Міністерство освіти і науки України

Житомирський державний технологічний університет

Кафедра інженерії програмного забезпечення

**Є.О.Гришкун, С.М. Кравченко, Сугоняк І.І.**

Методичні рекомендації

до виконання лабораторних робіт   
з дисципліни «Людино-машинний інтерфейс»

Житомир

2019

УДК 004.65

Рекомендовано до друку Навчально-методичною радою Житомирського державного технологічного університету (протокол № 3 від 17.02.2017 р.)

**Рецензенти:**

Зав. кафедри комп’ютерної інженерії Житомирського державного технологічного університету к.т.н. Єфіменко А.А.

Доцент кафедри програмного забезпечення систем, к.т.н., доцент Грабар О.І.

Сугоняк І.І. Методичні вказівки до виконання лабораторних робіт з дисципліни "Людино-машинна взаємодія"/ Є.О.Гришкун, А.М. Ковальчук,

І.І. Сугоняк – Житомир: ЖДТУ, 2019. – 55 с.

У методичній розробці наведено рекомендації щодо виконання лабораторних робіт, теоретичні відомості, вимоги до звітності, контрольні питання, завдання та приклади виконання робіт. Методичні рекомендації призначені для студентів, що навчаються за напрямком 121 «Інженерія програмного забезпечення» денної та заочної форм навчання.

Зміст

[Вступ 5](#_Toc483387315)

[Лабораторна робота № 1 6](#_Toc483387316)

[Лабораторна робота № 2 20](#_Toc483387325)

[Лабораторна робота № 3 29](#_Toc483387330)

[Лабораторна робота № 4 50](#_Toc483387345)

# Вступ

Метою викладання навчальної дисципліни “Людино-машина взаємодія” є надання майбутнім фахівцям знань про сучасні концепції, методи та засоби створення інтерфейсів прикладного програмного забезпечення автоматизованих інформаційних систем на базі використання різноманітних сучасних програмних засобів.

Основними завданнями вивчення дисципліни “Людино-машина взаємодія” є формування сукупності знань та вмінь для створення інтерфейсів прикладного програмного забезпечення автоматизованих інформаційних систем на базі використання різноманітних сучасних програмних засобів.

Програма вивчення навчальної дисципліни “Людино-машина взаємодія” складена відповідно до освітньо-професійної програми підготовки за напрямом 121 «Інженерія програмного забезпечення».

Програма навчальної дисципліни складається з таких змістових модулів:

1. Психологічні принципи людино-машинної взаємодій
2. Функціональні компоненти та властивості людино-машинного інтерфейсу
3. Засоби розробки людино-машинного інтерфейсу
4. Оцінювання якості людино-машинного інтерфейсу

Методичні рекомендації містять 4 лабораторні роботи. Вони можуть бути використані для самостійного освоєння практичної складової курсу «Людино-машинна взаємодія», під час підготовки до складання підсумкового контролю з дисципліни та як доповнення до лекційних курсів, що викладаються у вищих навчальних закладах.

# Лабораторна робота № 1

**Тема роботи:** Інтерфейс віконного додатку.

**Мета роботи:** Дослідження основних підходів та технологій реалізації інтерфейсу універсального додатку Windows.

Обладнання: ПК, Visual Studio не нижче v.2010 (.Net Framework 4.5)

## 1. Завдання на лабораторну роботу

Засобами інтегрованого середовища розробки Visual Studio побудувати текстовий редактор. Потрібно реалізувати наступні функції:

* 1. Створення, відкриття та збереження файлів
  2. Копіювання, переміщення тексту.
  3. Виділення тексту та знищення тексту.
  4. Відміну останньої дії.
  5. Стрічку, меню та меню швидкого доступу для управління функціональними компонентами додатку.
  6. Персоналізувати параметри проекту (назву додатку, оригінальну піктограму, вікно AboutBox тощо).
  7. Виконати індивідуальне завдання за варіантом.
  8. Додатково виконати завдання на самостійну роботу.

## Індивідуальні завдання

1. Додати Сontrol та діалогове вікно що дозволяє зміну вирівнювання виділеного фрагмента тексту.
2. Додати Сontrol та діалогове вікно що дозволяє зміну відступу від лівого краю першого рядка виділеного фрагмента тексту.
3. Додати пункт Сontrol та діалогове вікно що дозволяє встановити маркери до виділеного фрагмента тексту.
4. Додати пункт меню та кнопку що дозволяє встановити нумерований список до виділеного фрагмента тексту.
5. Додати пункт меню та кнопку що дозволяє зміну відступу від лівого краю виділеного фрагмента тексту.
6. Додати пункт меню та кнопку що дозволяє зміну відступу від правого краю виділеного фрагмента тексту.
7. Додати пункт меню та кнопку що дозволяє зміну інтервалу перед та після фрагменту тексту.
8. Додати пункт меню та кнопку що дозволяє визначити позицію табуляції в тексті.
9. Додати пункт меню що визначає позицію курсору та відображує її у командному рядку.

10. Додати кнопки що дозволяють вставляти часто використані символи грецького алфавіту в текст (α,β ,µ).

11. Реалізувати пошук символів в тексті.

12. Реалізувати заміну символів в тексті.

13. Колір фону.

14. Колір тексту.

15. Границі тексту.

## Індивідуальні завдання підвищеного рівня складності

1. Додати кнопку, що визначає параметри форматування фрагменту тексту та відображує її у діалоговому вікні.
2. Реалізувати друк документів з редактору.
3. Реалізувати кнопку відміни форматування для виділеного фрагменту тексту.
4. Реалізувати журнали операцій для відміни та повернення останньої дії.
5. Вставку об’єкта.
6. Багаторівневий список.
7. Колонтитули.

## Завдання на самостійну роботу

* 1. Реалізувати MDI інтерфейс текстового редактора
  2. реалізувати системи контекстних меню додатку
  3. реалізувати довідникову систему додатку.

## 2. Хід роботи

Взаємодія користувача з комп’ютером шляхом використання ним додатків для виконання своїх функціональних задач. Фокусом взаємодії в таких випадках є інтерфейс додатку.

Класичний інтерфейс WIMP додатку має наступні елементи:

* *робоча область:* внутрішня частина вікна, призначена для вдображення інформації та виконання завдань користувача;
* *границі:* рамка, що обмежує вікно із чотирьох сторін. Розміри вікна можна змінювати, переміщаючи границю мишею;
* *заголовок:* рядок безпосередньо під верхньою границею вікна, що містить назву вікна;
* *значок системного меню:* кнопка ліворуч у рядку заголовка відкриває меню переміщення й зміни розмірів вікна;
* *рядок горизонтального меню:* розташовується безпосередньо під заголовком, містить пункти меню, забезпечує доступ до команд;
* *панель інструментів:* розташовується під рядком меню, являє собою набір кнопок, забезпечує швидкий доступ до деяких команд;
* *кнопки Згорнути, Розгорнути/Відновити, Закрити* розташовані у верхній правій частині вікна.

Більш сучасна версія інтерфейсу замість горизонтального меню та панелі інструментів використовує елемент керування *Стрічка (Ribbon).*

До стандартних можливостей будь-якого додатку є:

* + Створення, відкриття та збереження вмісту робочої області.
  + Попередній перегляд та друк вмісту робочої області;
  + Робота з буфером обміну;

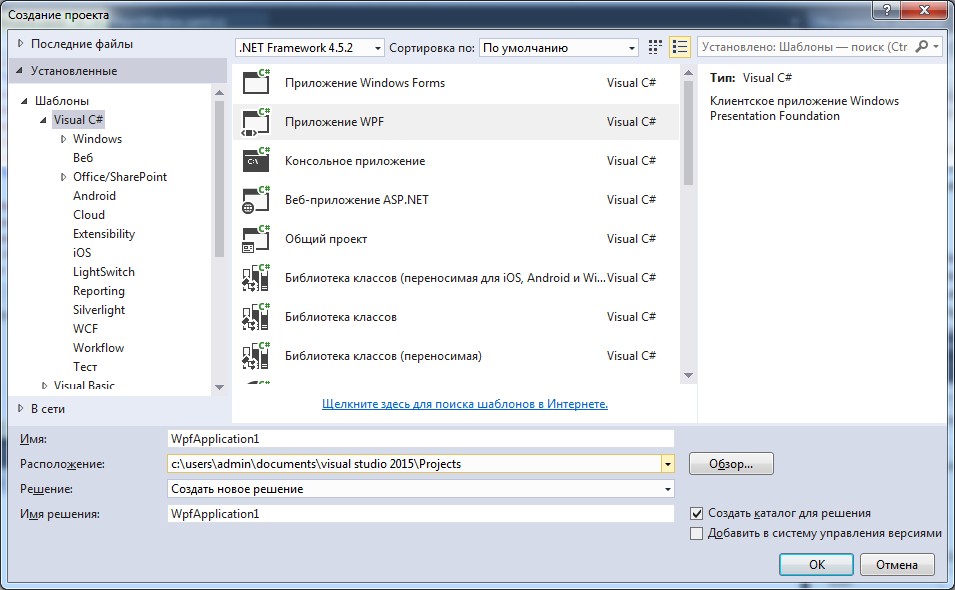
До функціональних можливостей текстових редакторів відносяться:

* + Керування виглядом текстових символів (керування шрифтами)
  + керування розташуванням тексту (форматування абзаців);
  + операції пошуку та заміни символів;
  + вставка зовнішніх об’єктів (зображень, таблиць, формул тощо)

Будь-який додаток має гнучку систему допомоги що складається із централізованої довідки, підказок та повідомлень в рядку стану.

Додатки поділяються на SPI (однодокументні) та MDI (багатодокументні). Класичний MDI реалізується багатовіконним додатком на основі системи Parent-Child Windows. В більш сучасних в сучасних інтерфейсах використовуються вкладки або технологія псевдоSDI.

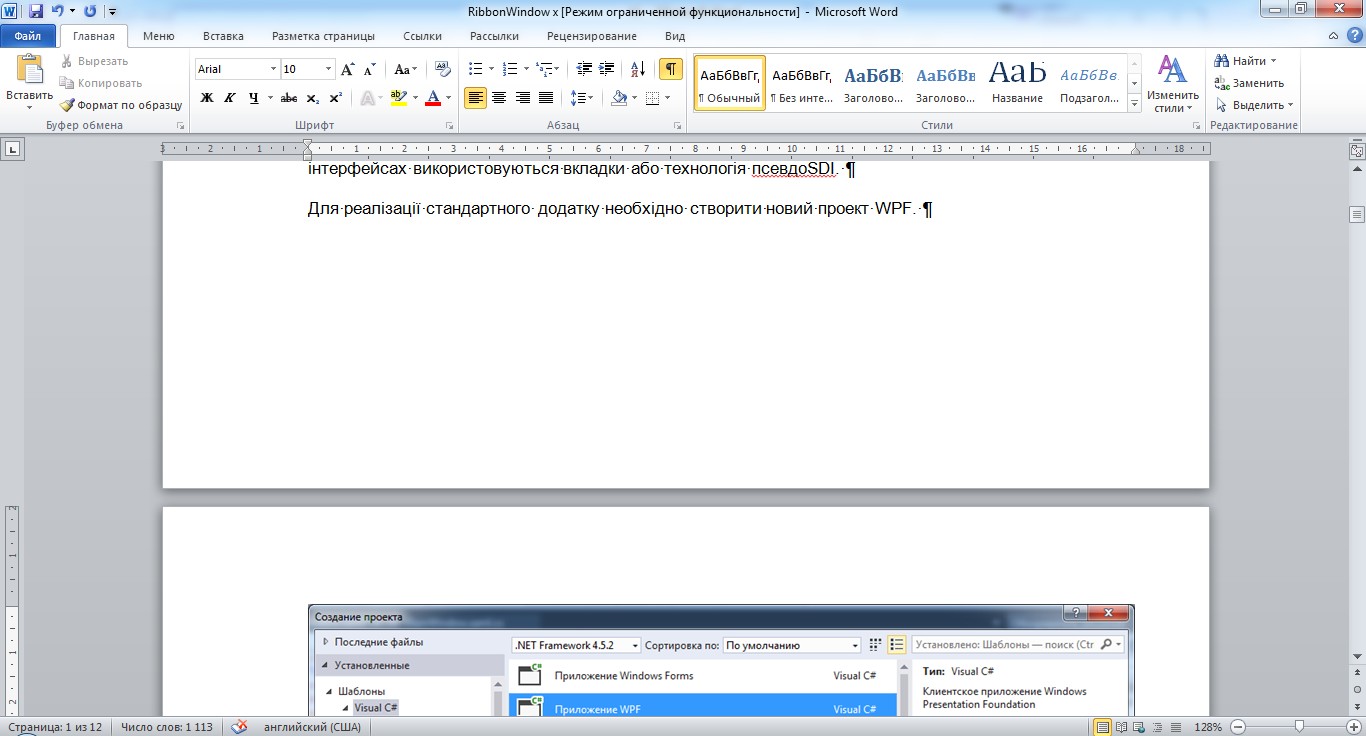
Windows Presentation Foundation (WPF [1]) - система для побудови клієнтських додатків для Windows з візуально привабливими можливостями взаємодії з користувачем, графічна (презентаційна) підсистема у складі .NET Framework (починаючи з версії 3.0), яка використовує мову XAML [2].

Для реалізації стандартного додатку необхідно створити новий проект WPF.

За зразок візьмемо інтерфейс текстового процесора Microsoft Word 2010.

