|  |  |  |
| --- | --- | --- |
|  |  |  |
| МИНОБРНАУКИ РОССИИ | | |
| Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«МИРЭА – Российский технологический университет»**  **РТУ МИРЭА** | | |

Институт Информационных технологий

Кафедра Математического обеспечения и стандартизации информационных технологий

**Отчет по практической работе №4**

по дисциплине «Разработка мобильных приложений»

|  |  |
| --- | --- |
| **Выполнил:**  Студент группыИКБО-21-23 | Муравьев А. О. |
| **Проверил:**  Старший преподаватель кафедры МОСИТ | Шешуков Л.С. |

Москва 2025 г.

**СОДЕРЖАНИЕ**

[1 ТЕОРЕТИЧЕСКАЯ ЧАСТЬ 4](#_Toc195872138)

[1.1 BottomBar 4](#_Toc195872139)

[1.2 ActionBar 15](#_Toc195872140)

[1.3 Navigation Drawer 21](#_Toc195872141)

[2 ПРАКТИЧЕСКАЯ ЧАСТЬ 33](#_Toc195872142)

[2.1 Подготовка ресурсов 33](#_Toc195872143)

[ЗАКЛЮЧЕНИЕ 34](#_Toc195872144)

* ВВЕДЕНИЕ

В данной практической работе предполагается изучение основ разработки Android-приложений через реализацию нескольких Activity. Первое задание включает создание двух экранов: на первом отображается список категорий продуктов, при выборе которых пользователь переходит ко второму экрану с подробным списком элементов выбранной категории и возможностью добавления и удаления элементов. Во втором задании акцентируется внимание на использовании RecyclerView для отображения данных с изображениями и текстом. Третье задание посвящено созданию Activity с ScrollView, содержащим элементы TextView и EditText для демонстрации прокрутки и ввода текста. В четвертом задании разрабатывается Activity с выпадающим списком (Spinner) для выбора значений из предустановленного списка. Все задания направлены на углубление понимания разработки Android-приложений и создания интуитивно понятного интерфейса.

# ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

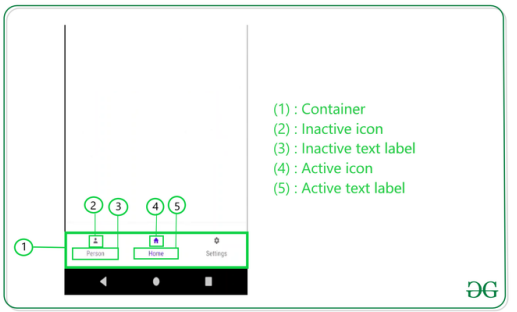
## BottomBar

В предыдущих практических работах были рассмотрены переходы между разными экранами в приложении с использованием простых элементов Button. Однако, в контексте современной Android разработки такой метод является не интуитивным и устаревшим. Поэтому, на замену такой навигации были придуманы объекты пользовательского интерфейса BottomBar и Drawer.

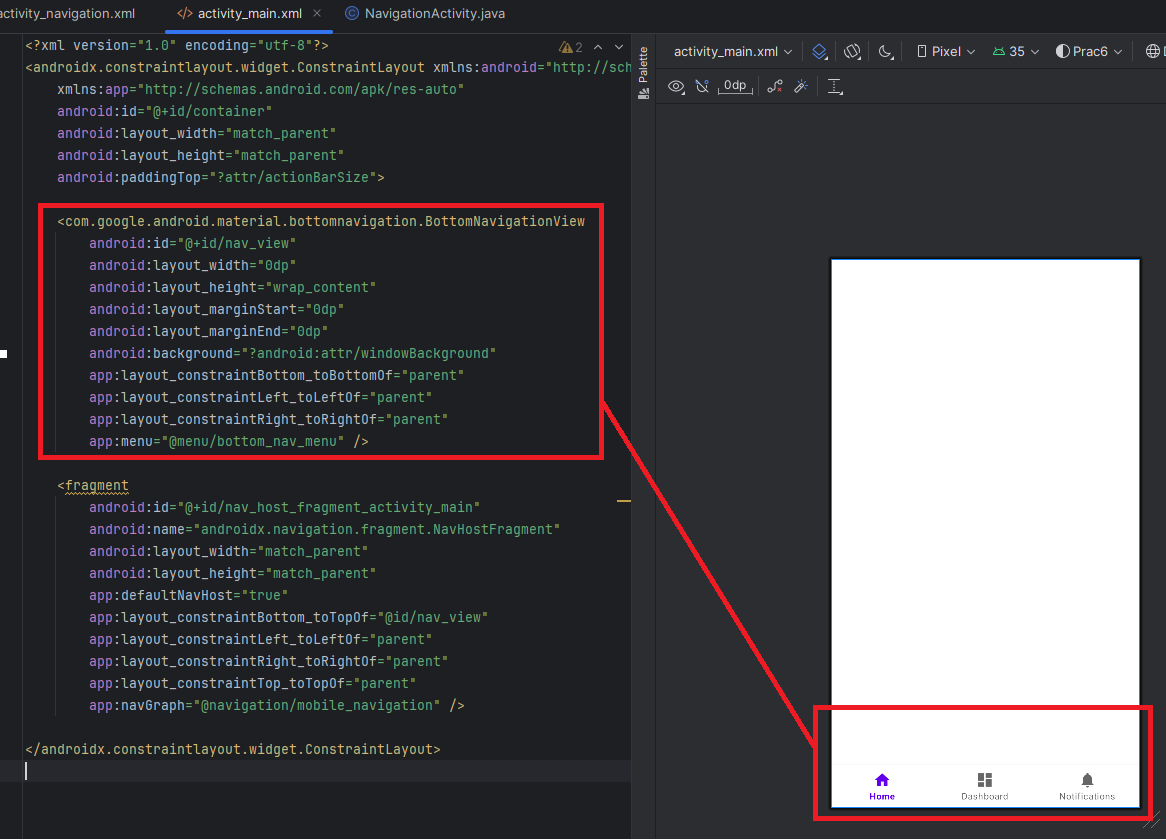
BottomBar, или нижняя панель навигации, является ключевым элементом пользовательского интерфейса в мобильных приложениях, работающих под управлением Android. Этот элемент предназначен для улучшения навигации и повышения удобства использования приложения за счёт предоставления быстрого доступа к основным разделам приложения. В контексте перемещения между экранами BottomBar выполняет несколько важных функций:

* улучшение пользовательского опыта: BottomBar делает навигацию по приложению интуитивно понятной и удобной. Располагаясь в нижней части экрана, он легко доступен для пользователя, что особенно важно при использовании устройства одной рукой. Это облегчает переход между ключевыми разделами приложения, не отвлекая пользователя от основного контента;
* повышение производительности: использование BottomBar позволяет сократить количество нажатий, необходимых для перехода между разделами приложения, тем самым ускоряя взаимодействие пользователя с приложением и повышая его общую производительность;
* организация контента: BottomBar помогает организовать контент в приложении, выделяя основные разделы или функции, которые должны быть всегда под рукой. Это упрощает структуру приложения и делает его более понятным для пользователя.

BottomBar представляет собой контейнер, который содержит в себе различные элементы (рисунок 1.1.1).

  
Рисунок 1.1.1 – Наполнение BottomBar-а

Для использования BottomNavigationView необходимо добавить этот элемент в макет активности (рисунок 1.1.2).

  
Рисунок 1.1.2 – Описание BottomNavigationView в файле разметки

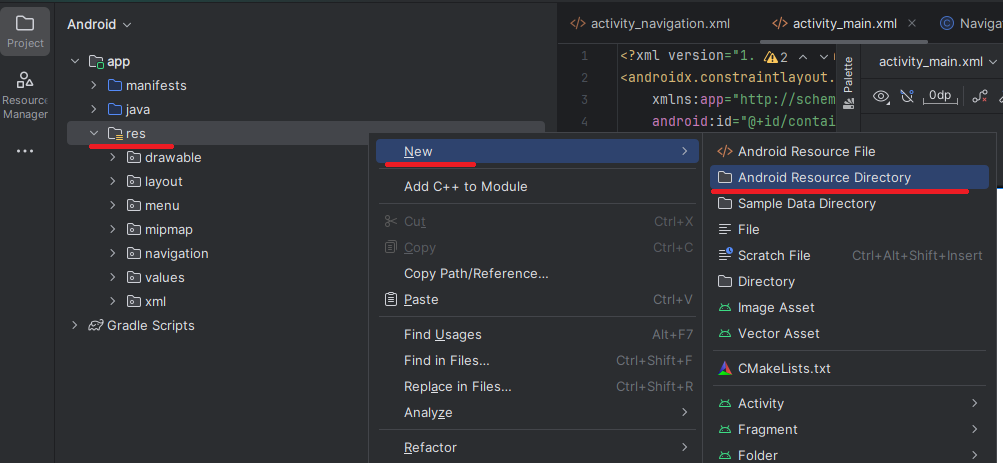
CoordinatorLayout – это контейнер, который расширяет FrameLayout.

