всегда следует учитывать возможность использования в многозадачном окружении
- ->
- Все открытые методы компонента должны быть синхронизированными
- Методы, не создающие проблем, моно оставить без синхрониз, но не всегда понятно
- какие из них какие. Чаще всего таковыми являются атомарные методы или просто небольшие
- При инициировании многоадресного события слушателем нужно иметь в виду возможность добавления или удаления слушателей
- из списка во время перемещения по списку
- Проблема синхронизации решается также по средствам реализации одноадресных событий
- Добавить к методам ключевое слово synchronized было несложно
- Для определение необходимости синхронизации переопределенных методов нужно учитывать следующие факторы
- Изменяет ли метод состояние критических (используемых для чтения\записи) переменных
- Зависит ли метод от состояния критических переменных
- Синхронизирован ли переопределенный метод в базовом классе (синхронизация не передается по
- наследству)
- Насколько важна скорость выполнения (синхронизация замедляет)

- Упаковка компонента Bean
- Чтобы объект можно было использовать в системе визуальной разработки необходимо упаковать
- его в стандартный контейнер - файл формата JAR который содержит манифест утверждающий
- что данный класс - компонент Bean
- Файл манифеста - текстовый файл имеющий определенную структуру
- Первая строка - содержит версию используемой схемы манифеста
- вторая - имя файла - путь имя класса должно включать имя пакета
- Третья - Java-Bean True - отвечает за распознование класса как визуального компонента
- При распоковке файла Jar получите другой вариант манифеста
- К каждому файлу добавлена цифровая подпись
- После помещения бина в джар можно будет загрузить его в среде разраб с поддержкой JavaBean

- Поддержка более сложных компонентов Bean
- Одно из направлений повышения гибкости - свойства компонента
- Рассмотренные примеры использовали одиночные свойства, однако допустимы массивы свойств -
- Определить подходящие методы, использовать Introspector для распоз индексированных свойств

- Свойства могут быть связанными друг с другом (bound) -> будут оповещать другие объекты о своих
- изменениях
- Свойства могут быть ограниченными -> другие объекты вправе запрещать изменение свойства

- Возможно переопр способ представления компонента Bean
- Можно предоставить сственный список свойств для определенного компонента Bean
- Можно создать специальный редактор для определенного свойства - будет использоваться
- обычный список свойств компонента, но при редактировании будет запускаться ваш редактор
- Разрешается применять собственный класс с информацией о компоненте beanInfo, отличный от
- Introspector
- Можно включить и выключить режим эксперта -> отделение основных свойств компонента от
- доп возможностей

- Приложение SWT как аналог Swing
- Использование родных компонентов ОС и синтез отсутствующих
- Возможна установка с сайта Eclipse или установка самого редактора Eclipse