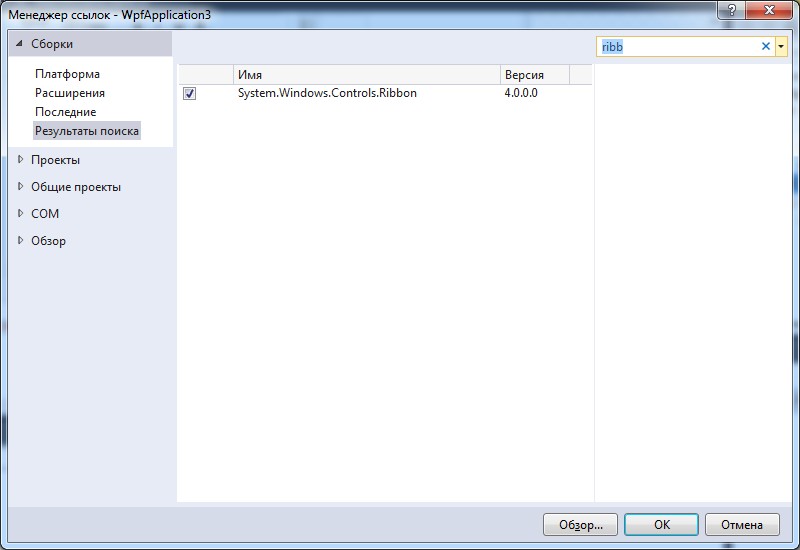
Послідовність створення SDI додатку в нашому випадку наступна:

* реалізації системи керування додатком (стрічки);
* реалізація робочої області;
* реалізація функціональних можливостей додатку.



# Реалізація стрічки (Ribbon).

Для реалізації системи керування в проект необхідно додати посилання на зборку System.Windows.Controls.Ribbon:

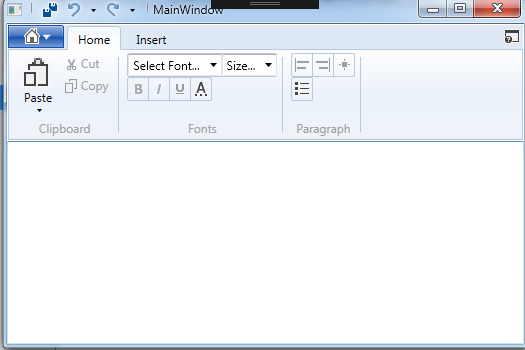


Додатково про System.Widows.Controls.Ribbon дивимось за посиланнями:

<https://www.codeproject.com/Articles/100081/Introducing-Ribbon-UI-Control-for-WPF> <https://thebeyond.ru/2011/03/10/microsoft-ribbon-for-wpf/>

<http://www.c-sharpcorner.com/UploadFile/0b73e1/ribbon-control-in-wpf-4-5/> [https://msdn.microsoft.com/en-](https://msdn.microsoft.com/en-us/library/system.windows.controls.ribbon.ribbon(v%3Dvs.110).aspx) [us/library/system.windows.controls.ribbon.ribbon(v=vs.110).aspx](https://msdn.microsoft.com/en-us/library/system.windows.controls.ribbon.ribbon(v%3Dvs.110).aspx) <http://www.drdobbs.com/windows/whats-new-in-wpf-45/240009614?pgno=1>

Дизайн інтерфейсу додатку буде виконано із використанням мови XAML. У результаті буде реалізований простий текстовий редактор, типу:



Для цього переходимо у файл з XAML розміткою головного вікна додатку (в даному випадку MainWindow.xaml) та першим кроком робимо заміну типу вікна з Windows на RibbonWindows:

*<RibbonWindow x:Class="WpfApplication3.MainWindow" xmlns="*[*http://schemas.microsoft.com/winfx/2006/xaml/presentation*](http://schemas.microsoft.com/winfx/2006/xaml/presentation)*" xmlns:x="*[*http://schemas.microsoft.com/winfx/2006/xaml*](http://schemas.microsoft.com/winfx/2006/xaml)*" xmlns:d="*[*http://schemas.microsoft.com/expression/blend/2008*](http://schemas.microsoft.com/expression/blend/2008)*" xmlns:mc="*[*http://schemas.openxmlformats.org/markup-compatibility/2006*](http://schemas.openxmlformats.org/markup-compatibility/2006)*" xmlns:local="clr-namespace:WpfApplication3"*

*mc:Ignorable="d"*

*Title="MainWindow" Height="350" Width="525">*

*<Grid>*

*</Grid>*

*</RibbonWindow>*

Наступним кроком в межах Grid визначаємо для розміточної сітки 2 рядки – перший для стрічки, другий для робочої області.

*<Grid.RowDefinitions>*

*<RowDefinition Height="Auto" />*

*<RowDefinition Height="\*" />*

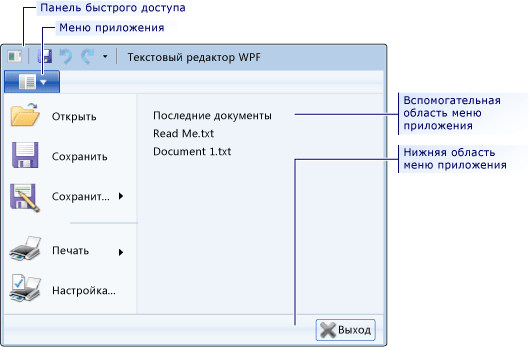
*</Grid.RowDefinitions>*

Далі розташовуємо стрічку в першому рядку:

*<Ribbon Grid.Row="0" Margin="1,1,1,1">*

*</Ribbon>*

Основними елементами стрічки є:



Далі реалізуємо інтерфейс *Довідникового меню*:

*<Ribbon.HelpPaneContent>*

*<RibbonButton SmallImageSource="images\HelpApplication.png" KeyTip="F"/>*

*</Ribbon.HelpPaneContent>*

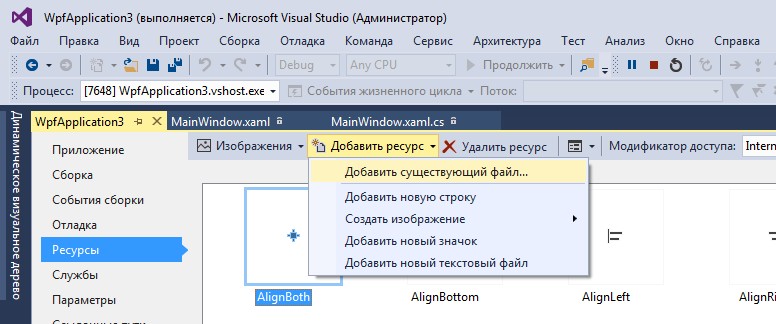
Для роботи із зображенням кнопки HelpPane та іншими кнопками необхідно виконати наступні підготовчи дії:

1. Створити в папці проекту папку images.
2. Зберегти в ній піктограми для наступних дій: Create, Open, Close, Save, Propetrty, Print Сopy, Paste, Cut, FontColor, Bold, Italic, Underline, AlignLeft, AlignRight, AlignBoth, AlignCenter, BulletList, InsertObject, InsertTable, InsertImage та для індивідуального завдання.

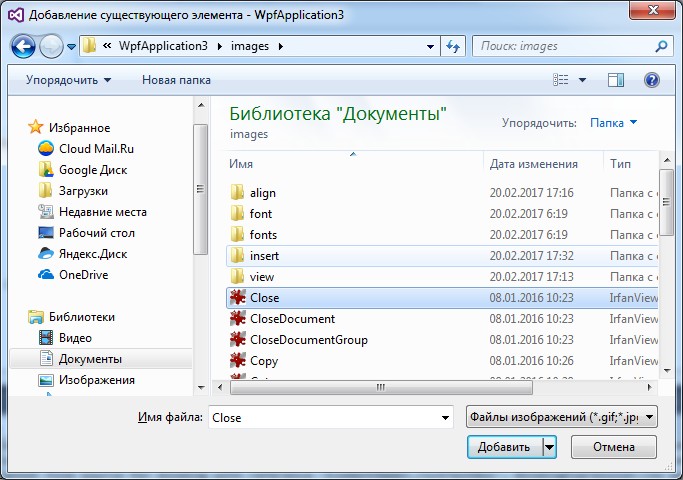
Для виконання даного завдання слід виконати пошук в Інтернеті колекції піктограм або скористатись VSImageLibrary за посиланням <https://www.microsoft.com/en-us/download/details.aspx?id=35825> або використати типовий набір кнопок із VSImageLibrary, що використані в даному зразку за посиланням

1. Додати проект ресурси зображень

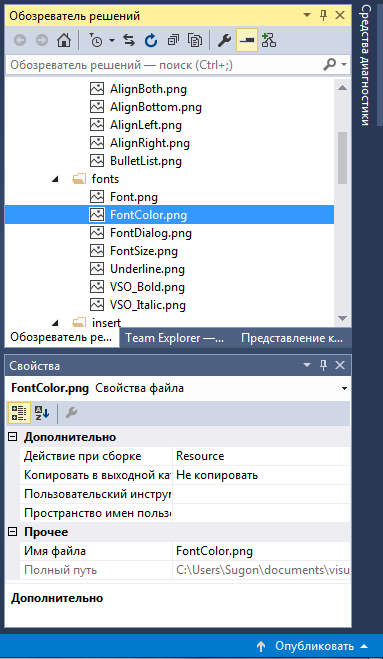
Для цього перейти в меню *Проект->Свойства*: на вкладку *Ресурси* та обрати *Изображения:*

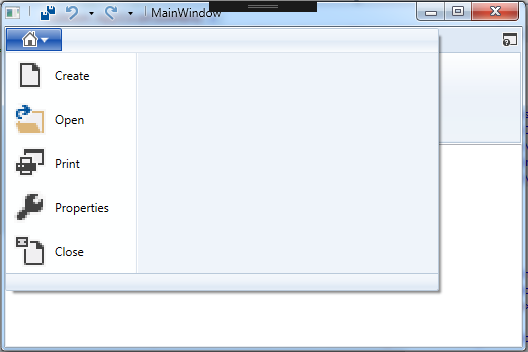


або за допомогою контекстного меню проекта у *Обозреватели решений* створити папку images и добавить зображення (команда *Добавить существующий елемент* в контекстному меню):



Обов’язково перевірити у властивостях всіх рисунків, що параметр *Действие при сборке* має значення Recource.



Наступним кроком є створення *Головного меню* додатку:

*<Ribbon.ApplicationMenu >*

*<RibbonApplicationMenu SmallImageSource="images\icon-home.png" >*

*<RibbonApplicationMenuItem Header="Create" ImageSource="images\Document.png" KeyTip="C" />*

*<RibbonApplicationMenuItem Header="Open" ImageSource="images\OpenFolder.png" KeyTip="O" />*

*<RibbonApplicationMenuItem Header="Print" ImageSource="images\PrintDialog.png" KeyTip="P"/>*

*<RibbonApplicationMenuItem Header="Properties" ImageSource="images\Property.png" />*

*<RibbonApplicationMenuItem Header="Close" ImageSource="images\CloseDocument.png" KeyTip="C" Click="btnClose\_Click"/>*

*</RibbonApplicationMenu>*

*</Ribbon.ApplicationMenu>*

Далі створимо *Меню швидкого доступу*:



*<Ribbon.QuickAccessToolBar>*

*<RibbonQuickAccessToolBar>*

*<RibbonButton SmallImageSource="images\SaveAll.png" />*

*<RibbonSplitButton x:Name="Undo" SmallImageSource="images\Undo.png" >*

*<RibbonSplitMenuItem Header="Undo1"></RibbonSplitMenuItem>*

*</RibbonSplitButton>*

*<RibbonSplitButton x:Name="Redo" SmallImageSource="images\Redo.png" >*

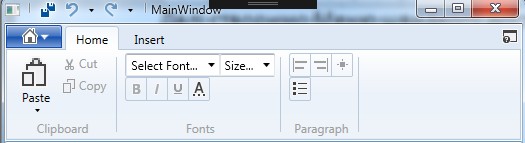
*<RibbonSplitMenuItem Header="Redo1"></RibbonSplitMenuItem>*

*</RibbonSplitButton>*

*</RibbonQuickAccessToolBar>*

*</Ribbon.QuickAccessToolBar>*

Наступним кроком є створення та заповнення вкладки *Home:*



Зверніть увагу, що дана вкладка поділена на три групи: Clipboard, Fonts, Paragraph.

*<RibbonTab Header="Home">*

*<!-- Home/Clipboard group-->*

*<RibbonGroup Header="Clipboard">*

*<RibbonMenuButton LargeImageSource="Images\paste.png" Label="Paste" KeyTip="V" >*

*<RibbonMenuItem ImageSource="Images\Paste.png" Header="Keep Text Only" KeyTip="T" />*

*<RibbonMenuItem ImageSource="Images\Paste.png" Header="Paste Special..." KeyTip="S"/>*

*</RibbonMenuButton>*

*<RibbonButton SmallImageSource="Images\Cut.png" Label="Cut" KeyTip="X" />*

*<RibbonButton SmallImageSource="Images\Copy.png" Label="Copy" KeyTip="C" />*

*</RibbonGroup>*

*<!-- Home/Colors group-->*

*<RibbonGroup x:Name="fonts" Header="Fonts" Width="Auto" >*

*<RibbonControlGroup>*

*<ComboBox ItemsSource="{Binding Source={x:Static Fonts.SystemFontFamilies}}" Text="Select Font..." IsEditable="True"/>*

*<ComboBox x:Name="\_fontSize" Text="Size..." IsEditable="True"></ComboBox>*

*</RibbonControlGroup>*

*<RibbonControlGroup>*

*<RibbonButton SmallImageSource="Images\fonts\VSO\_Bold.png" KeyTip="B" />*

*<RibbonButton SmallImageSource="Images\fonts\VSO\_Italic.png" KeyTip="I" />*

*<RibbonButton SmallImageSource="Images\fonts\Underline.png" KeyTip="U" />*

*<RibbonButton SmallImageSource="Images\fonts\FontColor.png" KeyTip="A"/>*

*</RibbonControlGroup>*

*</RibbonGroup>*

*<RibbonGroup x:Name="paragraph" Header="Paragraph">*

*<RibbonControlGroup>*

*<RibbonButton SmallImageSource="Images\align\AlignLeft.png" />*

*<RibbonButton SmallImageSource="Images\align\AlignRight.png" />*

*<RibbonButton SmallImageSource="Images\align\AlignBoth.png" />*

*</RibbonControlGroup>*

*<RibbonControlGroup>*

*<RibbonButton SmallImageSource="Images\align\BulletList.png"/>*

*</RibbonControlGroup>*

*</RibbonGroup>*

*</RibbonTab>*

Для коректного відображення розмірів шрифту у відповідному ComboBox необхідно у файлі коду головного вікна (в даному випадку MainWindow.xaml.cs) визначити структуру даних, приблизно такого вмісту:

*public double[] FontSizes*

*{*

*get*

*{*

*return new double[] { 3.0, 4.0, 5.0, 6.0, 6.5, 7.0, 7.5, 8.0, 8.5, 9.0,*

*9.5, 10.0, 10.5, 11.0, 11.5, 12.0, 12.5,13.0,13.5,14.0, 15.0,16.0, 17.0, 18.0, 19.0,*

*20.0, 22.0, 24.0, 26.0, 28.0, 30.0,32.0, 34.0, 36.0, 38.0, 40.0, 44.0, 48.0, 52.0, 56.0,*

*60.0, 64.0, 68.0, 72.0, 76.0,80.0, 88.0, 96.0, 104.0, 112.0, 120.0, 128.0, 136.0, 144.0*

};

}

}

та визначити її як джерело даних для елемента керування ComboBox

*public MainWindow()*

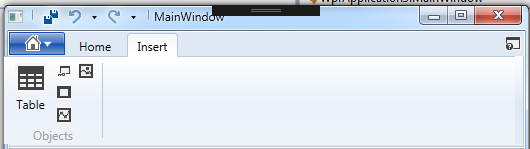
*{*

*InitializeComponent();*

*\_fontSize.ItemsSource = FontSizes;*

}

Останнім кроком є створення вкладки Insert:



*<RibbonTab Header="Insert" Margin="-8,0,8,0" UseLayoutRounding="True" ScrollViewer.VerticalScrollBarVisibility="Auto">*

*<RibbonGroup x:Name="objects" Header="Objects">*

*<RibbonButton LargeImageSource="Images\insert\Table.png" Label="Table"/>*

*<RibbonButton SmallImageSource="Images\insert\ApplicationAccess.png"/>*

*<RibbonButton SmallImageSource="Images\insert\Rectangle.png"/>*

*<RibbonButton SmallImageSource="Images\insert\LineChart.png"/>*

*<RibbonButton SmallImageSource="Images\insert\Image.png"/>*

*</RibbonGroup>*

*</RibbonTab>*

Компілюємо проект та перевіряємо помилки (сподіваємось, що це вже 3 або 4 раз). Дивимось на результат.