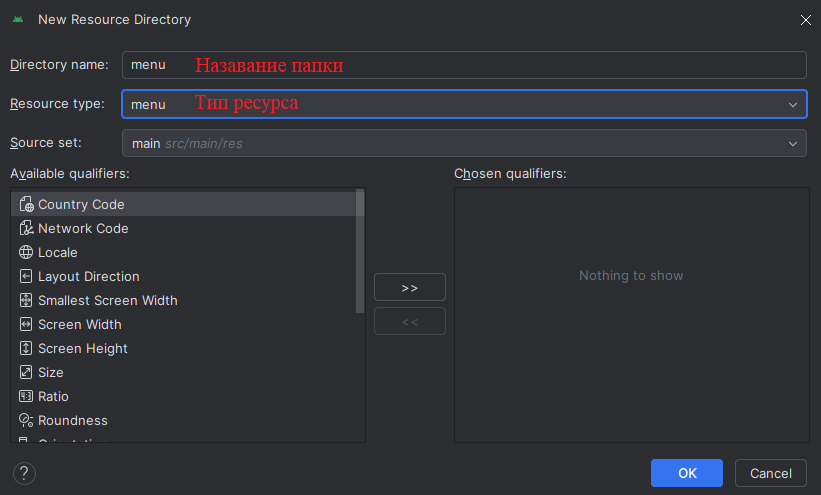
CoordinatorLayout предназначен для двух основных вариантов использования:

* в качестве декора приложения верхнего уровня или макета Chrome;
* как контейнер для определенного взаимодействия с одним или несколькими дочерними представлениями.

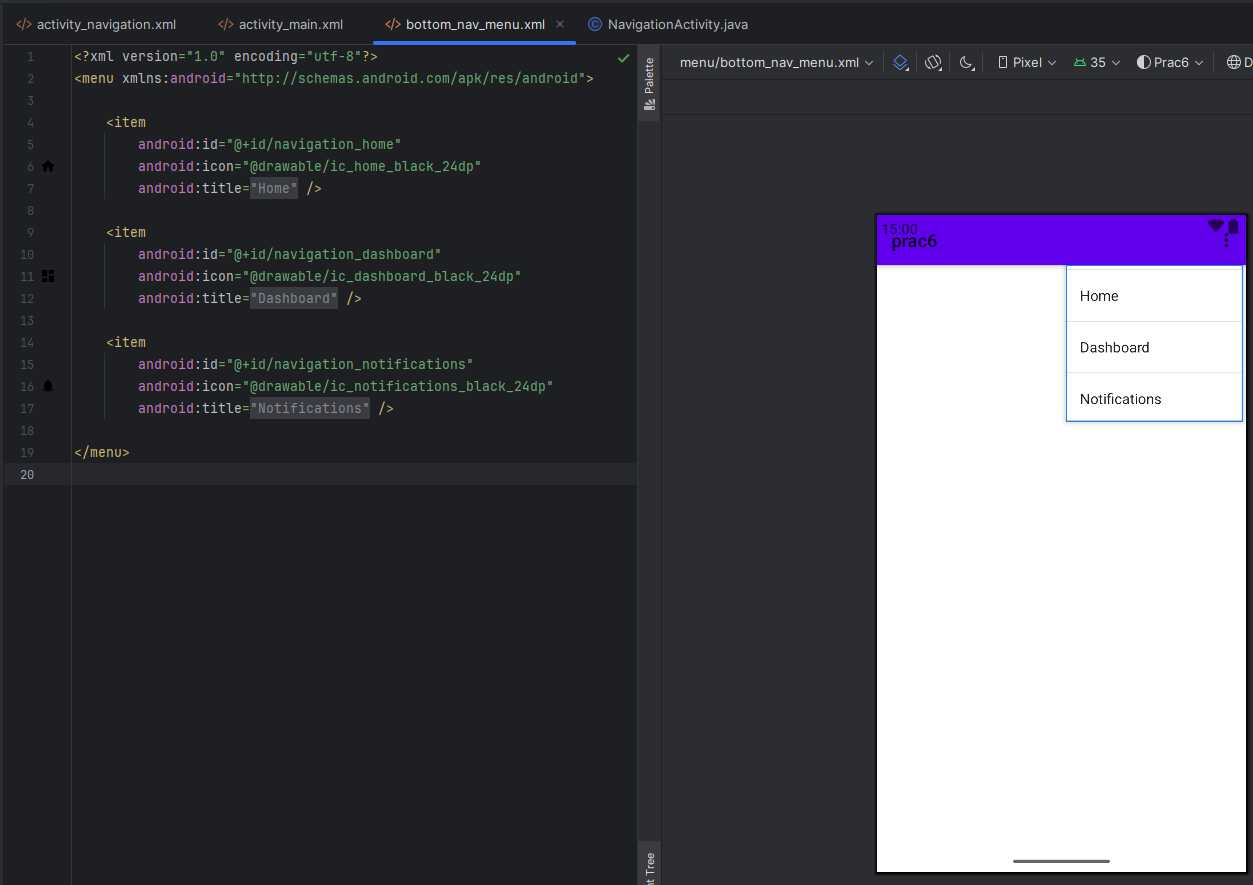
После добавления BottomNavigationView в активность необходимо наполнить его элементами (экранами, между которыми будет осуществляться перемещение). Для этого создается XML-файл в папке res/menu (например, bottom\_nav\_menu.xml) и добавляются в него пункты меню. Если папка menu отсутствует в проекте, то сначала ее необходимо добавить.

Для создания новой папки нажимаем правой кнопкой мыши на "res"→ "New" → "Android Resource Directory" (рисунок 1.1.3). Вводим название папки, выбираем тип ресурса из выпадающего списка (рисунок 1.1.4).

  
Рисунок 1.1.3 – Создание новой папки в проекте

  
Рисунок 1.1.4 – Настройка создаваемой папки

После создания XML-файла его можно наполнить пунктами меню

  
Рисунок 1.1.5 - Графический и кодовый вид нижнего меню

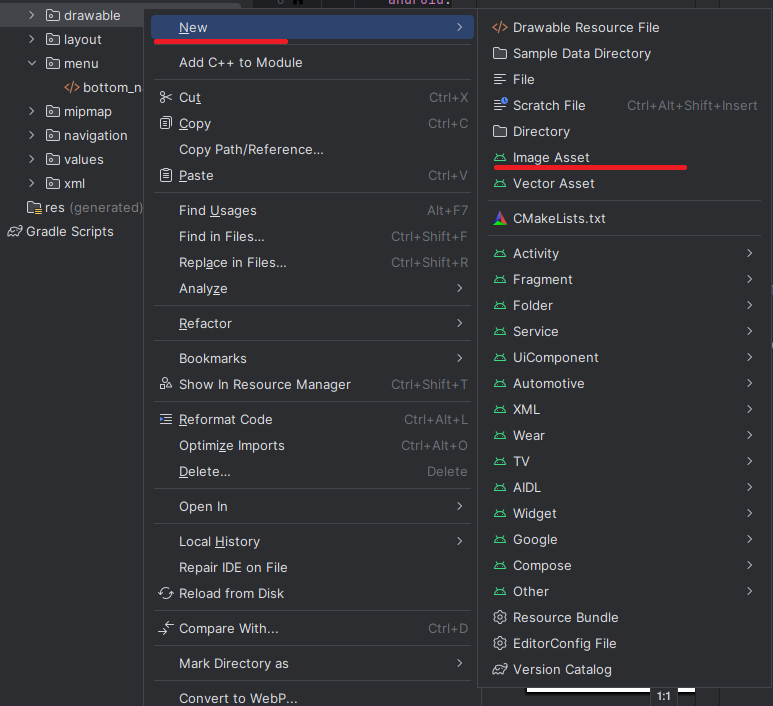
Тег <menu> является корневым узлом файла и определяет меню, состоящее из одного или нескольких элементов <item> и <group>.

Элемент <item> представляет объект MenuItem, которой является одним из элементов меню. Этот элемент может содержать внутренний подэлемент <menu>, с помощью которого создается подменю.

