Печатных листов: 130/16\*0,93=7,55

План

1. ОБЗОР И СРАВНИТЕЛЬНАЯ ХАРАКТЕРИСТИКА МОБИЛЬНЫХ ПЛАТФОРМ
   1. Операционная система IOS
   2. Андроид
   3. Windows Phone
   4. Symbian
   5. Особенности разработки мобильных приложений
2. ПРИЕМЫ ОБЪЕКТНОГО ПРОГРАММИРОВАНИЯ НА JAVA (JAVA для Android-разработчиков)
   1. Перед началом работы. Установка компонентов среды разработки
   2. Лабораторная работа №1. Работа в командной строке - компиляция и запуск на выполнение.
   3. [Лабораторная работа №2. Разработка программы в NetBeans, простейшие UML-диаграммы. Наследование и реализация полиморфизма в Java](#_Разработка_программы_в)
   4. Лабораторная работа №3. Разработка и использование интерфейсов.
   5. Лабораторная работа №4.  [Библиотеки AWT и Swing для построения графического интерфейса пользователя. Обработка событий.](#_3.3.__Библиотеки)

[2.4.1. Общие сведения о графических интерфейсах](#_4.1._Общие_сведения)

* 1. [Лабораторная работа №5. Задачи многопоточност](#_Многопоточность.)и в JAVA

1. РАЗРАБОТКА МОБИЛЬНЫХ ПРИЛОЖЕНИЙ НА ПЛАТФОРМЕ ANDROID
   1. Перед началом работы. Установка компонентов среды разработки
   2. Лабораторная работа №1А. Разработка первого Android-приложения

## Создание нового Android-приложения

## Структура Android-приложения

## Позиционирование элементов (Layout)

## [Задание к ЛР№1](#_1A.4._Задание_к)А

* 1. [Лабораторная работа №2А. Отслеживание состояний Активности](#_2A.1._Теоретическая_часть.)

## Теоретическая часть. Android –Activity

## Создание Активности

## Жизненный цикл Активности

## Обработка событий

## [Задание к ЛР№](#_1A.4._Задание_к)2А

* 1. [Лабораторная работа №3А. Работа с ресурсами](#_3A.1._Теоретическая_часть.)

## Теоретическая часть. Ресурсы

## Изображения, анимация, меню

## Локализация приложения с помощью внешних ресурсов

## [Задание](#_3А.5._Задание_3.2.) к ЛР№3А

* 1. [Лабораторная работа №4А. Адаптеры в Андроид](#_3A.1._Теоретическая_часть.)

## Теоретическая часть. Ресурсы

## Изображения, анимация, меню

## Локализация приложения с помощью внешних ресурсов

## [Задание](#_3А.5._Задание_3.2.) к ЛР№4А

* 1. [Лабораторная работа №5А. Намерения. Запуск новой Активности.](#_3A.1._Теоретическая_часть.)

## Теоретическая часть. Ресурсы

## Изображения, анимация, меню

## Локализация приложения с помощью внешних ресурсов

## [Задание](#_3А.5._Задание_3.2.) к ЛР№5А

* 1. [Лабораторная работа №6А. Фрагменты](#_3A.1._Теоретическая_часть.)

## Теоретическая часть. Ресурсы

## Изображения, анимация, меню

## Локализация приложения с помощью внешних ресурсов

## [Задание](#_3А.5._Задание_3.2.) к ЛР№6А

* 1. [Лабораторная работа №7А. Хранилища данных. Работа с базой данных SQLite](#_3A.1._Теоретическая_часть.)

## Теоретическая часть. Ресурсы

## Изображения, анимация, меню

## Локализация приложения с помощью внешних ресурсов

## [Задание](#_3А.5._Задание_3.2.) к ЛР№7А

* 1. [Лабораторная работа №8А. Датчики в системе Android](#_3A.1._Теоретическая_часть.)

## Теоретическая часть. Аппаратные и программные датчики

## Изображения, анимация, меню

## Локализация приложения с помощью внешних ресурсов

## [Задание](#_3А.5._Задание_3.2.) к ЛР№7А

AWT Event-Handling

Java принимает так называемую модель программирования «Event-Driven» (или «Event-Delegation») для обработки событий, аналогичную большинству языков визуального программирования, таких как Visual Basic.

В программировании, управляемом событиями, часть кодов обработки событий выполняется (или вызывается графической подсистемой), когда событие вызывается в ответ на ввод пользователя (например, нажатие кнопки мыши или нажатие клавиши ВВОД в тексте). поле).

В приведенных выше примерах метод actionPerformed () известен как метод обратного вызова. Другими словами, вы никогда не вызываете actionPerformed() в своих кодах явно. ActionPerformed() вызывается графической подсистемой при определенных обстоятельствах в ответ на определенные действия пользователя.

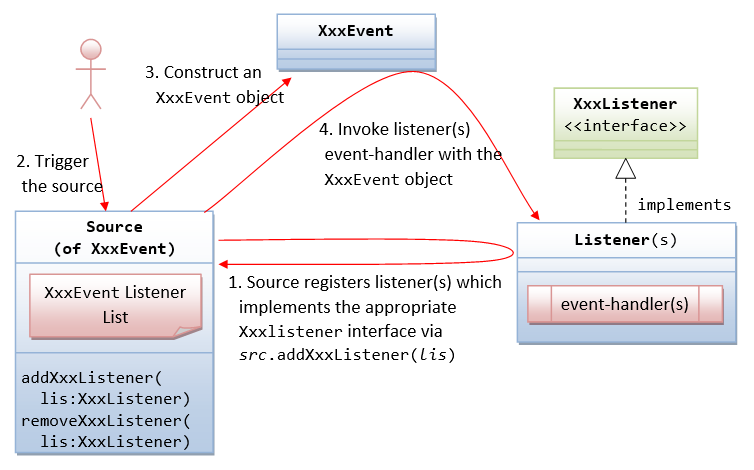
Объекты Source, Event и Listener

Классы обработки событий AWT хранятся в пакете java.awt.event.

Три типа объектов участвуют в обработке событий: источник, слушатель (и) и объект события.

Исходный объект (например, кнопка и текстовое поле) взаимодействует с пользователем. После запуска исходный объект создает объект события для захвата действия (например, щелчок мышью по x и y, введенные тексты и т. Д.). Этот объект события будет передан всем зарегистрированным объектам (объектам) прослушивателя, и соответствующий метод обработчика события (ов) прослушивателя будет вызван для обеспечения ответа. Другими словами, запуск источника запускает событие для всех его слушателей и вызывает соответствующий обработчик событий слушателей.

Чтобы выразить интерес к событию определенного источника, слушатели должны быть зарегистрированы в источнике. Другими словами, слушатель (и) «подписывается» на событие источника, а источник «публикует» событие всем своим подписчикам при активации. Это известно, как шаблон оформления подписки-публикации или наблюдения-наблюдателя.



1. Источник регистрирует слушателя (ей), который реализует (имплементирует) XxxListener интерфейс путем вызова метода src.addXxxListener(lis)
2. Вызов слушателя(ей) и его обработчика event-handler в ответ на событие XxxEvent
3. Событие

**XxxEvent**

**Источник события**

**(XxxEvent)**

addXxxListener( lis:XxxListener)

removeXxxListener( lis:XxxListener)

XxxEvent Listener

List

implements

**XxxListener**

**<<interface>>**

**Listener(s)**

**event-handler(s)**

обработчик события

1. Создание объекта XxxEvent