# Реалізація робочої області

Для реалізації робочої області використаємо компонет RichTextBox.

*<RichTextBox x:Name="doc1" Grid.Row="1">*

*</RichTextBox>*

Даний компонент містить вбудований функціонал для підтримки стандартних операцій з текстом, таких як робота з буфером обміну та параметрами шрифту та абзацу.

Для використання вбудованих властивостей компоненту використаємо розширення розмітки x:Static (див. [https://msdn.microsoft.com/ru-](https://msdn.microsoft.com/ru-ru/library/ms742135(v%3Dvs.110).aspx)) [ru/library/ms742135(v=vs.110).aspx)](https://msdn.microsoft.com/ru-ru/library/ms742135(v%3Dvs.110).aspx)). За його допомоги реалізуємо наступні команди:

* команди рівня додатку: Cut, Copy, Paste, Undo, Rendo.
* команди редагування: Bold, Italic, Underline, AlignLeft, AlignRight, AlignJustify.

Додаємо відповідні дефініції в код розмітки вікна:

в *Меню швидкого доступу*:

*<RibbonSplitButton x:Name="Undo" SmallImageSource="images\Undo.png" Command="{x:Static ApplicationCommands.Undo}" CommandTarget="{Binding ElementName=\_richTextBox}">*

*…*

*<RibbonSplitButton x:Name="Redo" SmallImageSource="images\Redo.png" Command="{x:Static ApplicationCommands.Redo}" CommandTarget="{Binding ElementName=\_richTextBox}">*

*на вкладку Insert*

*…*

*<RibbonMenuItem ImageSource="Images\Paste.png" Header="Keep Text Only" KeyTip="T"*

*Command="{x:Static ApplicationCommands.Paste}" CommandTarget="{Binding ElementName=\_richTextBox}"/>*

*…*

*<RibbonButton SmallImageSource="Images\Cut.png" Label="Cut" KeyTip="X" Command="{x:Static ApplicationCommands.Cut}" CommandTarget="{Binding ElementName=\_richTextBox}"/>*

*<RibbonButton SmallImageSource="Images\Copy.png" Label="Copy" KeyTip="C" Command="{x:Static ApplicationCommands.Copy}" CommandTarget="{Binding ElementName=\_richTextBox}"/>*

*…*

*<RibbonButton SmallImageSource="Images\fonts\VSO\_Bold.png" KeyTip="B" Command="{x:Static EditingCommands.ToggleBold}" CommandTarget="{Binding ElementName=\_richTextBox}"/>*

*<RibbonButton SmallImageSource="Images\fonts\VSO\_Italic.png" KeyTip="I" Command="{x:Static EditingCommands.ToggleItalic}" CommandTarget="{Binding ElementName=\_richTextBox}"/>*

*<RibbonButton SmallImageSource="Images\fonts\Underline.png" KeyTip="U" Command="{x:Static EditingCommands.ToggleUnderline}" CommandTarget="{Binding ElementName=\_richTextBox}"/>*

*…*

*<RibbonButton SmallImageSource="Images\align\AlignLeft.png" Command="{x:Static EditingCommands.AlignLeft}" CommandTarget="{Binding ElementName=\_richTextBox}"/>*

*<RibbonButton SmallImageSource="Images\align\AlignRight.png" Command="{x:Static EditingCommands.AlignRight}" CommandTarget="{Binding ElementName=\_richTextBox}"/>*

*<RibbonButton SmallImageSource="Images\align\AlignBoth.png" Command="{x:Static EditingCommands.AlignJustify}" CommandTarget="{Binding ElementName=\_richTextBox}"/>*

# Реалізація основних функцій додатку

Для реалізації функції відкриття файлів визначимо та реалізуємо процедуру btnOpen\_Click:

*<RibbonApplicationMenuItem Header="Open" ImageSource="images\OpenFolder.png" KeyTip="O" Click="btnOpen\_Click"/>*

реалізація має наступний вид:

*private void btnOpen\_Click(object sender, RoutedEventArgs e)*

*{*

*OpenFileDialog dlg = new OpenFileDialog(); dlg.Filter = "Document files (\*.rtf)|\*.rtf"; var result = dlg.ShowDialog();*

*if (result.Value)*

*{*

*TextRange t = new TextRange(doc1.Document.ContentStart, doc1.Document.ContentEnd); FileStream file = new FileStream(dlg.FileName, FileMode.Open);*

*t.Load(file, System.Windows.DataFormats.Rtf);*

*}*

*}*

Для реалізації функції збереження файлів визначимо та реалізуємо процедуру btnSave\_Click:

<RibbonButton SmallImageSource="images\SaveAll.png" Click="btnSave\_Click"/>

реалізація має наступний вид:

*private void btnSave\_Click(object sender, RoutedEventArgs e)*

*{*

*SaveFileDialog savefile = new SaveFileDialog();*

*// set a default file name savefile.FileName = "unknown.doc";*

*// set filters - this can be done in properties as well savefile.Filter = "Document files (\*.rtf)|\*.rtf";*

*if (savefile.ShowDialog() == true)*

*{*

*TextRange t = new TextRange(doc1.Document.ContentStart, doc1.Document.ContentEnd); this.Title = this.Title + " " + savefile.FileName;*

*FileStream file = new FileStream(savefile.FileName, FileMode.Create); t.Save(file, System.Windows.DataFormats.Rtf);*

*file.Close();*

*}*

*IsSaved = true;*

*}*

Змінна IsSaved визначається як глобальна і в подальшому вкористовується для контролю збереження файлу перед закриттям додатку.

public partial class MainWindow : RibbonWindow

{

public bool IsSaved = false;

визначення процедури закриття додатку:

*<RibbonApplicationMenuItem Header="Close" ImageSource="images\CloseDocument.png" KeyTip="C" Click="btnClose\_Click"/>*

реалізація:

*private void btnClose\_Click(object sender, RoutedEventArgs e)*

*{ if (IsSaved == false)*

*if (MessageBox.Show("Do you want save changes ?", "Message", MessageBoxButton.YesNo) == MessageBoxResult.Yes)*

*//Если была нажата кнопка Yes, вызываем метод Save*

*{*

*this.btnSave\_Click(sender, e);*

*}*

*this.Close();*

*}*

Останнім кроком є реалізація процедури визначення типу та розміру шрифта. Для цього визначимо наступні процедури:

*<ComboBox ItemsSource="{Binding Source={x:Static Fonts.SystemFontFamilies}}" SelectionChanged="FontFamili\_SelectionChange" Text="Select Font..." IsEditable="True"/>*

*<ComboBox SelectionChanged="FontSize\_SelectionChange" x:Name="\_fontSize" Text="Size..." IsEditable="True"></ComboBox>*

та реалізуємо їх наступним чином:

*void ApplyPropertyValueToSelectedText(DependencyProperty formattingProperty, object value)*

*{*

*if (value == null) return;*

*doc1.Selection.ApplyPropertyValue(formattingProperty, value);*

*}*

*private void FontFamili\_SelectionChange(object sender, SelectionChangedEventArgs e)*

*{*

*try*

*{*

*FontFamily editValue = (FontFamily)e.AddedItems[0]; ApplyPropertyValueToSelectedText(TextElement.FontFamilyProperty, editValue);*

*}*

*catch (Exception) { }*

*}*

*private void FontSize\_SelectionChange(object sender, SelectionChangedEventArgs e)*

*{*

*try*

*{*

*ApplyPropertyValueToSelectedText(TextElement.FontSizeProperty, e.AddedItems[0]);*

*}*

*catch (Exception) { }*

*}*

*}*

*}*

Компілюємо, дивимось на результат та переходимо до виконання індивідуального завдання.

**Контрольні питання**

1. **Людино-машинна(комп'ютерна) взаємодія**
2. вивчає методи розробки, оцінки та впровадження інтерактивних комп'ютерних систем, призначених для використання людиною;
3. вивчає методи прийняття рішень при цьому переважно розглядаються формалізовані задачі;
4. займається дослідженням процесів збереження, накопичення, перетворення, передачі даних та інформації із застосуванням комп'ютерної техніки;
5. вивчає системи зі зворотнім зв'язком і аспект керування інформацією в цих системах, розглядаючи при цьому строго формалізовані задачі.
6. **Задачі** людино-комп'ютерної взаємодії:
7. методологією і розвитком проектування інтерфейсів (тобто, виходячи з вимог і класу користувачів, проектування найкращого інтерфейсу в заданих рамках, оптимізація під необхідні властивості, такі як здатність до навчання і ефективність використання);
8. методами реалізації інтерфейсів (наприклад, програмні інструментарії, бібліотеки та раціональні алгоритми);
9. методами для оцінки та порівняння таких інтерфейсів;
10. розробкою нових інтерфейсів і технологій взаємодії;
11. розвитком описових і прогнозованих моделей, і теорією взаємодії.
12. **Вузол взаємодії включає в себе кілька аспектів:**
13. Область завдань;
14. Область машини;
15. Області інтерфейсу;
16. Вхідний потік;
17. Вихідний потік;
18. Зворотній зв'язок;
19. Геометрічні параметри;
20. Фонове оформлення;
21. Тематичний напрямок;
22. Довідникова система.
23. **Область завдань:**
24. умови і цілі, орієнтовані на користувача;
25. середовизе з яким взаємодіє комп'ютер;
26. непересічні області, що стосуються процесів людини і комп'ютера, не відносяться до сфери взаємодії;
27. потік інформації, який починається в області завдань, коли користувач має кілька завдань, які вимагають використання комп'ютера;
28. потік інформації, який виникає в машині;
29. вузли взаємодії, що проходять через інтерфейс, оцінюються, модеруються та підтверджуються.
30. **Типи діалогів. поставте відповідність**
31. Структура діалогу типу «питання-відповідь» (Q & A);
32. заснована на аналогії зі звичайним інтерв'ю. Система бере на себе роль інтерв'юера і отримує інформацію від користувача у вигляді відповідей на питання;
33. Діалог на основі меню;
34. є найбільш природним механізмом для роботи з пристроями вказівки і вибору: складається із зображень тих об'єктів, які вибираються користувачем;
35. Діалог на основі екранних форм;
36. допускає обробку на одному кроці кількох відповідей. Використовуються в основному там, де облік будь-якої діяльності вимагає введення досить стандартного набору даних;
37. діалог на основі командного мови;
38. програмна система не виводить нічого, крім постійної підказки (запрошення на введення команди), яка означає готовність системи до роботи. Кожну команду вводять з нового рядка і зазвичай закінчують натисканням клавіші «введення».

# Лабораторна робота № 2

**Тема роботи:** Класифікація інтерфейсу.

**Мета роботи:** аналіз ефективності інтерфейсів із застосуванням закону Фітса.

Обладнання: ПК, Visual Studio не нижче v.2010 (.Net Framework 4.5)

## 1. Завдання на лабораторну роботу

1. створення алгоритму і програми для дослідження особливостей застосування закону Фітса в дизайні інтерфейсів;
2. проведення експериментів по дослідженню швидкості і точності фізичних дій користувача;
3. побудова і інтерполяція графіків залежності тривалості фізичних дій користувача і кількості помилок від параметрів інтерфейсу засобами Microsoft Excel.

# Індивідуальні завдання

У всіх варіантах необхідно в середовищі візуального програмування розробити програми для рішення приведених в них задач.

Пропонуються наступні групи задач в різних варіантах:

1. Визначення залежності часу досягнення об’єкту від його розміру і дистанції до нього.

2. Визначення залежності числа помилок, пов’язаних з промахами при досягненні (за обмежений час) дрібних об’єктів від розміру об’єкта і дистанції до нього.

3. Визначення впливу різних сполучень кольорів фону і об’єкта (розмір, що сприймається суб’єктивно) на швидкість його досягнення.

4. Порівняльний аналіз нескінченних і звичайних кнопок по швидкості їх досягнення.

Перша група задач має на увазі три серії експериментів:

1. При фіксованому розмірі D, заданому в табл. 1, визначити вплив дистанції S на час досягнення об’єкта. Для дистанцій S = 0, 20, 40, 60, 100, 150, 200, 250, 300, 350 (в пікселях) необхідно провести по 5 експериментів (натискань) і визначити середній час досягнення об’єкта при кожній дистанції.
2. При фіксованій дистанції S (див. табл. 1) визначити вплив розміру D на час досягнення об’єкта. Для розмірів D = 8, 10, 12, 15, 20, 30, 50, 70, 100 (в пікселях) необхідно провести по 5 експериментів (натискань) і визначити середній час досягнень об’єкта при кожному розмірі.
3. При змінюючому розмірі D і дистанції S (2 вкладених цикла), які задані в попередніх серіях експериментів, визначити для кожного відношення S/D середній час досягнення об’єкта по 3 натисканням.