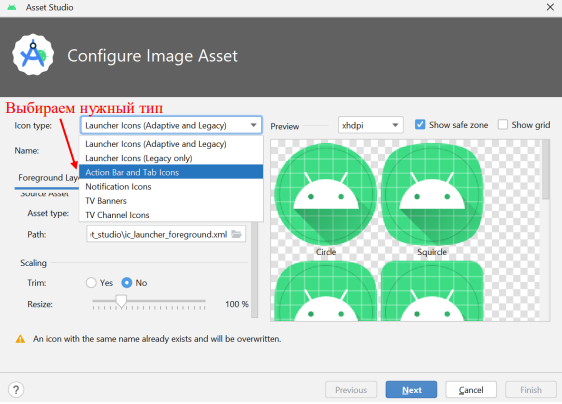
Элемент <item> включает следующие атрибуты, которые определяют его внешний вид и поведение:

* android:id: уникальный id элемента меню, который позволяет его опознать при выборе пользователем и найти через поиск ресурса по id;
* android:icon: ссылка на ресурс drawable, который задает изображение для элемента (android:icon="@drawable/ic\_help");
* android:title: ссылка на ресурс строки, содержащий заголовок элемента. По умолчанию имеет значение "Settings";

Теперь пользователь может создавать столько элементов, сколько захочет, в bottom\_nav\_menu.xml файле. Пользователю также необходимо создать значок для отображения каждого из этих элементов. Чтобы создать значок, нажмите на "drawable"→ "New" → "Image Asset" (рисунок 1.1.6).

  
Рисунок 1.1.6 – Процесс создания новой иконки

Далее идёт выбор нужного типа иконки (рисунок 1.1.7).

  
Рисунок 1.1.7 – Выбор необходимого типа иконки

В открывшемся окне пользователь может назвать значок как угодно, но он не должен содержать ни одной заглавной буквы. Пользователь может выбрать нужный значок, выполнив поиск по нему, а когда пользователь закончит, то необходимо нажать "Next"→ "Finish" (рисунки 1.1.8-1.1.10).

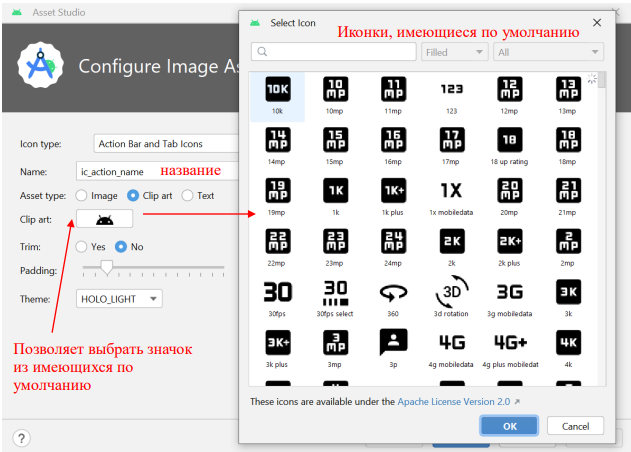
  
Рисунок 1.1.8 – Часть 1 настройки иконки

  
Рисунок 1.1.9 - Часть 2 настройки иконки

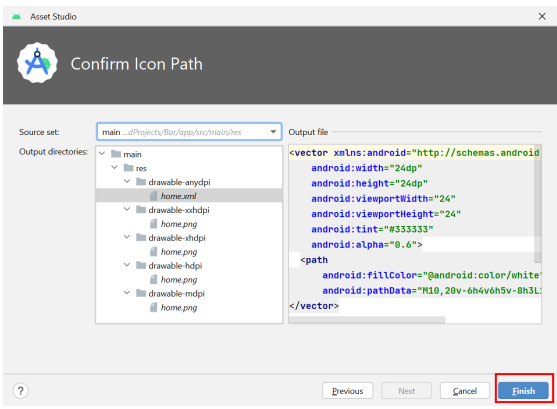
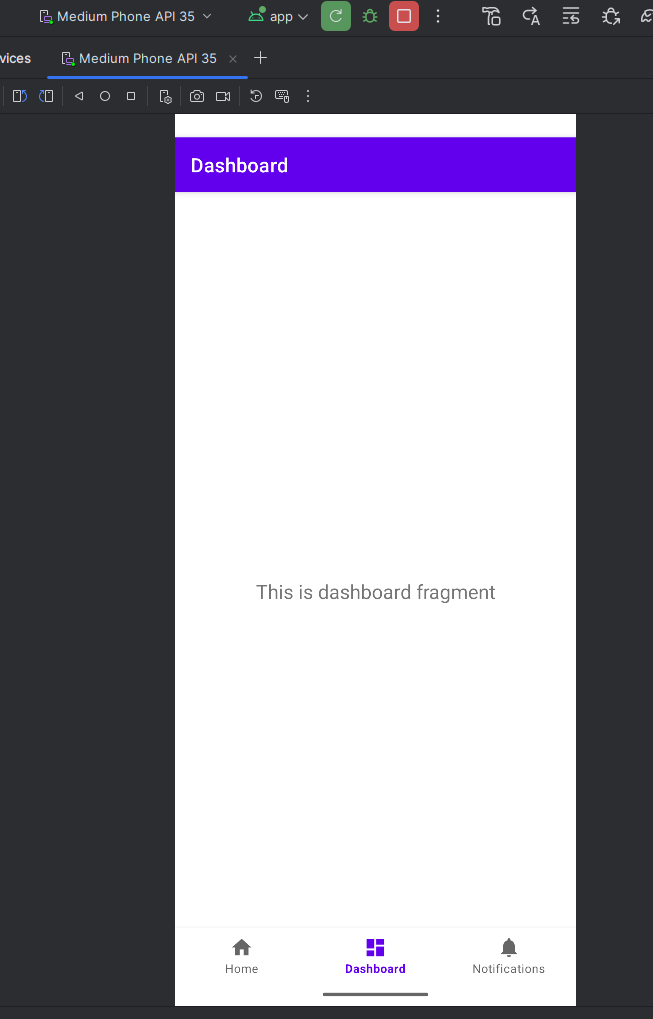
  
Рисунок 1.1.10 - Часть 3 настройки иконки

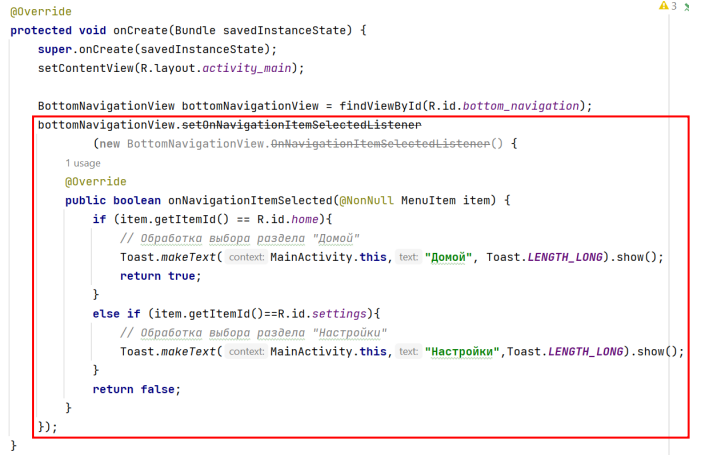
Image Asset поможет создать следующие типы значков:

* иконки лаунчера;
* значки панели действий и вкладок;
* значки уведомлений.

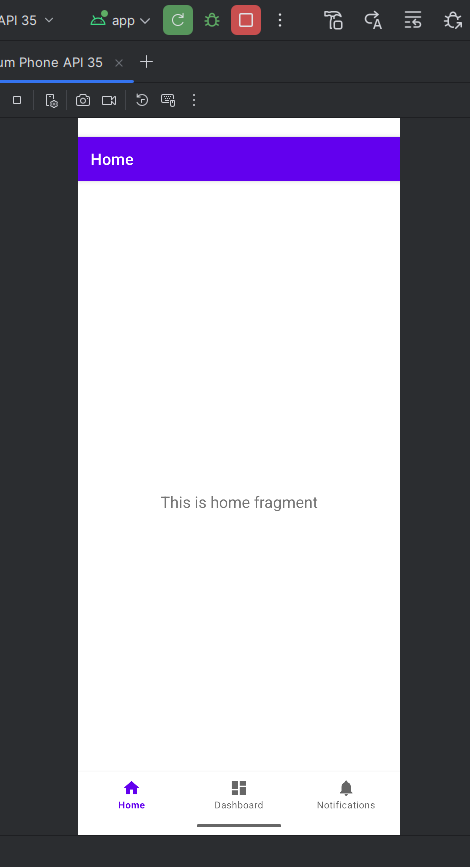
Если запустить сборку проекта, то можно получится отображение меню в нижней части экрана (рисунок 1.1.11).