*Таблиця 1*

*Варіанти завдань для визначення залежності часу досягнення об’єкта від його розміру і дистанції до нього*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| №  вар. | Об’єкт | Розмір (D), пікс. | Дистанція  (S), пікс. | Колір  об’єкта | Колір  фону |
|  | Прямокутник  D x 2D | 7 | 300 | чорний | білий |
|  | Прямокутник  D x 2,5D | 8 | 350 | сірий | білий |
|  | Прямокутник  D x 3D | 9 | 400 | синій | білий |
|  | Кнопка  D x 2D | 10 | 500 | сірий | синій |
|  | Кнопка  D x 3D | 12 | 550 | сірий | чорний |
|  | Кнопка  D x 4D | 15 | 600 | сірий | синій |
|  | Квадрат | 8 | 200 | чорний | синій |
|  | Квадрат | 10 | 250 | сірий | синій |
|  | Квадрат | 12 | 300 | жовтий | синій |
|  | Квадрат | 15 | 350 | сірий | жовтий |
|  | Квадрат | 17 | 400 | чорний | жовтий |
|  | Круг | 8 | 300 | червоний | синій |
|  | Круг | 10 | 400 | червоний | білий |
|  | Круг | 12 | 500 | синій | білий |
|  | Круг | 15 | 600 | сірий | чорний |
|  | Круг | 17 | 650 | жовтий | синій |
|  | Еліпс D x 2D | 9 | 500 | жовтий | сірий |
|  | Еліпс D x 2,5D | 11 | 400 | червоний | сірий |
|  | Еліпс D x 3D | 12 | 300 | синій | сірий |
|  | Еліпс D x 3,5D | 14 | 200 | білий | зелений |
|  | Еліпс D x 4D | 16 | 150 | сірий | синій |

На основі отриманих середніх значень часу досягнення об'єктів в MS Excel повинні бути побудовані 3 графіка: т (S), т (D), т (S / D). До кожного графіку необхідно побудувати апроксимуючу криву - логарифмічну лінію тренда - і вивести рівняння графіка функції.

У другій групі завдань необхідно провести 2 досліди, час на виконання яких обмежений:

1. При D фіксованому розмірі, заданому в табл. 2, визначити вплив дистанції S на число помилок досягнення об'єкта. Для дистанцій S = 0, 20, 40, 60, 100, 150, 200, 250, 300, 350 (в пікселях) необхідно провести по 8 експериментів (успішних натискань) і визначити сумарне число помилкових натискань досягнення об'єкта для кожної дистанції.

2. При S фіксованою дистанції (див. табл. 2) визначити вплив розміру D на число помилок досягнення об'єкта. Для розмірів D = 3, 4, 5, 6, 7, 8, 10, 12, 14 (в пікселях) необхідно провести по 8 експериментів (успішних натискань) і визначити сумарне число помилкових натискань досягнення об'єкта для кожного розміру.

*Таблиця 2*

*Варіанти завдань для визначення залежності числа помилок при досягненні об’єктів від їх розмірів і дистанції до них*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| №  вар. | Об’єкт | Час,  (t), мс | Розмір (D), пікс. | Дистанція  (S), пікс. | Колір  об’єкта | Колір  фону |
|  | Прямокутник  D x 2D | 1200 | 4 | 100 | чорний | синій |
|  | Прямокутник  D x 3D | 1500 | 3 | 200 | жовтий | синій |
|  | Кнопка  D x 2D | 1800 | 3 | 250 | сірий | зелений |
|  | Кнопка  D x 3D | 1300 | 4 | 300 | сірий | жовтий |
|  | Квадрат | 1400 | 4 | 400 | синій | білий |
|  | Квадрат | 1500 | 5 | 550 | жовтий | сірий |
|  | Квадрат | 1800 | 3 | 600 | червоний | сірий |
|  | Круг | 1000 | 5 | 100 | синій | сірий |
|  | Круг | 1500 | 4 | 150 | білий | зелений |
|  | Круг | 1600 | 4 | 250 | чорний | білий |
|  | Еліпс D x 2D | 1200 | 5 | 300 | сірий | білий |
|  | Еліпс D x 2,5D | 1500 | 4 | 350 | синій | білий |
|  | Еліпс D x 3D | 1800 | 3 | 400 | чорний | сірий |

На основі отриманих значень числа помилок при досягнені об'єкта в MS Excel повинні бути побудовані 2 графіка: N (S), N (D), де N - число помилок. Для кожного графіка необхідно побудувати апроксимуючу криву - логарифмічну лінію тренда.

У третій групі завдань необхідно провести 2 досліди:

1. При D фіксованому розмірі, S дистанції і кольорі об'єкта, заданому в табл. 3, визначити вплив кольору фону на час досягнення об'єкта. Для кожного з 15 кольорів стандартної 16-кольоровий палітри (виключаючи колір об'єкта) необхідно провести по 10 експериментів (натискань) і визначити середній час досягнення об'єкта при кожному кольорі фону.

2. При D фіксованому розмірі, S дистанції і кольорі фону (див. табл. 3) визначити вплив кольору об'єкта на час його досягнення. Для кожного з 15 кольорів стандартної 16-колірної гами (виключаючи колір фону) необхідно провести по 10 (експериментів) натискань і визначити середній час досягнення об'єкта при кожному кольорі фону.

*Таблиця 3*

*Варіанти завдань для визначення залежності швидкості роботи від кольору фону і об’єкта*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| №  вар. | Об’єкт | Час,  (t), мс | Розмір (D), пікс. | Колір  об’єкта | Колір  фону |
|  | Прямокутник  D x 2D | 4 | 150 | сірий | синій |
|  | Прямокутник  D x 3D | 3 | 200 | жовтий | синій |
|  | Квадрат | 5 | 450 | сірий | чорний |
|  | Квадрат | 4 | 500 | жовтий | синій |
|  | Квадрат | 3 | 600 | червоний | сірий |
|  | Круг | 5 | 100 | синій | сірий |
|  | Круг | 4 | 250 | чорний | білий |
|  | Еліпс D x 2,5D | 4 | 350 | зелений | білий |
|  | Еліпс D x 3D | 3 | 400 | чорний | сірий |
|  | Еліпс D x 4D | 4 | 500 | сірий | сірий |

##### На основі отриманих значень в MS Excel повинні бути побудовані 2 стовпчастих: t(CФ), t(Cо), де СФ - колір фону, Cо - колір об'єкта.

##### Всім дослідам і варіантам відповідають наступні вимоги і рекомендації:

##### 1. Типовий алгоритм роботи програми повинен бути наступним: а) вибрати номер досвіду з меню; б) намалювати об'єкт встановити курсор в задані позиції; в) по об'єкту клацнути, відраховуючи час від моменту початку руху; г) зберегти проміжні результати; д) перемалювати об'єкт в новому місці вікна, встановити курсор в початкову позицію і т.д.

## 2. Методи підвищення доступності кнопки

Із закону Фітса можна зробити висновок, що кращий спосіб підвищити доступність кнопки полягає у тому, щоб робити її великою і розташовувати ближче до курсора.

У цього правила є два слідства. Щоб «нескінченно» прискорити натискання кнопки, її, по-перше, можна зробити нескінченного розміру і, по-друге, дистанцію до неї можна зробити нульовою.

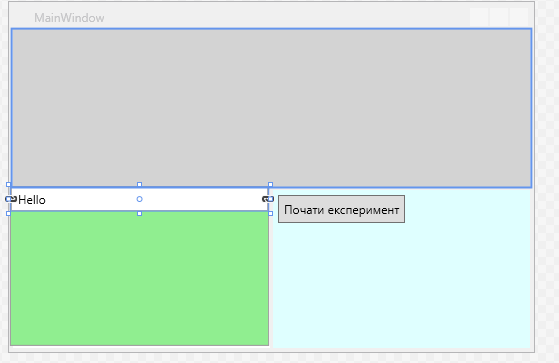
Варіант 1. Кнопка нескінченного розміру. При підведенні курсору до краю екрану він зупиняється, навіть якщо рух миші триває. Це означає, що кнопка, росташована впритул до верхнього або нижнього краю екрану, має нескінченну висоту (так само як кнопка біля лівого або правого краю має нескінченну ширину). Таким чином, швидкість досягнення такої кнопки залежить тільки від відстані до неї і точності вибору початкового напрямку руху.

Отже, кнопка, розташована в кутку екрану (наприклад, кнопка виклику головного меню Windows, «Пуск»), має ще більшу перевагу. Для досягнення такої кнопки від користувача потрібно всього лише перемістити мишу в потрібному напрямку, не піклуючись про її швидкості і не роблячи спроб зупинити її в потрібному місці.

Варіант 2. Нульова дистанція до кнопки. Контекстне меню, яке викликається після натискання правої кнопки миші, завжди відкривається під курсором, відповідно відстань до будь-якого його елемента завжди мінімально. Тому контекстне меню є одним з найшвидших і ефективних елементів управління.

Зменшувати відстані до цілі можна і в діалогових вікнах, які також є контекстно-залежними. За замовчуванням вони відкриваються в центрі екрану, але виходячи з принципу скорочення відстані до їх кнопок, відкривати їх під курсором набагато ефективніше (якщо вони не будуть перекривати важливу інформацію на екрані).

Приклад інтерфейсу для завдання 1 (WPF)



## Хід роботи

XAML-розмітка компонентів



Реалізація основних функцій

Перевірити підключені ресурси:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

using System.Timers;

Змінні, які потрібні:



Початок експерименту – натискання кнопки:



Промальовування фігур - тут маленькі червоні квадратики:



Проведення 1 вимірювання - для постійних розмірів і згенерованих координат:



Оброблювач подій миші - вимір часу + перехід до наступного виміру:



Контрольні питання

1. Інтерфейс складається:
2. набір завдань користувача, які він вирішує за допомогою системи;
3. використовувана системою метафора (наприклад, робочий стіл в MS Windows тощо);
4. елементи управління системою;
5. навігація між блоками системи;
6. візуальний (і не тільки) дизайн екранів програми.
7. Види інтерфейсів:
8. командний інтерфейс;
9. графічний користувальницький інтерфейс;
10. SILK-інтерфейс;
11. семантичний інтерфейс;
12. об'єктно-орієнтований інтерфейс;
13. з віддаленим керуванням.
14. SILK-інтерфейс
15. користувач дає команди комп'ютеру, який їх виконує і видає результат користувачеві;
16. даний підхід полягає в символічному зображенні доступних дій у вигляді ікон (icons) на екрані і надання користувачеві можливості вибирати дії за допомогою миші або іншого координатного пристрою введення;
17. Інтерфейс найбільш наближений до звичайної, людської формі спілкування. При цьому комп'ютер визначає команди, аналізуючи людську мову і знаходячи в ній ключові фрази. Результат виконання команд комп'ютер перетворює в зрозумілу людині форму.
18. Графічний користувальницький інтерфейс
19. користувач дає команди комп'ютеру, який їх виконує і видає результат користувачеві
20. даний підхід полягає в символічному зображенні доступних дій у вигляді ікон (icons) на екрані і надання користувачеві можливості вибирати дії за допомогою миші або іншого координатного пристрою введення \*
21. Інтерфейс найбільш наближений до звичайної, людської формі спілкування. При цьому комп'ютер визначає команди, аналізуючи людську мову і знаходячи в ній ключові фрази. Результат виконання команд комп'ютер перетворює в зрозумілу людині форму

# Лабораторна робота № 3

**Тема роботи:** Основи програмування під ОС Android.

**Мета роботи:** Розробка додатків з використанням Android IDE.

**Обладнання:** ПК, Android Studio.

## 1. Завдання на лабораторну роботу

## Завдання 1. Основні етапи розробки додатків з використанням Android IDE.

## Завдання 2. Проектування інтерфейсу додатку

## Вимоги до звітності:

## 1. Скріни додатків реалізованих у завдання 1,2

## 2. Лістинг робочого коду додатку

## 3. Опис порядку встановлення та

## тестування на мобільному пристрої.

## Ресурси:

## http://developer.alexanderklimov.ru/android/ [*http://www.intuit.ru/studies/courses/12643/1191/info*](http://www.intuit.ru/studies/courses/12643/1191/info)

## Відеоуроки

## *https://www.youtube.com/watch? v=mmUgcokRJ5k&index=1&list=PLIU76b8Cjem7oSU18zT KNfsyzrmf3Zubl*

## Індивідуальні завдання

1. Каталог електронних книг

2. Журнал читача

3. Книга контактів

4. Каталог відео

5. Каталог фотографій

6. Дані про замовлення від магазинів

7. Дані про замовлення в кафе(ресторані)

8. Бронювання номерів

9. Дані про об’єкти нерухомості

10. Перелік справ на день

11. To-do List для проекту

12. Щоденник

13. Історичні пам’ятники

14 Перелік покупок

15. Реєстрація пацієнтів

16. План тренувань

17. Дані техоглядів

18. Реєстрація дітей в дитячі садочки

19. Інвентарний опис техніки

20. Товари на складі

21. Перелік публікацій

22. Участь у проектах

23. Нотатки

24. Реєстр виборців

25. Реєстр співробітників 26. Паркування автомобілів

27. Ремонт техніки

28. Реєстрація клієнтів

29. Реєстрація учасників форуму

30. Профілі користувачів

## Хід роботи

**Основні етапи розробки програми з використанням ANDROID IDE.**

**Крок 1. Введення.**

Для досягнення поставленої мети в лабораторній роботі створимо додаток в середовищі розробки Android IDE (Eclipse і ADT), докладно розглянемо структуру отриманого проекту і розберемо призначення основних його елементів.

Щоб подальші дії придбали певний сенс, сформулюємо завдання, яке буде вирішувати наш додаток, назвемо його "Вгадай число". Суть програми в тому, що програма випадковим чином "загадує" число від 0 до 100, а користувач повинен вгадати це число. При кожному введенні числа, програма повідомляє користувачеві результат: введене число більше задуманого, менше або ж "Ура, перемога!" число угадано.

Розроблюється, виконує свої функції тільки коли мабуть на екрані, коли воно не видимо його робота припиняється, т. Е. Маємо справу з додатком переднього плану. Для виконання всієї роботи досить визначити одну активність в додатку, фонові процеси не передбачені.

Далі в роботі розглянемо найпростіші елементи інтерфейсу користувача і додамо їх в додаток, а також розглянемо питання, пов'язані безпосередньо з програмуванням: навчимося обробляти події, що виникають при взаємодії додатки з користувачем; реалізуємо логіку перевірки числа на збіг з задуманим.

**Крок 2. Створення програми та вивчення його структури**

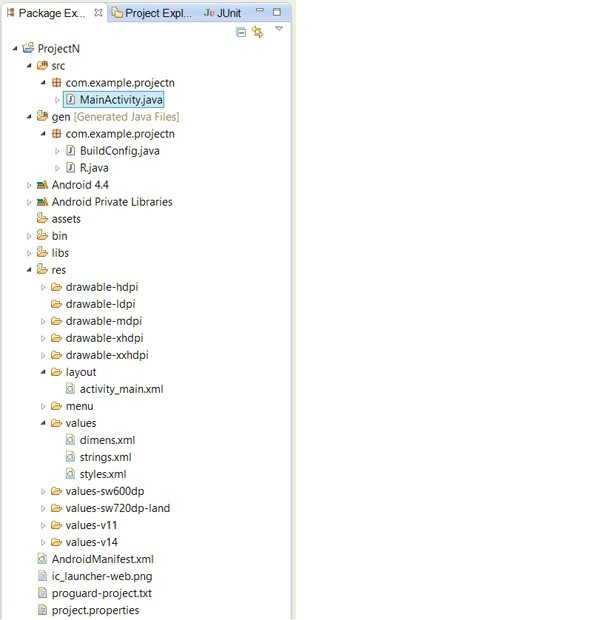
Створіть новий проект в середовищі Android IDE (Eclipse з ADT). Процес створення нового проекту і опис основних налаштувань детально розглянуто в лабораторній роботі до першої лекції.

У процесі створення проекту, ми назвали його ProjectN, середовище розробки готує необхідні папки і файли. Повний ієрархічний список обов'язкових елементів проекту можна побачити на вкладці Package Explorer (аналогічну інформацію надає вкладка Project Explorer), ієрархія отриманих папок і файлів для нашого проекту зображена на рисунку.

В даний час нас буде цікавити призначення декількох файлів і папок. Розглянемо папки:

- папка src - містить файли з вихідним кодом на мові Java. Саме в цій папці розміщуються всі класи, створювані в процесі розробки програми. Зараз в цій папці в пакеті com.example.projectn розміщується єдиний классFullscreenActivity.java. Цей клас визначає головну і єдину активність в цьому додатку.

Коментар 1: Ім'я пакету присвоюється в процесі створення програми в поле Package Name, використовувати com.example не рекомендується, так як. Пакет з таким ім'ям не можна завантажити в Google Play. Часто рекомендують використовувати в якості імені пакета назву сайту програміста, записане в зворотному порядку, можна просто використовувати свої ім'я та прізвище. Останнє слово в імені пакету формується автоматично і збігається з ім'ям проекту.



Коментар 2: Ім'я файлу присвоюється в процесі створення програми на етапі налаштування активності. Ім'я визначається в поле Activity Name.

Коментар 3: Package Explorer відображає структуру папок, яка створюється в каталозі, обраному в якості робочого (Workspace) при запуску Eclipse. Наприклад, робочий каталог називається workspaceADT, в ньому для нашого проекту з'явилася папкас ім'ям ProjectN, в ній є папка src, в ній com, в ній example, в ній projectn (зауважте, що назва пакета розпалося на три папки, кожне слово, відокремлене крапкою визначило нову папку). І вже в папці projectn знаходиться файл MainActivity.java і будуть розміщуватися інші java-файли проекту.

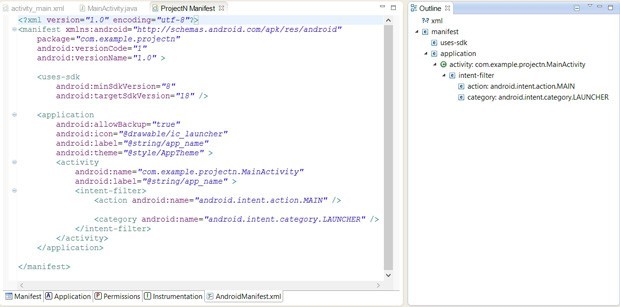
- папка gen - містить java-файли, які не потрібно змінювати і краще взагалі не чіпати. Ці файли генеруються автоматично. Нас може зацікавити файл R.java він містить ідентифікатори (ID) для всіх ресурсів програми.

- папка res - містить структуру папок ресурсів програми, розглянемо деякі з них:

- layout - в цій папці містяться xml-файли, які описують зовнішній вигляд форм і їх елементів, поки там знаходиться тільки activity\_main.xml;

- values ​​- містить XML файли, які визначають прості значення, таких ресурсів як, рядки, числа, кольору, теми, стилі, які можна використовувати в цьому проекті;

- menu - містить XML файли, які визначають все меню програми.

**Розглянемо файл AndroidManifest.xml - файл у форматі xml, який описує основні властивості проекту, дозвіл на використання ресурсів пристрою та ін. Відразу після створення програми файл AndroidManifest.xml виглядає так, як показано на рисунку.

**Крок 3 Налаштування інтерфейсу додатку**

До того, як почнемо формувати інтерфейс, має сенс підготувати можливість перевірки розробляється на помилки. Щоб не завантажувати кожен раз додаток на реальний пристрій, в Android SDK передбачена можливість використання віртуального пристрою (AVD або Android virtual device), емулює роботу реального смартфона.

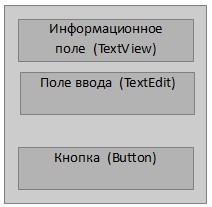
Процес створення віртуального пристрою або емулятора детально розписаний в "Установка і настройка середовища програмування ADT Bundle".

Прийшов час задуматися про зовнішній вигляд програми. Для початку необхідно визначити які елементи графічного інтерфейсу нам потрібні, як ці елементи будуть розташовуватися на формі і яким чином буде реалізовано взаємодію з користувачем.

Так як додаток дуже просте, то і інтерфейс особливою складністю відрізнятися не буде. Нам буде потрібно поле для введення чисел (TextEdit), текстова мітка для виведення інформації (TextView) і кнопка для підтвердження введеного числа (Button). Розташовувати елементи інтерфейсу будемо один під одним, зверху інформаційна частина, нижче поле введення, кнопку розмістимо в самому низу програми. Взаємодія програми з користувачем організовується дуже просто: користувач вводить число вполе для введення і натискає кнопку, читає результат в інформаційному полі і, або радіє перемозі, або вводить нове число.

Якщо користувач вводить правильне число, то додаток пропонує йому зіграти знову при цьому кнопка буде грати роль підтвердження, а в інформаційне поле буде виведено запрошення до повторної гри.

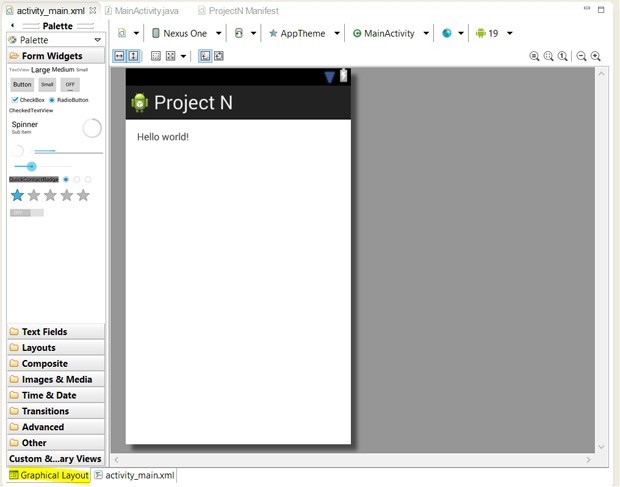
Схематично інтерфейс програми зображений далі:



Нам необхідно додати на форму три елементи: інформаційне поле (TextView), поле введення (TextEdit) і кнопку (Button).

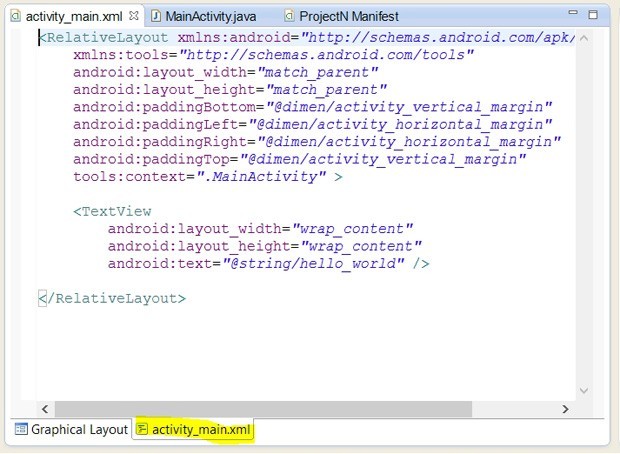
Android IDE підтримує два способи для виконання дій по формуванню інтерфейсу програми: перший заснований наXML-розмітці, другий належить до візуального програмування і дозволяє перетягувати об'єкти інтерфейсу і розміщувати їх на формі за допомогою миші. Вважається, що візуальний спосіб підходить для новачків, а більш просунуті розробники можуть писати код вручну, однак найчастіше використовується комбінований підхід.

Для формування інтерфейсу будемо працювати з файлом res / layout / activity\_main.xml. На рисунку можна побачити редактор, відповідний візуальному способу формування інтерфейсу, цього режиму відповідає вкладка Graphical Layout.

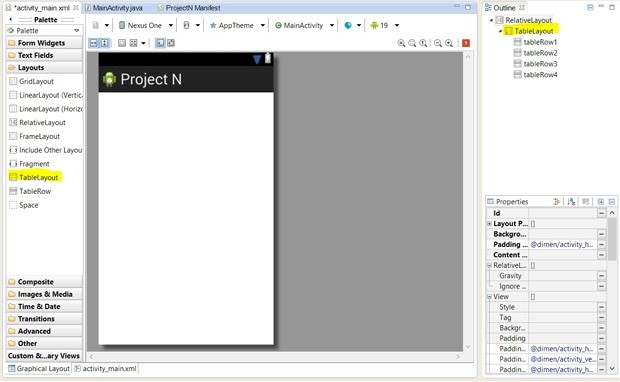


На рисунку поруч з вкладкою Graphical Layout розташована вкладка activity\_fullscreen.xml. Вона відповідає режиму редагування інтерфейсу шляхом формування XML файлу.

Тут можна побачити редактор XML файлу.



Задамо табличне розташування компонентів на формі, для цього виберемо вкладку Layouts, знайдемо там TableLayout і додамо його на форму. На рисунку можна побачити результат цих дій.

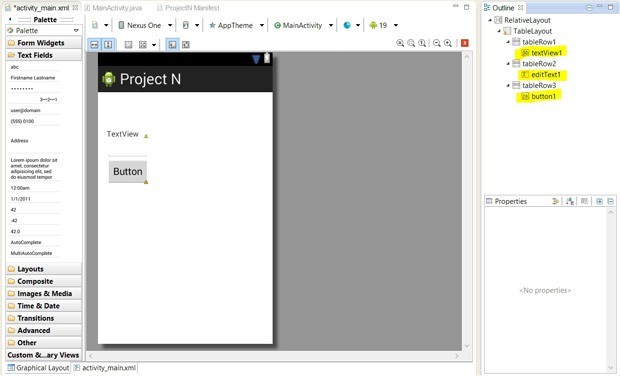


Тепер почнемо додавати елементи інтерфейсу, будемо використовувати графічний режим редагування.

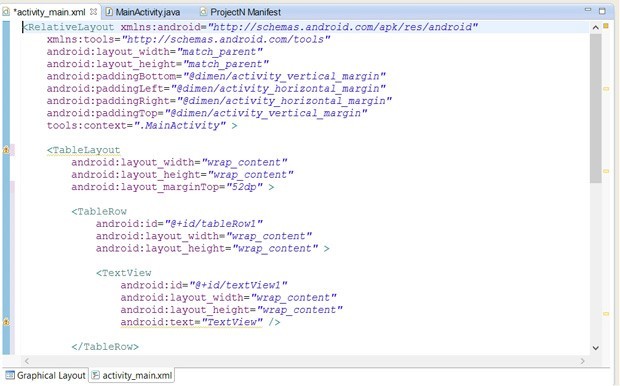
По-перше, нам необхідно додати інформаційне поле. Для цього на панелі = Palette = вибираємо вкладку Form Widgets, на цій вкладці знайдемо поле TextView, перенесемо у вікно програми, розмістимо в першому рядку таблиці (TableLayout).

По-друге, нам буде потрібно поле введення інформації, на вкладці Text Fields знайдемо текстове поле Number і розмістимо у другому рядку таблиці.

По-третє, повернемося на вкладку Form Widgets, виберемо там елемент Button і додамо в третій рядок таблиці. Чи не потрібну четвертий рядок таблиці видалимо, отримаємо наступний вигляд програми, див. рисунок.



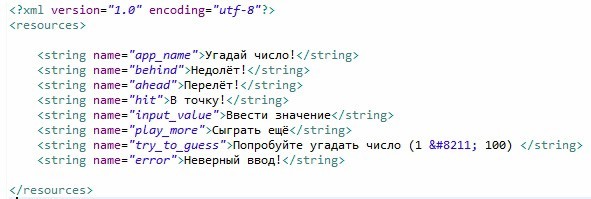
Після настройки інтерфейсу можна заглянути в файл activity\_main.xml, в цьому файлі прописано, що використовується TableLayout і дано опис кожної з трьох рядків. На рисунку можна побачити, як виглядає такий опис на прикладі першого рядка таблиці.