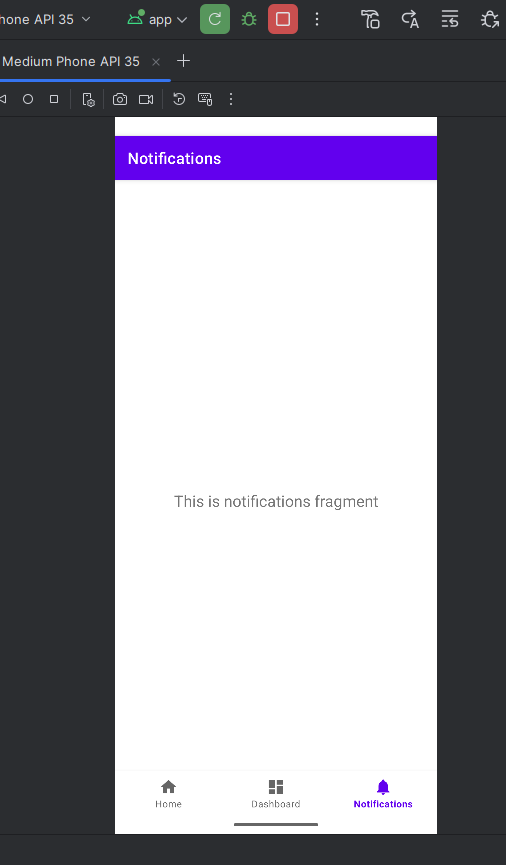
  
Рисунок 1.1.11 – Отображение нижнего меню в итоговом приложении

Однако только отображение не подходит для работы с BottomBar. Необходимо, чтобы при нажатии на элемент, происходили некоторые действия. Для этого необходимо установить слушатель нажатий в классе активности (рисунок 1.1.12).

  
Рисунок 1.1.12 – Реализация логики обработчика событий нижнего меню

Чтобы понять, какой пункт меню выбран, вначале получаем его идентификатор через item.getItemId(). Затем пробегаемся в конструкции if…else и выбираем нужный вариант и в зависимости от выбора производим определенные действия (рисунки 1.1.13-1.1.14).

  
Рисунок 1.1.13 – Часть 1 отображения результата возможных событий

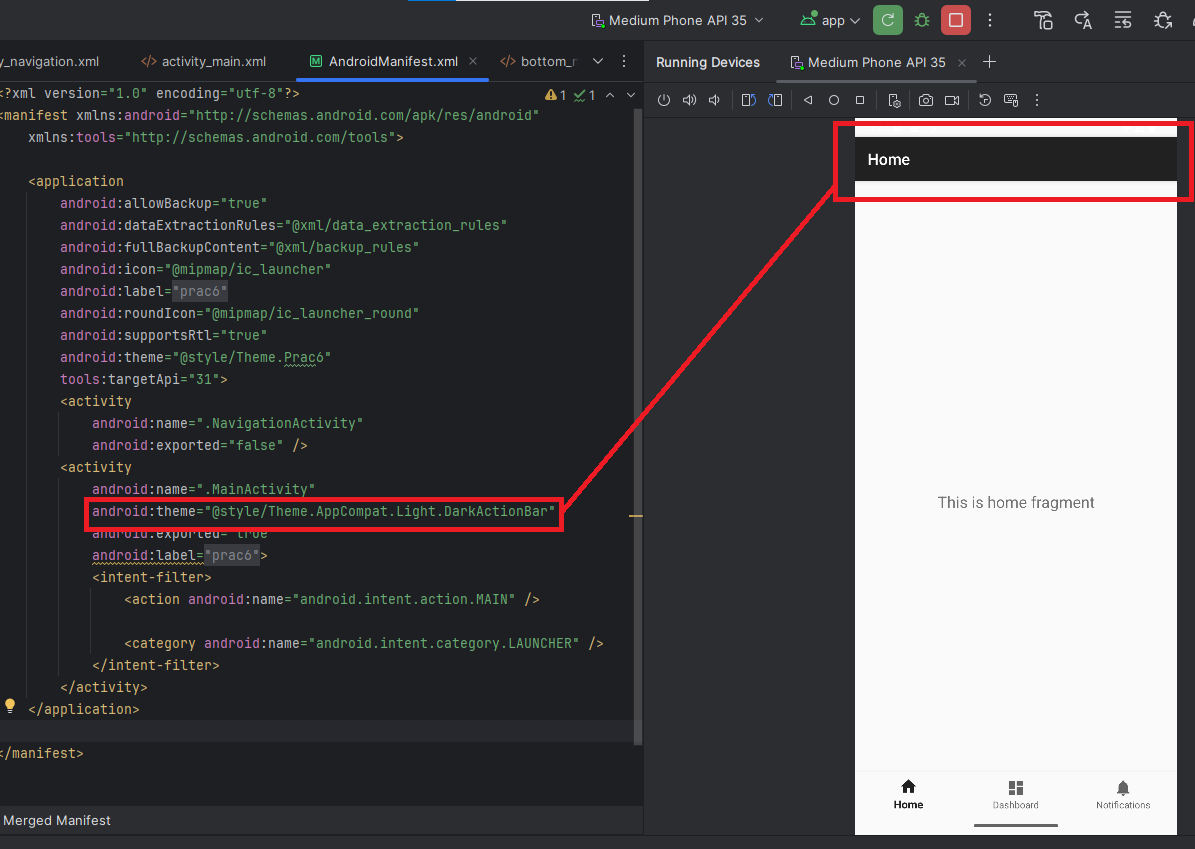
  
Рисунок 1.1.14 – Часть 2 отображения результата возможных событий

На рисунках 1.1.13 и 1.1.14 показан результат обработки нажатия пользователем на иконку “Notifications”.

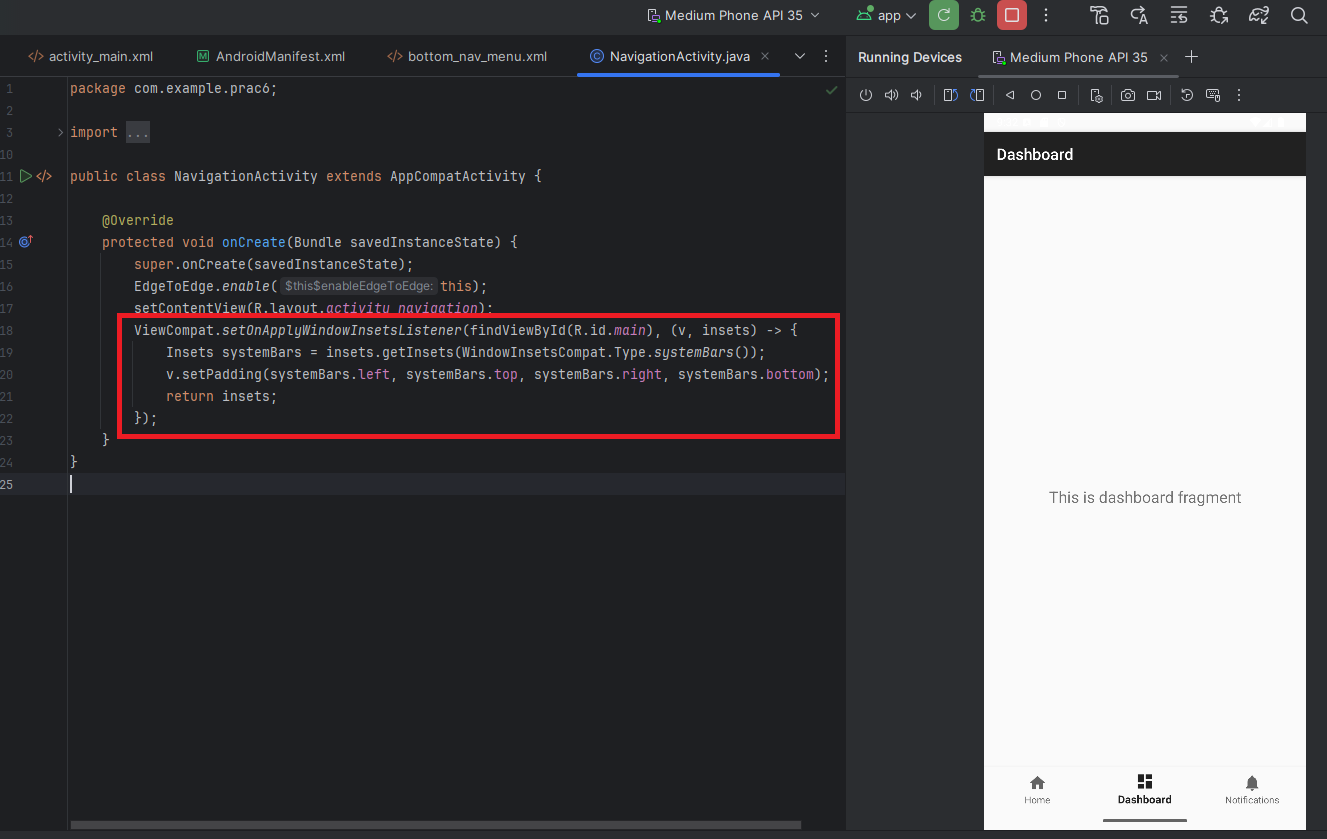
## ActionBar

ActionBar в Android представляет собой верхнюю панель приложения, которая обеспечивает удобный доступ к наиболее важным функциям приложения, а также поддерживает навигацию. Создание и использование ActionBar в связке с нижним и боковым меню может значительно улучшить пользовательский интерфейс и удобство навигации в приложении, например показывая текущий экран, на котором находится пользователь.

ActionBar уже встроен в стандартные темы Activity начиная с API Level 11 (Honeycomb). Для его использования необходимо убедиться, что выбранная Activity наследуется от AppCompatActivity, и используется одну из тем Theme.AppCompat. ActionBar будет отображаться вверху страницы (рисунок 1.2.1).

  
Рисунок 1.2.1 - Графический и кодовый вид ActionBar

Однако такой ActionBar не несет никакой пользы. Для дальнейшей работы его необходимо настроить. В активности можно настроить ActionBar на отображение какого-либо строкового значения (рисунок 1.2.2).

  
Рисунок 1.2.2 – Реализация логики обработки событий для ActionBoard

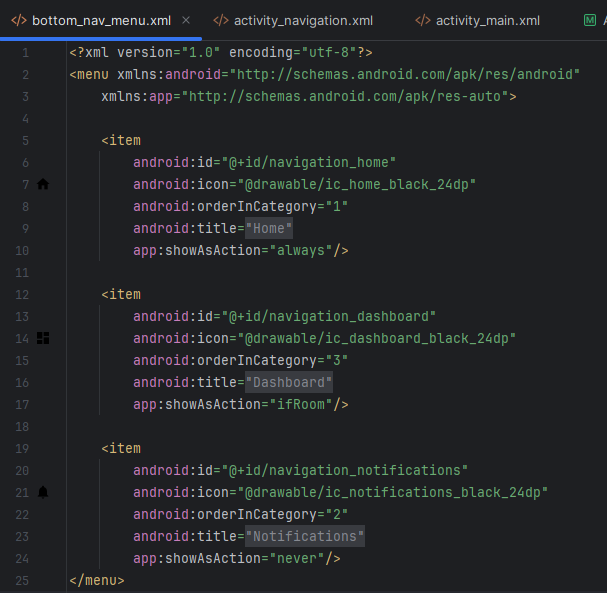
Однако этого тоже может быть недостаточно, так как нам может понадобиться быстрой доступ к некоторым элементам.

Используем ранее созданный файл bottom\_nav\_menu.xml, только немного его изменим, добавив элементам меню некоторые параметры для наглядности (рисунок 1.2.3).

* android:orderInCategory: значение этого атрибута определяет положение элемента в ActionBar. Есть два способа определить положение различных пунктов меню. Первый - предоставить одинаковое значение этого атрибута для всех элементов, и позиция будет определена в том же порядке, в каком они объявлены в коде. Второй способ - предоставить разные числовые значения для всех элементов, и тогда элементы будут располагаться в соответствии с порядком возрастания значения этого атрибута;
* app:showAsAction: этот атрибут определяет, как элемент будет присутствовать на панели действий.

На выбор для значения поля app:showAsAction предлагается четыре возможных флага:

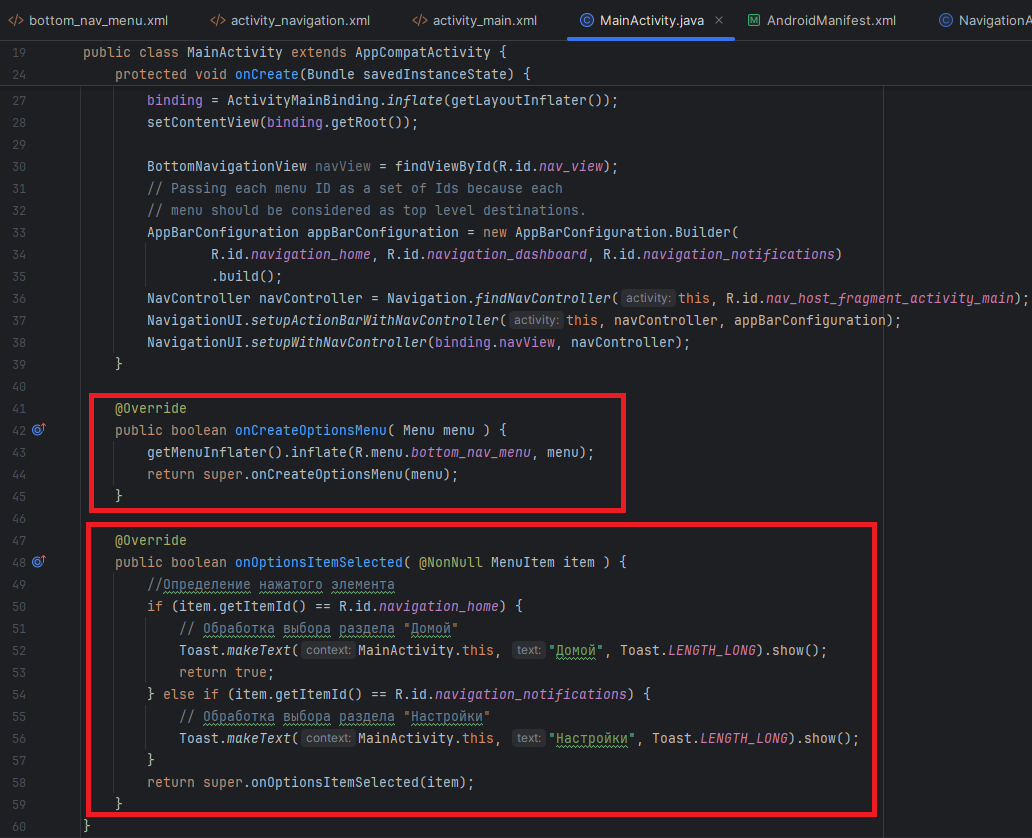
* always: постоянно отображать элемент на панели действий;
* ifRoom: сохранить элемент, если есть свободное место;
* never: с этим флагом элемент не будет отображаться в виде значка в ActionBar, но будет присутствовать в меню переполнения;
* withText: чтобы представить элемент одновременно в виде значка и заголовка, можно дополнить этот флаг флагом always или ifRoom (always|withText or ifRoom|withText).

  
Рисунок 1.2.3 – Изменённый файл разметки bottom\_nav\_menu.xml

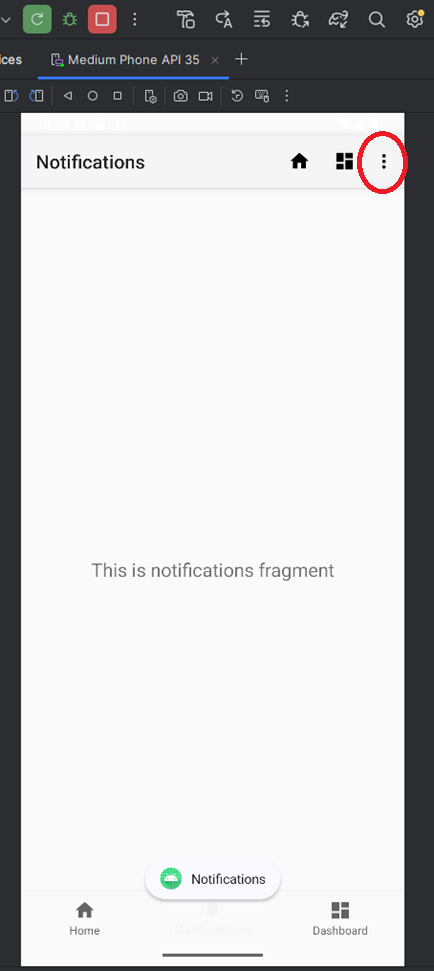
На рисунке 1.2.3 определено меню с тремя элементами, но само определение элементов в файле еще не создает меню. Это всего лишь описание. Чтобы вывести его на экран, надо использовать его в классе Activity. Для этого надо переопределить метод onCreateOptionsMenu.

Метод getMenuInflater возвращает объект MenuInflater, у которого вызывается метод inflate(). Этот метод в качестве первого параметра принимает ресурс, представляющий наше описание меню в xml, и наполняет им объект menu, переданный в качестве второго параметра.