Тепер необхідно наповнити наші елементи інтерфейсу змістом, нам знадобиться текст для спілкування з користувачем, при програмуванні під Android існує практика розділяти ресурси і код програми. Для зберігання будь-яких рядків, які можуть знадобитися з додатком, використовується файл strings.xml. Зберігання всіх строкових ресурсів в цьому файлі серйозно полегшує локалізацію додатки на інші мови. Цей файл можна знайти в Package Explorer в папці res / values. Відкриємо його і подивимося, що там є:



Приберемо зайві рядки і додамо нові, результат можна побачити тут:



Дані змінні будуть виконувати такі завдання:

• app\_name встановить "видиме" назва додатка;

• behind, ahead, hit сповістять користувача про його успіхи в грі;

• play\_more і try\_to\_guess встановить назву кнопки, яке пояснить її функції;

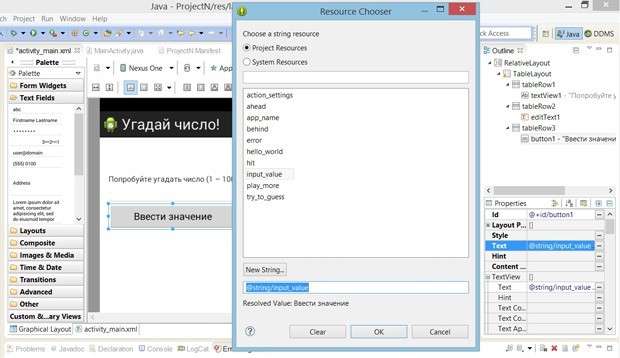
• input\_value запросить користувача до введення числа;

• error повідомить про невірному вводі.

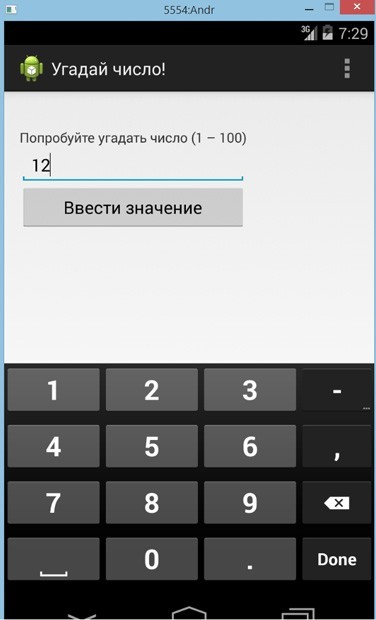
Після зміни strings.xml, при переході на іншу вкладку, не забудьте зберегти зміни (найшвидший спосіб - нажатьCtrl + S).

Налаштуємо текст в інформаційному полі. Для цього на вкладці Properties в правій частині вікна виберемо елемент textView1 (це і є наше інформаційне поле, має сенс придумати йому більш осмислене ім'я). Знайдемо властивість Text, підставимо в нього значення рядка з ім'ям try\_to\_guess.

Аналогічно можна налаштувати текст, яким нас вітатиме кнопка, тільки в цьому випадку треба працювати з елементом button1.



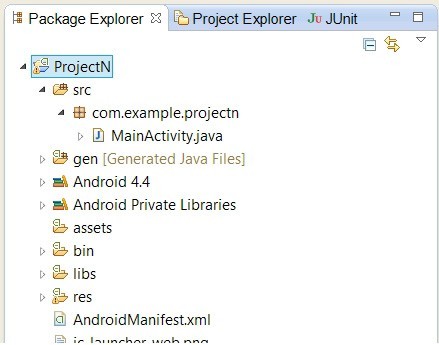
Настав час згадати про віртуальному пристрої, якщо воно працює, вже можна запустити проект і подивитися, як додаток буде виглядати на екрані пристрою, а виглядати воно може як показано на рисунку.



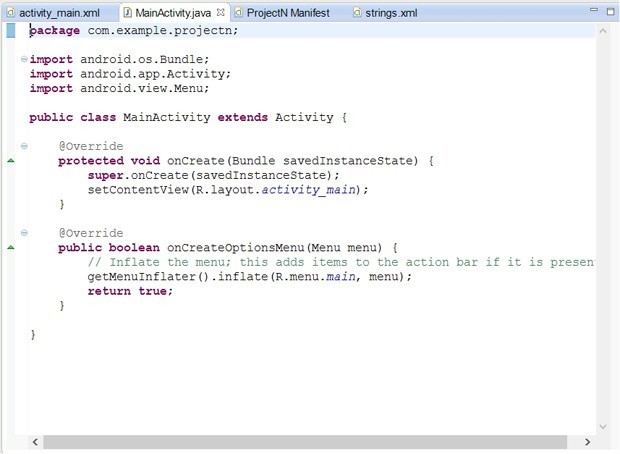
Додаток виглядає досить просто, але ми на багато і не розраховували. Головне, що нас цікавить, це наявність всіх елементів на екрані, вірний текст в кожному елементі, де він передбачений і можливість вводити числа в полі введення. Додаток є, його можна запустити на віртуальному чи реальному пристрої, але воно нічого не робить. Наступним кроком буде реалізація логіки додатка, тобто. Обробка події натискання на кнопку, як було прописано в завданні.

**Крок 4. Реалізація логіки додатку.**

Приступимо безпосередньо до програмування, працювати будемо з файлом src / com.example.projectn / MainActivity.java. Знайдемо цей файл в Package Explorer відкриємо і почнемо редагувати.



Поки файл виглядає наступним чином:



Можна помітити, що клас MainActivity є спадкоємцем класу Activity і в ньому вже реалізований методonCreate (), який запускається при початковому створенні активності, нам буде потрібно його доповнити, але про це трохи пізніше.

Ми припускаємо програмно змінювати інформацію в поле TextView, отримувати значення з поля EditText і обробляти події натискання на кнопку Button, тому необхідно оголосити відповідні змінні, як поля класу MainActivity:

*TextView tvInfo;*

*EditText etInput;*

*Button bControl;*

Щоб не було помилок, необхідно імпортувати пакет android.widget, який містить всі елементи графічного інтерфейсу:

*import android.widget. \*;*

Насправді середовище розробки підкаже, що робити.

Тепер необхідно пов'язати ці змінні з елементами інтерфейсу, вже доданими нами в activity\_main.xml, зробити це необхідно в методі onCreate (), а для отримання вже створеного елемента інтерфейсу скористаємося методомfindViewById (). Отже в метод onCreate () додамо наступні рядки:

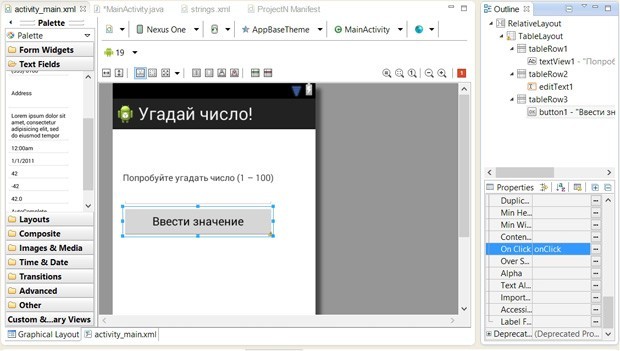
*tvInfo = (TextView) findViewById (R.id.textView1);*

*etInput = (EditText) findViewById (R.id.editText1);*

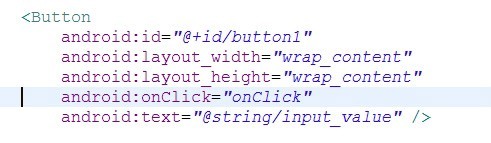
*bControl = (Button) findViewById (R.id.button1);*

Метод findViewById () повертає об'єкт класу View, який є спільним предком для всіх компонентів для користувача інтерфейсу, для того щоб уникнути можливих помилок в дужках перед викликом методу вказуємо до якого конкретно компонента необхідно звузити можливості об'єкта View.

Прийшов час виконати обробку натискання на кнопку. повернемося до файлу activity\_main.xml в графічний режим редагування, виберемо елемент Button і на вкладці з властивостями елемента знайдемо властивість On Click і запишемо в негоonClick - ім'я методу, який буде обробляти натискання на кнопку. Як це виглядає показує рисунок:



Ці ж дії можна виконати в файлі activity\_main.xml, досить дописати рядок:



Для настройки властивостей елементів інтерфейсу досить використовувати будь-який спосіб: графічний або редагування XMLфайла.

Повернемося в файл MainActivity.java, в клас активності необхідно додати метод:

*public void onClick (View v) {...}*

Ім'я методу не обов'язково має бути onClick (), головне, щоб воно збігалося з ім'ям, зазначеним у властивості On Click. У цьому методі і буде відбуватися все наше програмування в цій лабораторній роботі. Нам будуть потрібні дві змінні:

• цілочисельна для зберігання задуманого числа (випадкове число від 1 до 100);

• логічна для зберігання стану закінчена гра чи ні.

Обидві ці змінні має сенс оголосити як поля класу активності, початкові значення привласнити в методі onCreate.

Отримати цілочисельне значення з поля введення, можна за допомогою наступної конструкції:

*Integer.parseInt (etInput.getText (). ToString ())*

змінити значення тексту в інформаційному полі можна за допомогою наступної конструкції:

*tvInfo.setText (getResources (). getString (R.string.ahead));*

в даному випадку в інформаційному полі з'явиться значення строкового ресурсу з ім'ям ahead.

Залишилося реалізувати логіку додатка в методі onClick (). Пропонуємо написати код цього методу самостійно, для контролю в додатку запропонований лістинг, який містить один з варіантів коду описаного додатки.

**Крок 5 Трохи про роботу з емулятором**

При виконанні даної роботи ми використовували емулятор для перевірки працездатності програми. Розглянемо деякі можливості, що полегшують і прискорюють взаємодія з віртуальним пристроєм.

По-перше, існує набір корисних комбінацій клавіш для управління віртуальним пристроєм:

• Alt + Enter - розгортає емулятор до розмірів екрану;

• Ctrl + F11 - змінює орієнтацію емулятора з портретної на альбомну і назад;

• F8 - включає / вимикає мережу.

Повний список комбінацій клавіш для роботи з емулятором можна знайти за посиланням: http: //developer.android.com/tools/help/emulator.html.

По-друге, хто б не працював з емулятором, той на собі відчув наскільки терплячим треба бути, щоб взаємодіяти з ним, так повільно він працює. Існує рішення для прискорення роботи емулятора Android і цим рішенням є

Intel Hardware Accelerated Execution Manager (Intel® HAXM).

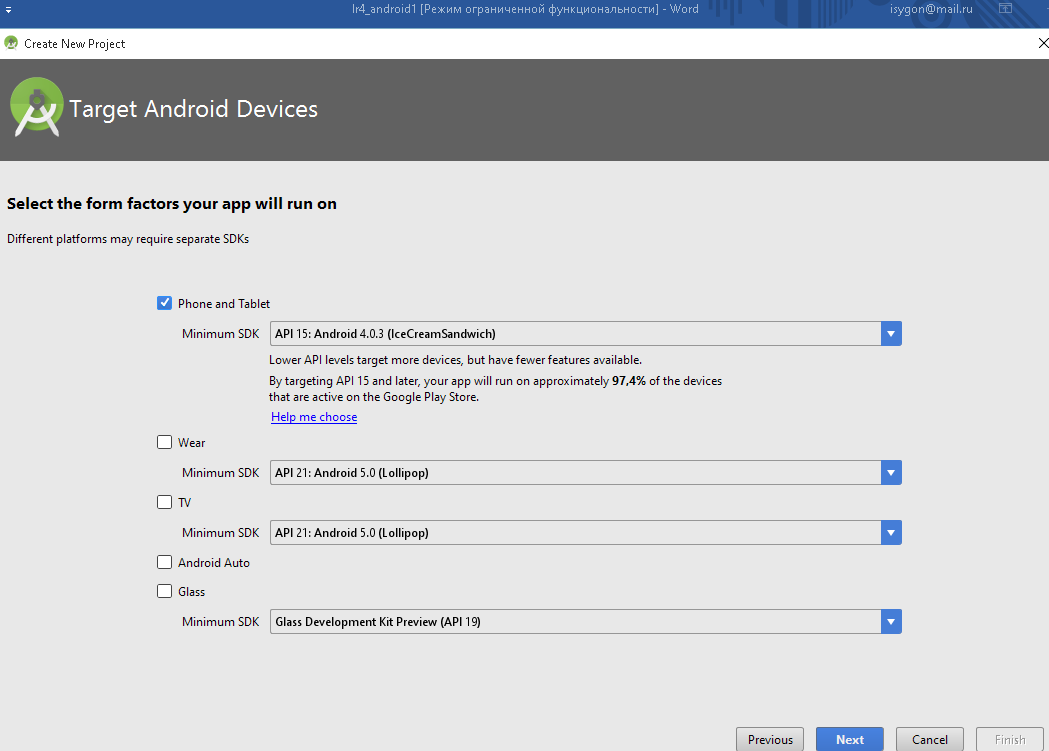
Intel Hardware Accelerated Execution Manager (Intel® HAXM) - це додаток з підтримкою апаратної віртуалізації (гипервизор), яке використовує технологію віртуалізації Intel для прискорення емуляції додатків Android на комп'ютері для розробки. (Http://software.intel.com/ru-ru/android/articles/intel-hardware-accelerated-execution- manager)

Intel HAXM здатний прискорити роботу емулятора для x86 пристроїв. При цьому емулятор буде працювати зі швидкістю, наближеною до швидкості роботи реального пристрою, що допоможе скоротити час на запуск і налагодження програми. Детально ознайомитися з установкою Intel HAXM і налаштуванням емулятора на роботу з прискорювачем можна в статті за посиланням: http: //habrahabr.ru/company/intel/blog/146114/.