Теперь, если мы нажмем на любой из пунктов меню, то ничего не произойдет. Чтобы привязать к меню действия, нам надо переопределить в классе activity метод onOptionsItemSelected (рисунок 1.2.4).

  
Рисунок 1.2.4 – Переопределение необходимых методов для ActionBar

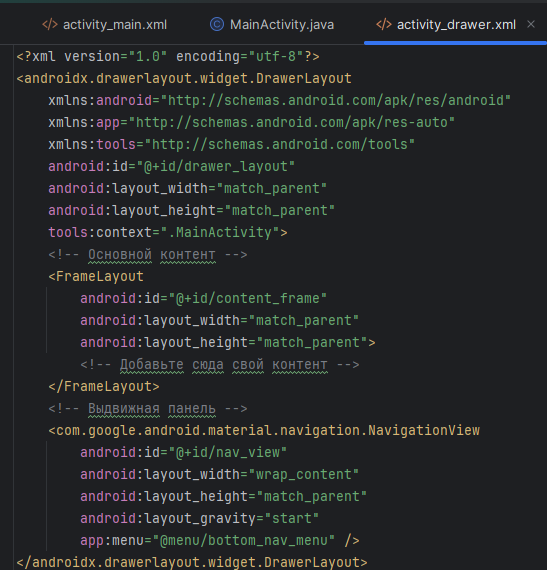
И при нажатии на троеточие в правом верхнем углу экрана, а после нажатия на “Notifications” появится сообщение “Notifications” (рисунок 1.2.5).

  
Рисунок 1.2.5 – Итоговый вид ActionBar-а в рабочем приложении

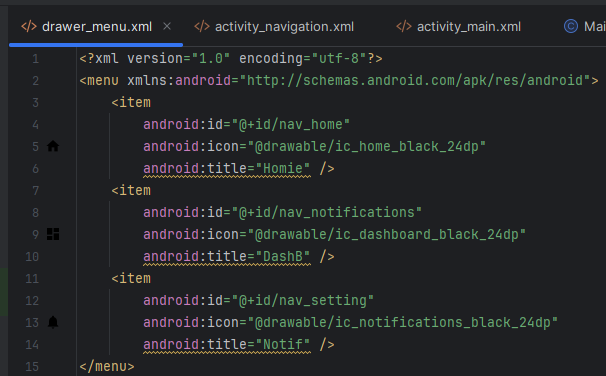
## Navigation Drawer

Создание выдвижной панели (Drawer) в Android обычно реализуется с помощью DrawerLayout и NavigationView. DrawerLayout используется как корневой контейнер интерфейса, который позволяет разместить основное содержимое приложения и выдвижную панель. NavigationView представляет собой панель навигации, которая отображается в DrawerLayout и содержит элементы меню для навигации по различным разделам приложения.

В файле макета активности необходимо добавить DrawerLayout как корневой элемент. Внутри DrawerLayout разместим основной контент и NavigationView для выдвижной панели (рисунок 1.3.1).

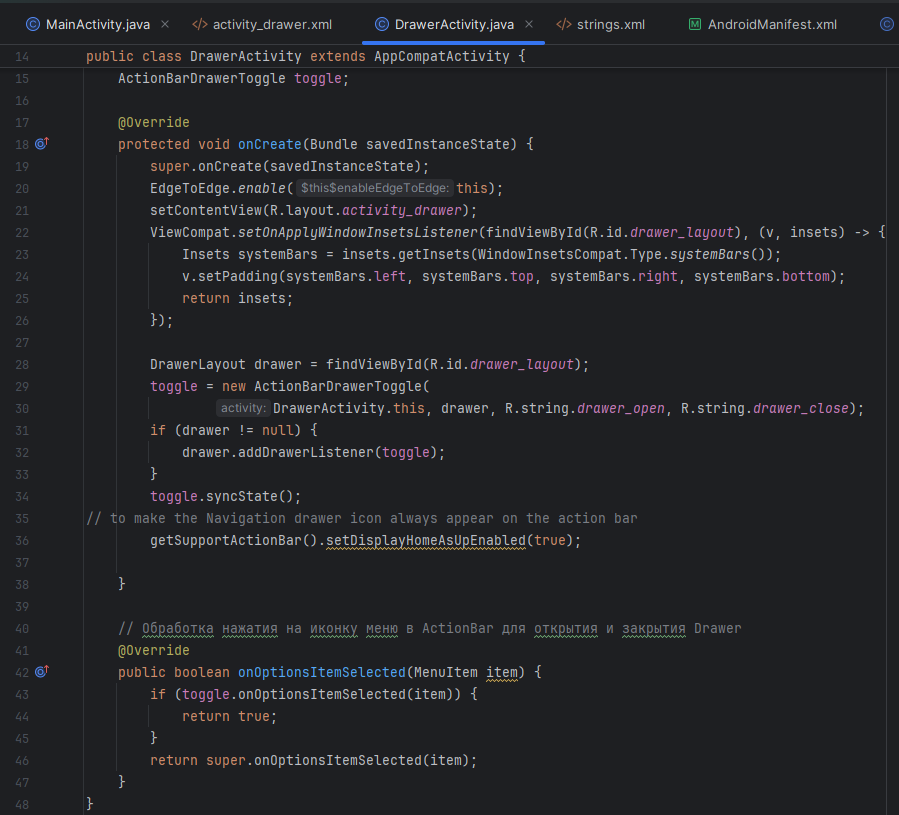
  
Рисунок 1.3.1 – Кодовый вид файла разметки activity\_drawer.xml

Создаётся XML-файл в папке res/menu (например, drawer\_menu.xml) и добавьте пункты меню для вашей выдвижной панели (рисунок 1.3.2).

  
Рисунок 1.3.2 – Описание элементов файла drawer\_menu.xml

Затем по аналогии с BottomBar создать слушатель нажатий для каждого элемента меню.

Для интеграции ActionBar с DrawerLayout используется ActionBarDrawerToggle, который добавит иконку меню для открытия и закрытия Drawer и обеспечит анимацию иконки (рисунок 1.3.3).

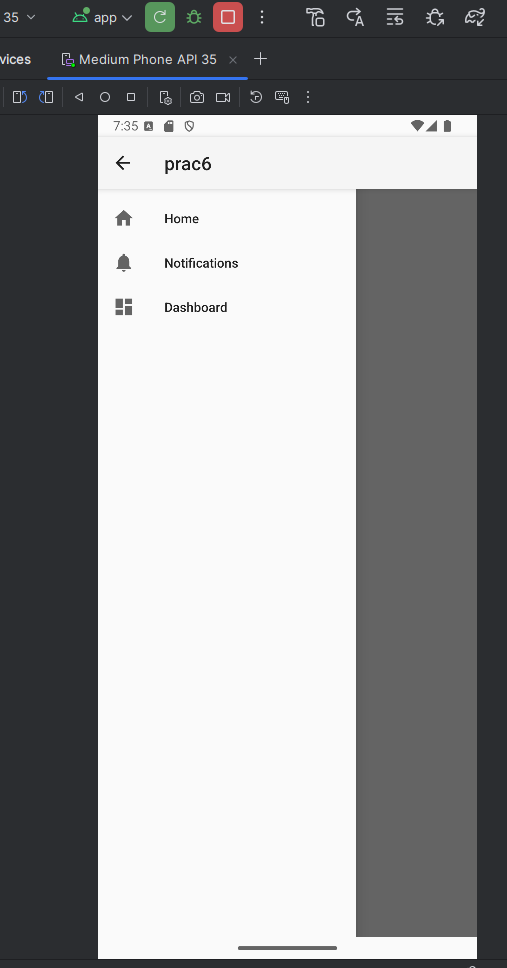
  
Рисунок 1.3.3 – Реализация логики Navigation Drawer

Также на рисунке 1.3.3 был добавлен слушатель для обработки нажатия на элементы (onOptionsItemSelected).

В итоге будет отображаться всплывающее меню с выбранными строковыми значениями (рисунки 1.3.4-1.3.5).

  
Рисунок 1.3.4 – Часть 1 отображения всплывающего меню

После запуска приложения пользователь нажмёт на иконку всплывающего меню, находящуюся в левом верхнем углу, и увидит содержимое всплывающего меню (рисунок 1.3.5).