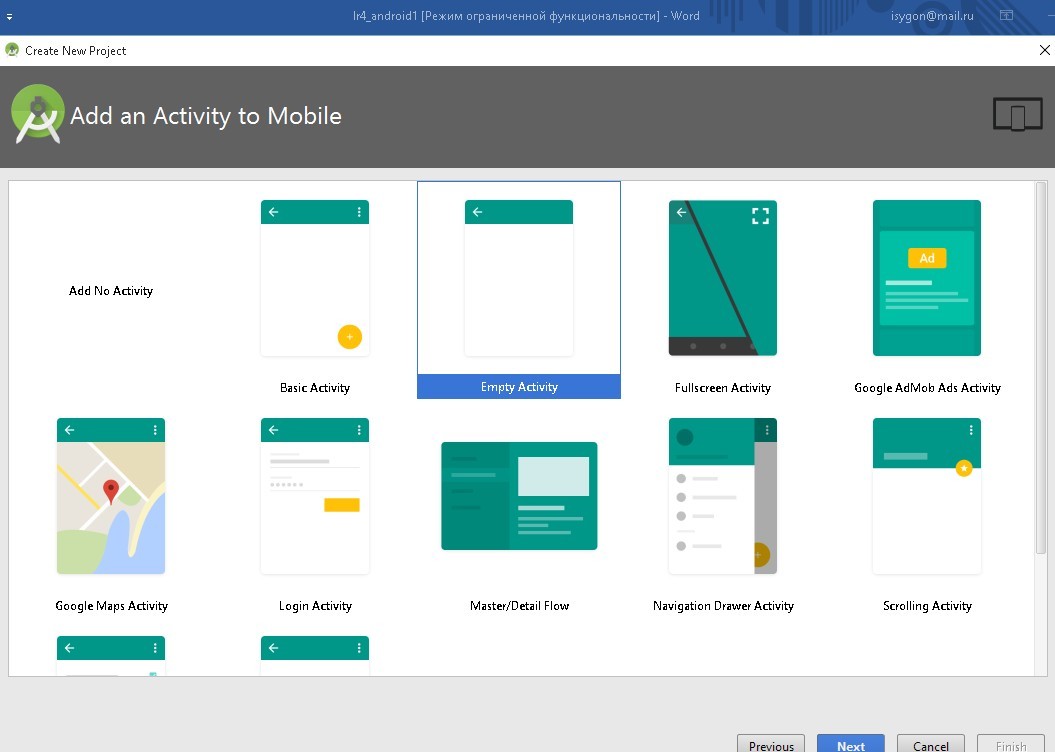
По-третє, не варто переривати процес запуску віртуального пристрою, наберіться терпіння, на старих комп'ютерах перший запуск AVD може зайняти до 10 хвилин, на сучасних - від однієї до трьох хвилин. Після того, як створили і запустили віртуальне пристрій має сенс залишити його відкритим, при всіх наступних запусках програми буде використовуватися вже відкрито віртуальний пристрій, що дозволить заощадити час.

**Завдання 2. Проектування інтерфейсу додатку**

1. Створюємо новий проект



2. Додаємо активність Empty Activity



Назву активності – MainActivity за бажанням можна змінити.

3. Додаємо в розмітку активності елементи, що забезпечать ефективне введення даних.

Для зразка нами вибрано тему – успішність студентів. Необхідно ввести рейтингові оцінки успішності групи студентів за наступною структурою – Прізвище та ініціали студента, Група, Дисципліна, Оцінка, Вид роботи.

Обрані елементи:

П.І.Б. – EditText; Група - Spinner

Дисципліна – AutoCompleteTextView

Оцінка – EditText

Вид роботи - RadioButton

Для підписів полів використовується елемент TextView. Основні завдання, що необхідно для цього виконати:

1. Створити макет, що визначає зовнішній вид додатку.

2. Визначити необхідні строкові та ін. ресурси

3. Визначити стилі для однотипних елементів

4. Визначити методи активності, для виконання дій.

Визначимо макет активності у файлі ActivityMain.xml (розташування res/lauout). Для зручності можна використовувати візуальний редактор.

В такому випадку основні властивості елементів, що додаються – id, text.

Для елемента Spinner створити у файлі strings.xml (розташування res/values) строковий ресурс наступного типу:

<**string-array name="name\_Group"**>

<**item**>ПІ-49</**item**>

<**item**>ПІ-50</**item**>

<**item**>ПІ-51</**item**>

<**item**>ПІ-52</**item**>

<**item**>ПІ-53</**item**>

</**string-array**>

Аналогічно для елемента аutoCompleteTextView:

<**string-array name="name\_Subject"**>

<**item**>Вища математика</**item**>

<**item**>Фізика</**item**>

<**item**>Основи програмування</**item**>

<**item**>Бази даних</**item**>

<**item**>Людино-машина взаємодія</**item**>

</**string-array**>

Для вищезазначених елементів у файлі розмітки activity\_main.xml додаємо прив’язку до ресурсів:

<**AutoCompleteTextView android:id="@+id/acTWSubject" android:hint="Оберіть дисципліну"**

**…**

**android:entries="@array/name\_Subject"**/>

<**Spinner**

**android:id="@+id/spGroup"**

**…**

**android:entries="@array/name\_Group"**/>

Зразок макета створеного за допомогою візуального редактора:

*<?xml version="1.0" encoding="utf-8"?>*

*<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android" xmlns:tools="http://schemas.android.com/tools" android:layout\_width="match\_parent" android:layout\_height="match\_parent" android:paddingBottom="@dimen/activity\_vertical\_margin" android:paddingLeft="@dimen/activity\_horizontal\_margin" android:paddingRight="@dimen/activity\_horizontal\_margin" android:paddingTop="@dimen/activity\_vertical\_margin" tools:context="com.example.valkiriya.myapplication.MainActivity">*

*<TextView*

*android:layout\_width="wrap\_content"*

*android:layout\_height="wrap\_content" android:textAppearance="?android:attr/textAppearanceMedium"*

*<!—Робота з розмірами шрифта стилі textAppearance http://developer.alexanderklimov.ru/android/theory/scales.php-->*

*android:text="@string/capt\_Stud" android:id="@+id/twStud" android:layout\_alignParentTop="true"*

*<EditText*

*android:layout\_width="wrap\_content" android:layout\_height="wrap\_content" android:inputType="textPersonName" android:ems="10" android:id="@+id/Stud\_name" android:layout\_below="@+id/twStud" android:layout\_alignParentLeft="true" android:layout\_alignParentStart="true" android:hint="П.І.Б." />*

*<TextView*

*android:layout\_width="wrap\_content"*

*android:layout\_height="wrap\_content" android:textAppearance="?android:attr/textAppearanceMedium" android:text="@string/capt\_Group"*

*android:id="@+id/twGroup"*

*android:layout\_below="@+id/Stud\_name" android:layout\_alignParentLeft="true"*

*android:layout\_alignParentStart="true" />*

*<AutoCompleteTextView android:layout\_width="wrap\_content"*

*android:layout\_height="wrap\_content" android:text="New AutoCompleteTextView" android:id="@+id/acTWSubject" android:layout\_below="@+id/twSubject" android:layout\_alignParentLeft="true"*

*android:layout\_alignParentStart="true" android:hint="Оберіть дисципліну"*

*android:entries="@array/name\_Subject"/>*

*<Spinner*

*android:layout\_width="wrap\_content" android:layout\_height="wrap\_content" android:id="@+id/spGroup" android:layout\_below="@+id/twGroup" android:layout\_alignParentLeft="true" android:layout\_alignParentStart="true" android:entries="@array/name\_Group"/>*

*<TextView*

*android:layout\_width="wrap\_content"*

*android:layout\_height="wrap\_content" android:textAppearance="?android:attr/textAppearanceMedium" android:text="@string/capt\_Subject" android:id="@+id/twSubject"*

*android:layout\_below="@+id/spGroup" android:layout\_alignParentLeft="true"*

*android:layout\_alignParentStart="true" />*

*<TextView*

*android:layout\_width="wrap\_content"*

*android:layout\_height="wrap\_content" android:textAppearance="?android:attr/textAppearanceMedium" android:text="@string/capt\_Reiting" android:id="@+id/twReiting"*

*android:layout\_below="@+id/acTWSubject" android:layout\_alignParentLeft="true"*

*android:layout\_alignParentStart="true" />*

*<EditText*

*android:layout\_width="wrap\_content" android:layout\_height="wrap\_content" android:inputType="number" android:ems="10" android:id="@+id/etReiting" android:layout\_below="@+id/twReiting" android:layout\_alignParentLeft="true" android:layout\_alignParentStart="true" android:hint="Оцінка" />*

*<RadioGroup*

*android:layout\_width="wrap\_content" android:layout\_height="wrap\_content" android:id="@+id/rgWork" android:layout\_alignParentLeft="true" android:layout\_alignParentStart="true" android:layout\_below="@+id/etReiting">*

*<RadioButton*

*android:layout\_width="wrap\_content" android:layout\_height="wrap\_content" android:text="@string/capt\_Exam" android:id="@+id/rbExam" android:layout\_below="@+id/etReiting" android:layout\_alignParentLeft="true" android:layout\_alignParentStart="true" />*

*<RadioButton*

*android:layout\_width="wrap\_content" android:layout\_height="wrap\_content"*

*android:text="@string/capt\_Zalik" android:id="@+id/rbZalik" android:layout\_below="@+id/rbExam" android:layout\_alignParentLeft="true" android:layout\_alignParentStart="true" />*

*</RadioGroup>*

*<Button*

*android:layout\_width="wrap\_content" android:layout\_height="wrap\_content" android:text="@android:string/ok" android:id="@+id/btSave" android:layout\_below="@+id/etReiting" android:layout\_alignParentLeft="true" android:layout\_alignParentStart="true" android:layout\_marginTop="64dp" />*

*</RelativeLayout>*

Використані текстові ресурси strings.xml:

*<resources>*

*<string name="app\_name">My Application</string>*

*<string name="capt\_Stud">П.І.Б. студента</string>*

*<string name="capt\_Group">Група</string>*

*<string name="capt\_Subject">Предмет</string>*

*<string name="capt\_Reiting">Рейтинг</string>*

*<string name="capt\_Exam">Іспит</string>*

*<string name="capt\_Zalik">Залік</string>*

*<string-array name="name\_Group">*

*<item>ПІ-49</item>*

*<item>ПІ-50</item>*

*<item>ПІ-51</item>*

*<item>ПІ-52</item>*

*<item>ПІ-53</item>*

*</string-array>*

*<string-array name="name\_Subject">*

*<item>Вища математика</item>*

*<item>Фізика</item>*

*<item>Основи програмування</item>*

*<item>Бази даних</item>*

*<item>Людино-машина взаємодія</item>*

*</string-array>*

*</resources>*

Для роботи автопідбору тексту в події OnCreate для MainActivity визначимо передачу масиву:

AutoCompleteTextView textView = (AutoCompleteTextView) findViewById(R.id.***acTWSubject***);

String[] subjects = getResources().getStringArray(R.array.***name\_Subject***); ArrayAdapter<String> adapter = **new** ArrayAdapter<String>(**this**, R.layout.***support\_simple\_spinner\_dropdown\_item***, subjects); textView.setAdapter(adapter);

Для забезпечення функціонування автопідбору необхідно імпортувати класи:

*import android.widget.AutoCompleteTextView;*

*import android.widget.ArrayAdapter;*

Для зменшення кількості рядків в макеті додатку використаємо стилі. В файлі styles.xml (розташування res/values) створимо новий стиль – Caption та визначимо його параметри наступним чином:

*<style name="Caption" parent = "@android:style/TextAppearance.Medium">*

*<item name = "android:layout\_width">wrap\_content</item>*

*<item name = "android:layout\_height">wrap\_content</item>*

*<item name="android:layout\_alignParentLeft">true</item>*

*<item name="android:layout\_alignParentStart">true</item>*

*<item name="android:textColor">@color/colorCaption</item>*

*</style>*

В файлі colors.xml (розташування res/values) визначимо нову змінну

*@color/colorCaption: <color name="colorCaption">#1d227e</color>*

І знищимо з макету дублювання коду елементів TextView які було винесено в стиль (позначені зеленим у лістингу). Натомість додамо елемента прив'язки до стилю:

*style="@style/Caption"*

Таким чином визначення TextView скоротилося до 4 параметрів id, text, below, style. Аналогічним чином можна відформатувати інші елементи інтерфйсу або взагалі створити нову тему для оформлення макетів.

Для збереження записів визначимо обробник події для кнопки Ок: В макет додаємо привязку до метода SaveRecord:

*<Button*

*android:text="@android:string/ok" android:id="@+id/btSave"*

*… android:onClick="SaveRecord"/>*

І в MainActivity.java – визначаємо обробник події:

*public void SaveRecord(View view){*

*Intent intent = new Intent(this, ReportActivity.class);*

//поки нову активність не додано не працює

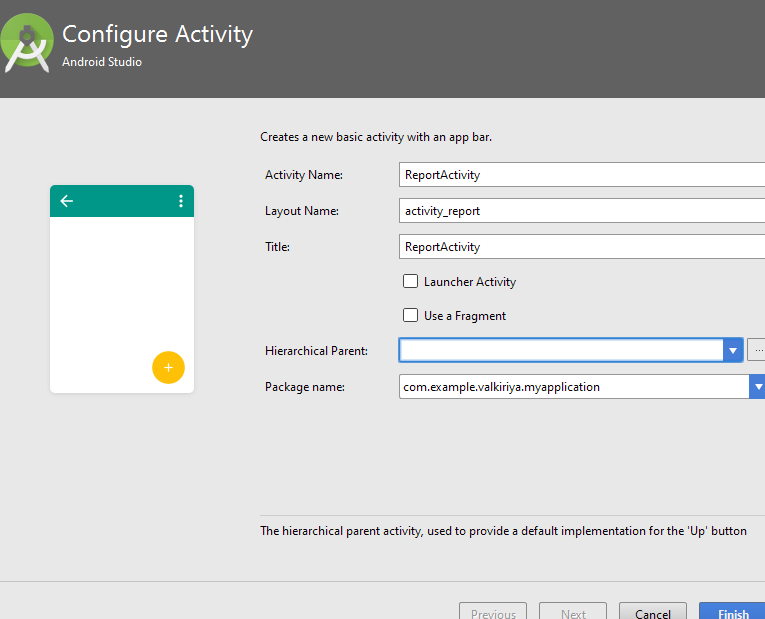
*startActivity(intent);*

*}*

Увага! для роботи з інтентом необхідно імпортувати клас:

*import android.content.Intent;*

Додати нову активність можна використовуючи Файл->New->Activities



В макеті активності визначаємо єдиний об’єкт - TextView наступним чином:

*<TextView*

*android:layout\_width="match\_parent" android:layout\_height="match\_parent" android:id="@+id/textStudent" android:layout\_alignParentTop = "true" android:layout\_alignParentLeft="true"*

###### */>*

Для отримання повідомлення опишемо у події onCreate процес отримання зовнішнього повідомлення:

Intent intent = getIntent();

String MessageText = intent.getStringExtra(***STUDENTS\_DATA***); TextView tv = (TextView)findViewById(R.id.***textStudent***); tv.setText(MessageText);

Змінну STUDENTS\_DATA – визначимо як атрибут класу ReportActivity:

**public class** ReportActivity **extends** AppCompatActivity {

**public static final** String ***STUDENTS\_DATA*** =**"Hello"**;

###### …

Для роботи під’єднаємо наступні класи:

*import android.view.View;*

*import android.widget.TextView;*

Загальний вигляд класу ReportACtivity.java

package com.example.valkiriya.myapplication;

*import android.content.Intent;*

*import android.os.Bundle;*

###### *import android.support.v7.app.AppCompatActivity;*

*import android.view.View;*

*import android.widget.TextView;*

*public class ReportActivity extends AppCompatActivity {*

*public static final String STUDENTS\_DATA ="Hello";*

###### *@Override*

*protected void onCreate(Bundle savedInstanceState) {*

###### *super.onCreate(savedInstanceState);*

*setContentView(R.layout.****activity\_report****);*

###### *Intent intent = getIntent();*

*String MessageText = intent.getStringExtra(****STUDENTS\_DATA****); TextView tv = (TextView)findViewById(R.id.****textStudent****); tv.setText(MessageText);*

*}*

*}*

**Контрольні питання**

1. Фонові програми ...

(Відзначте один правильний варіант відповіді.)

1. після настройки не припускають взаємодії з користувачем, більшу частину часу перебувають і працюють в прихованому стані;
2. невеликі додатки, які відображаються у вигляді графічного об'єкта на робочому столі;
3. виконують свої функції і коли видимі на екрані, і коли приховані іншими додатками;
4. більшу частину часу працюють у фоновому режимі, проте допускають взаємодію з користувачем і після настройки.
5. Яка папка в структурі Android-додатки містить файли з вихідним кодом на мові Java?
6. gen;
7. src;
8. bin;
9. res.
10. Взаємодія користувача з системою (не тільки комп'ютерної) складається з семи кроків. Упорядкуйте послідовність:
11. формування мети дій;
12. визначення загальної спрямованості дій;
13. визначення конкретних дій;
14. виконання дій;
15. сприйняття нового стану системи;
16. інтерпретація стану системи;
17. оцінка результату.
18. Для продовження роботи користувач повинен знати:
19. на якому етапі він зупинився;
20. які команди і параметри він вже дав системі;
21. що саме він повинен зробити на поточному кроці;
22. куди було звернуто його увагу на момент відволікання;
23. що йому робити далі;
24. стан системи в поточний момент;
25. результати виконання попередніх операцій;
26. навіщо він виконує роботу.
27. Способи боротьби з помилками:
28. блокування потенційно небезпечних дій користувача до отримання підтвердження правильності дії;
29. перевірка системою всіх дій користувача перед їх прийняттям;
30. самостійний вибір системою необхідних команд або параметрів, при якому від користувача потрібно тільки перевірка;
31. створення системи сигнальних повідомлень;
32. реалізація можливості скасування останньої дії.

# Лабораторна робота № 4

**Тема роботи:** Основи програмування під ОС Android.

**Мета роботи:** Тестування на смартфоні під ОС Android.

**Обладнання:** Смартфон, ОС Android, AndroidDebugBridge.

## 1. Завдання на лабораторну роботу

Завдання 1. Ознайомлення зі структурою проекту.

Завдання 2. Тестування на реальному пристрою.

## Хід роботи

Метою даного завдання є написання мінімального додатку під Android. Але ми ніколи не будемо точно знати, чи змогли ми написати щось працездатне, не запустивши його на реальному пристрої. Цим ми і займімося в цьому завданні.

Можливість тестування на смартфоні надається ADB (AndroidDebugBridge). Налаштуємо його і запустимо наш додаток на цьому смартфоні.

**Налаштування Android-пристроя для роботи з ADB**

Для того, щоб використовувати ADB з пристроєм, підключеним по USB, необхідно дозволити USB-налагодження в системних наслаштуваннях телефону або планшета в розділі "Параметри розробника" (назва може відрізнятися). Залежно від телефону, цей розділ за замовчуванням прихований. Розглянемо кроки в разі, коли немає потрібного розділу налаштувань.

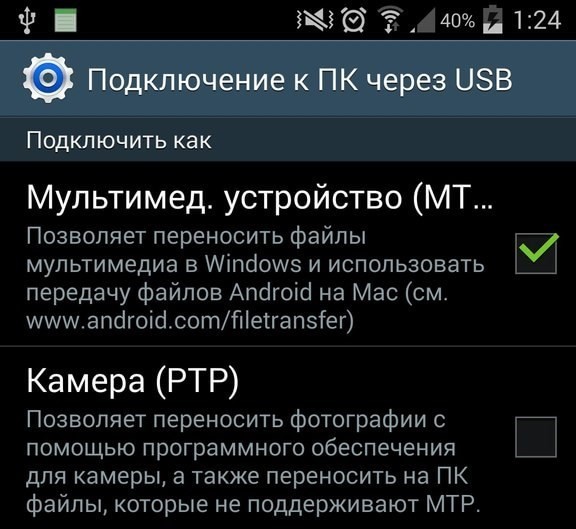
1. Зайдіть в налаштування, розділ "Інформація про пристрій"

2. Знайдіть пункт "Номер збірки" і клацніть по ньому 7 разів. Повинно з'явитися вікно, оповіщає про те, що активований режим розробника.

Тепер в налаштуваннях має з'явитися розділ параметрів розробника.

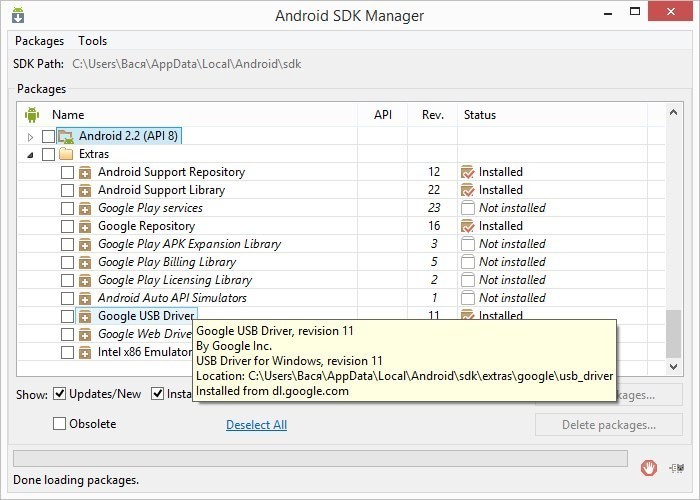
3. Включіть в ньому опцію "Налагодження USB".

Тепер, коли ви підключаєте пристрій до комп'ютера, в залежності від моделі у вас може з'явитися новий варіант підключення.



**Налаштування ADB на ОС Windows**

При налаштуванні ОС Windows, по-перше, переконайтеся, що у вас встановлений Google USB Driver.

Зайдіть в SDK Manager в розділ Extras і знайдіть Google USB Driver, встановіть його в якщо буде потреба.

Тепер слід оновити драйвер. Підключіть девайс до комп'ютера, перейдіть в Панель управління -> Обладнання та звук -> Диспетчер пристроїв знайдіть свій пристрій.

Клацніть правою клавішею за своїм устроєм, щоб відкрити контекстне меню і "Виберіть Оновити драйвери ...". Драйвер можна знайти в директорії sdk в папці \{Директорія sdk} \ extras \ google \ usb\_driver.

Як перевірити правильність налаштувань ADB?

Для перевірки працездатності ADB підключіть пристрій до комп'ютера, запустіть в папці \ {директорія sdk} \ platform-tools командний рядок і введіть в неї команду:

adb devices

Повинен з'явиться список на зразок цього:

List of devices attached

5cdb2ced

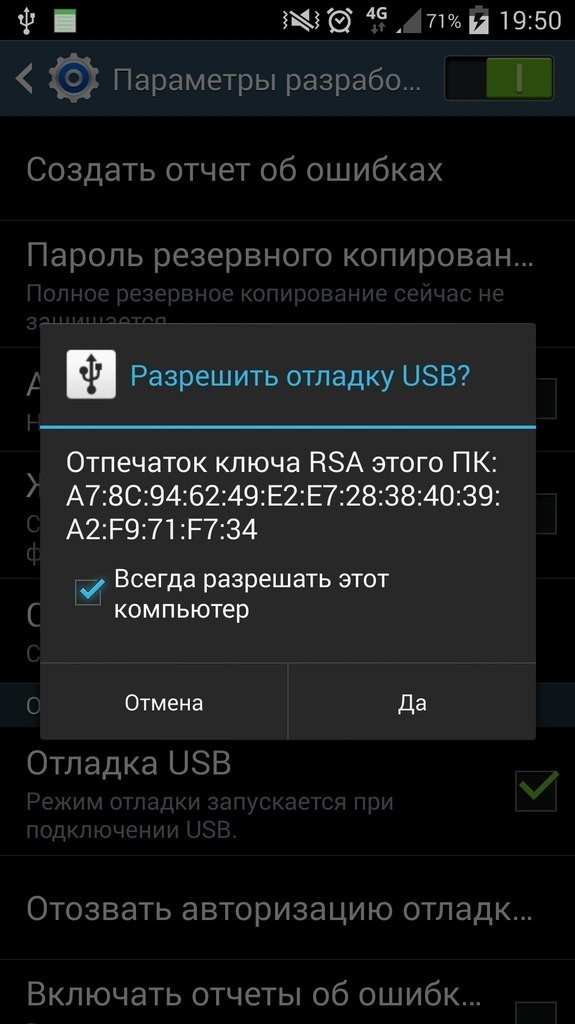
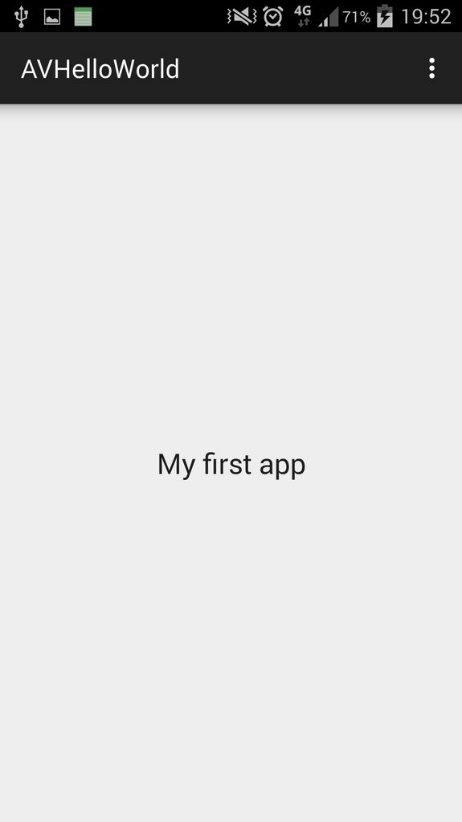
device

**Запуск програми на реальному пристрої**

Все те ж саме, що і в разі запуску на емуляторі. Відкрийте в Android Studio наш проект, натисніть на зелений трикутник, але у вікні вибору пристрою виберіть ваш девайс.



Якщо написано, що девайс ofine, ще раз вмикніть USB і дозвольте USB-налагодження комп'ютера:

В результаті на екрані телефону або планшета з’явиться наш додаток.

**Контрольні питання**

1. Типовий процес прототипирования, включає наступні елементи:
2. створення ескізів;
3. оцінку;
4. створення моделі;
5. випробування;
6. реалізація інтерфейсу;
7. реалізація обробки подій;
8. тестування пріпложенія.
9. Для запуску додатків, розроблених в Android IDE, необхідно:
10. налаштувати середу і запустити проект на пристрої
11. запустити віртуальну машину з встановленою ОС Android
12. налаштувати комп'ютер (для Windows необхідно встановити потрібний драйвер вручну, потрібні права адміністратора)
13. налаштувати пристрій (включити режим налагодження по USB)
14. Яку назву отримала версія Android 4.4?

(Відзначте один правильний варіант відповіді.)

1. Krispy Kreme
2. kream-karamel
3. kitekat
4. Kit Kat
5. Яка графічна бібліотека входить в набір бібліотек ОС Android?
6. OpenCV
7. DirectX
8. Open GL
9. OpenCL
10. Додатки, які не мають GUI і працюють у фоновому режимі - це
11. Intents;
12. Content Providers;
13. Activities;
14. Services;

**Список літератури**

.

1. Ткачук М.В. Уніфіковани програмні сервіси та візуальні інтерфейси в інтранет-системах управління технологічними процесами - Системні дослідження та інформаційні технології -№1 - 2004.

2. Доценко С. І. Людино-машинний інтерфейс: навч. посібник. – Харків: УкрДУЗТ, 2022. – 135 с.

3. Інтерфейс "Користувач-комп'ютер": Навчальний посібник / В.П. Майданюк, А.М. Пєтух. - Вінниця: ВДТУ, 1999. - 66 с.

4. Пупена О.М. Розроблення людино-машинних інтерфейсів та систем збирання даних з використанням програмних засобів SCADA/HMI. Навчальний посібник / Пупена О.М. - Видавництво: Ліра-К., 2020. - 594 с.

Навчально-методичне видання

**МЕТОДИЧНІ РЕКОМЕНДАЦІЇ**

**ДЛЯ ВИКОНАННЯ ЛАБОРАТОРНИХ РОБІТ**

**З ДИСЦИПЛІНИ**

**“Людино-машинна взаємодія”**

Підготували

**Гришкун Євгеній Олександрович**

**Ковальчук Андрій Михайлович**

**Сугоняк Інна Іванівна**

Технічний редактор – **І.І.Сугоняк**

Комп’ютерне верстання **– Є.О.Гришкун**

Підписано до друку 15.02.19. Формат 60×84/16.

Ум. друк. арк. 3.016. Зам. офс.