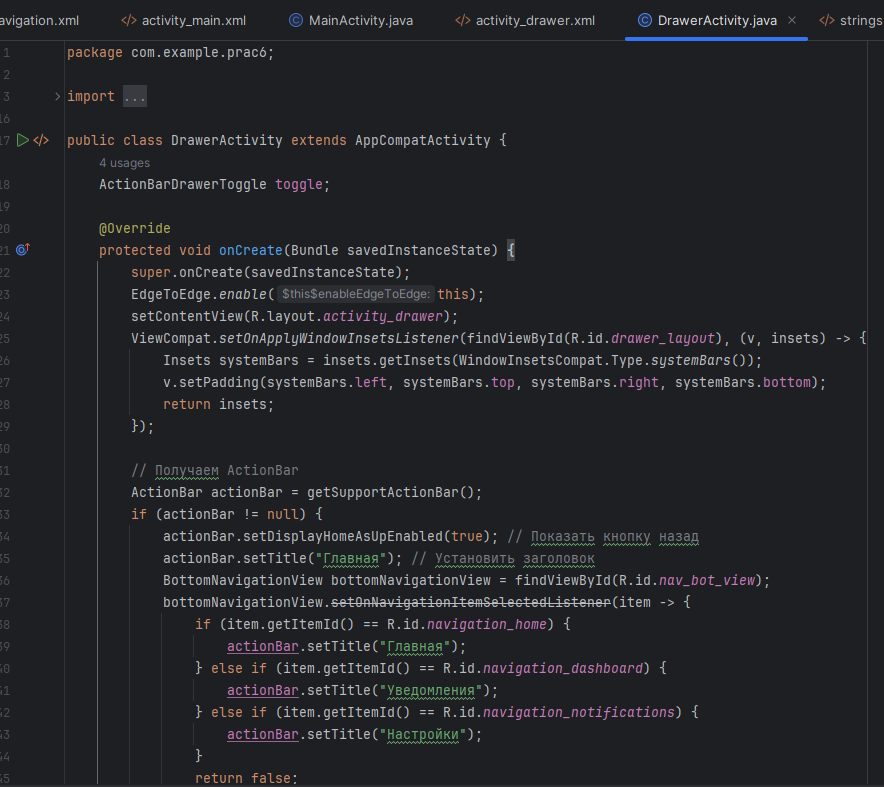
  
Рисунок 1.3.5 - Часть 2 отображения всплывающего меню

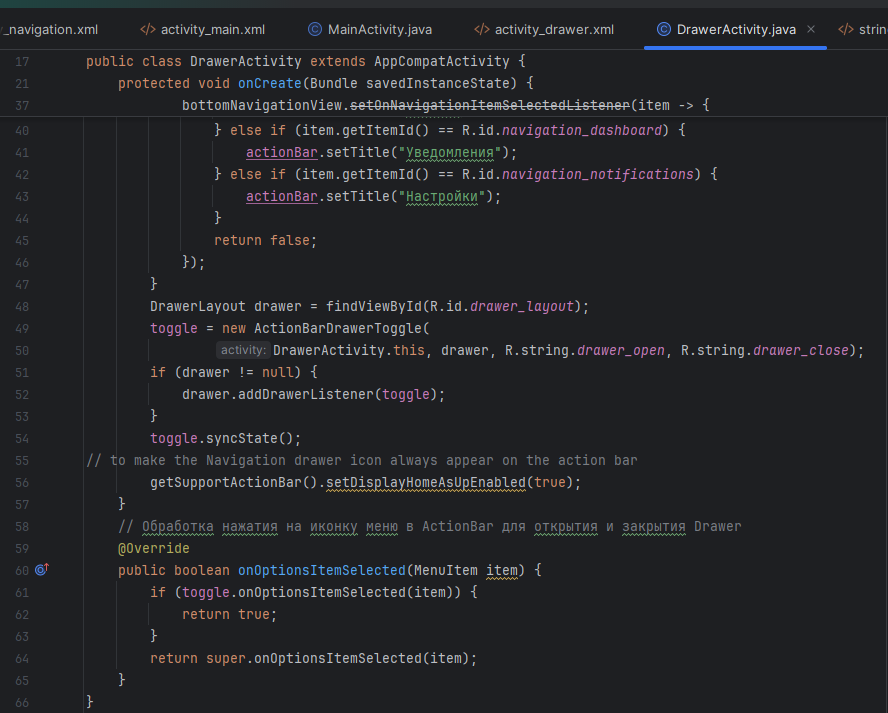
Чтобы использовать ActionBarDrawerToggle, нужно создать экземпляр при помощи конструктора со следующими аргументами:

* Activity, в котором размещается боковая панель навигации;
* DrawerLayout – drawable ресурс, используемый в качестве индикатора панели. Стандартный навигационный значок панели доступен в Download the Action Bar Icon Pack;
* Строковый ресурс для обозначения открытой панели (для специальных возможностей);
* Строковый ресурс для обозначения закрытой панели (для специальных возможностей).

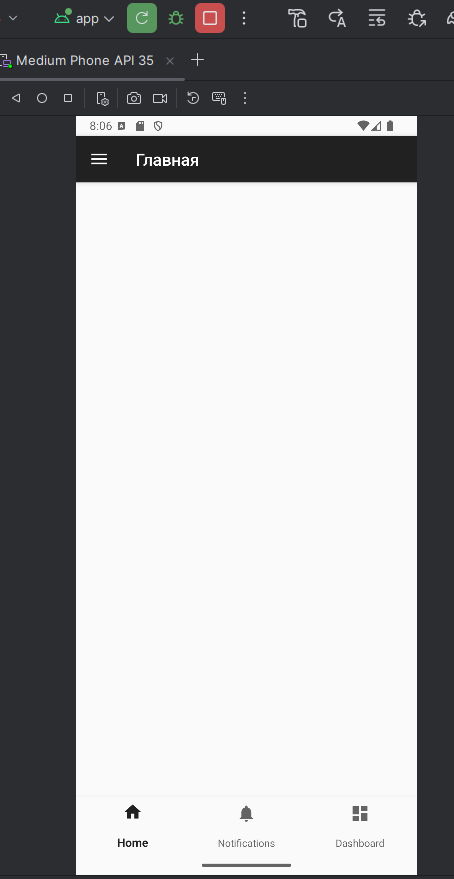
Теперь, в зависимости от использования класса ActionBarDrawerToggle в списке панели навигации, необходимо вызвать ActionBarDrawerToggle в нескольких местах жизненного цикла activity.

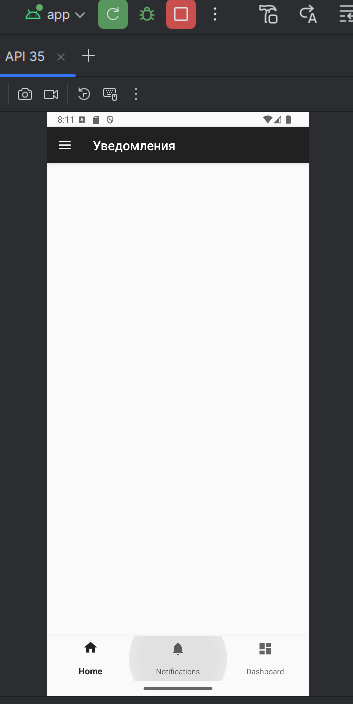
Кроме этого, ActionBar можно использовать вместе с BottomNavigationView для создания единой системы навигации. В этом случае ActionBar обычно служит для отображения контекстной информации (например, заголовка текущей страницы), а BottomNavigationView для навигации между основными разделами приложения (рисунки 1.3.6-1.3.7).

  
Рисунок 1.3.6 – Часть 1 реализации логики BottomNavigationView

  
Рисунок 1.3.7 - Часть 2 реализации логики BottomNavigationView

У пользователя появится возможность переходить по нижней панели на различные фрагменты (рисунки 1.3.8-1.3.9).

  
Рисунок 1.3.8 – Часть 1 отображения итогового вида приложения

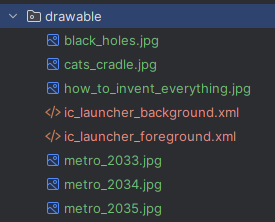
  
Рисунок 1.3.9 – Часть 2 отображения итогового вида приложения

Это позволяет пользователю быстро переключаться между разделами приложения с помощью нижнего меню, в то время как верхняя панель (ActionBar) отображает контекстную информацию о выбранном разделе или предоставляет дополнительные опции действий, связанных с текущим экраном. Использование ActionBar в сочетании с боковым и нижним меню позволяет создать гибкую и интуитивно понятную систему навигации для пользователей, обеспечивая легкий доступ ко всем важным функциями в приложении.

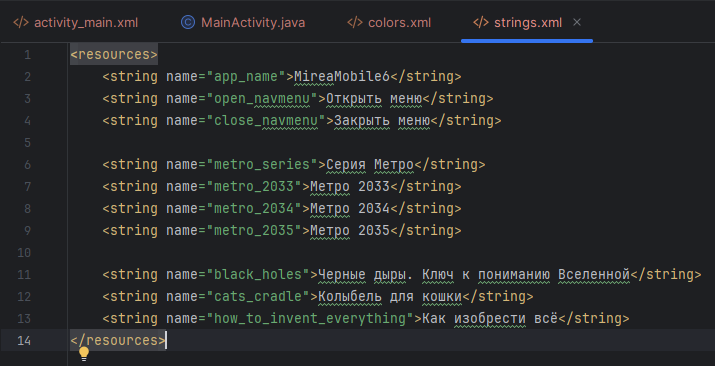
# ПРАКТИЧЕСКАЯ ЧАСТЬ

## Подготовка ресурсов

Сперва поместим шесть изображений в res/drawable (по три на каждую из двух Activity), которые будем использовать для наполнения фрагментов. Итоговую структуру res/drawable можно увидеть на рисунке 2.1.1.

  
Рисунок 2.1.1 – Структура res/drawable

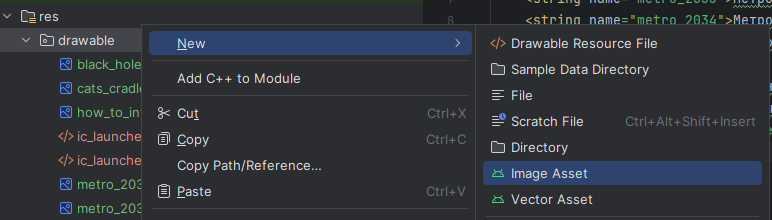
Затем определим необходимые строковые ресурсы в файле strings.xml (рисунок 2.1.2).

  
Рисунок 2.1.2 – Итоговый вид файла ресурсов string.xml

Файл strings.xml содержит:

* строки, использующиеся для установки описаний состояний бокового меню, что важно для доступности и улучшения пользовательского опыта,
* строку с заголовком MainActivity («metro\_series»),
* строки с названиями разделов MainActivity (книги),
* строки с названиями разделов SecondActivity (книги).

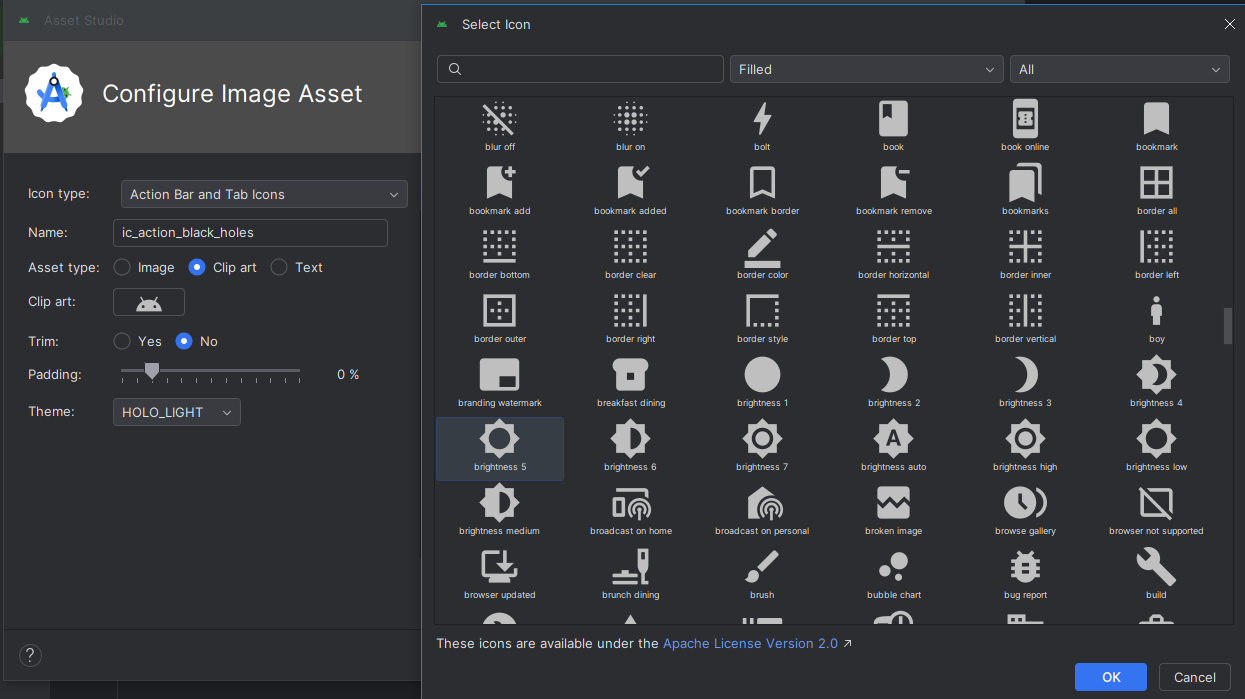
Далее добавим иконки, которые будут отображаться в разделах навигации. Для этого нажмём на "drawable"→ "New" → "Image Asset" (рисунок 2.1.3).

  
Рисунок 2.1.3 – Запуск Asset Studio в Android Studio

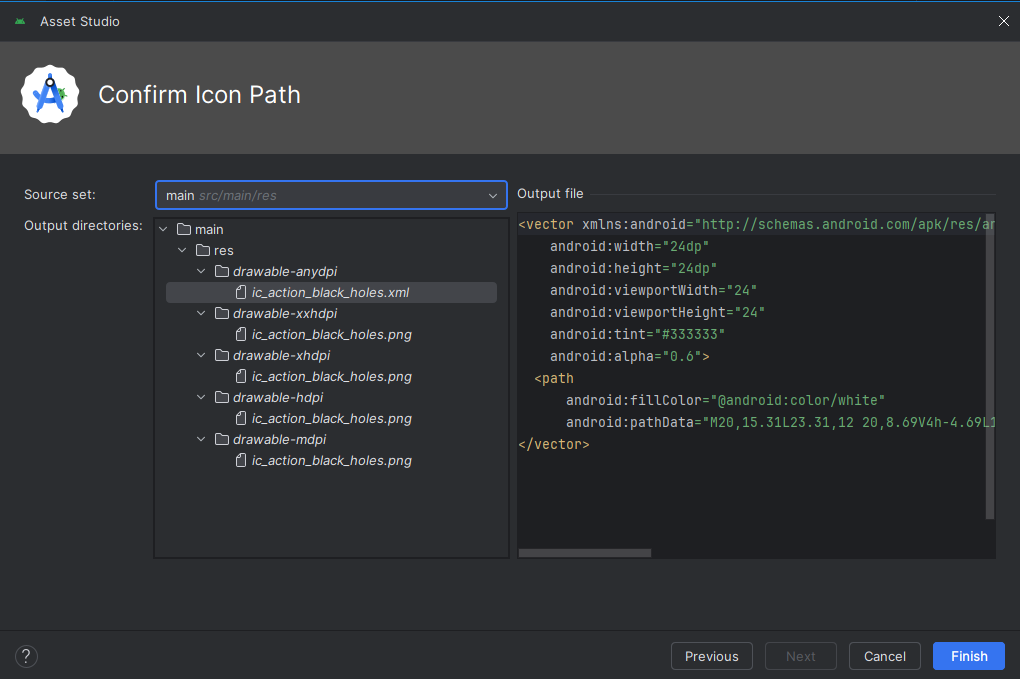
В разделе Configure Image Asset мы выбираем «Action Bar and Tab Icons» в поле Icon Type «Clip art» в качестве Asset Type. Затем задаём название для иконки действия и нажимаем на область рядом с пунктом Clip art.

После этого открывается окно с коллекцией Google Material Icons, где мы выбираем подходящую иконку. Подтверждаем выбор, нажав ОК, закрываем окно Select Icon и идём дальше, нажимая Next.

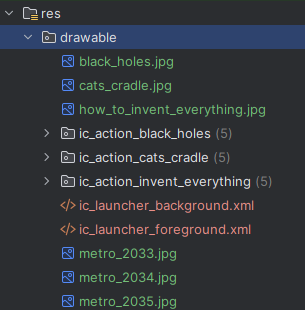
Указанные настройки продемонстрированы на рисунке 2.1.4.

  
Рисунок 2.1.4 – Создание иконки с помощью Asset Studio (часть 1)

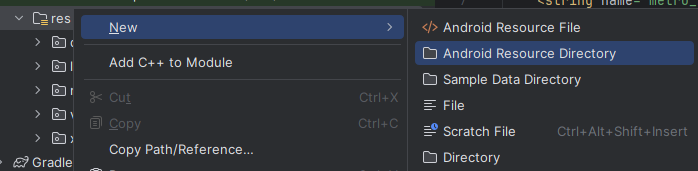
В разделе Confirm Icon Path смотрим пути к создаваемым файлам и завершаем процесс создания иконки, нажав кнопку Finish (рисунок 2.1.6).

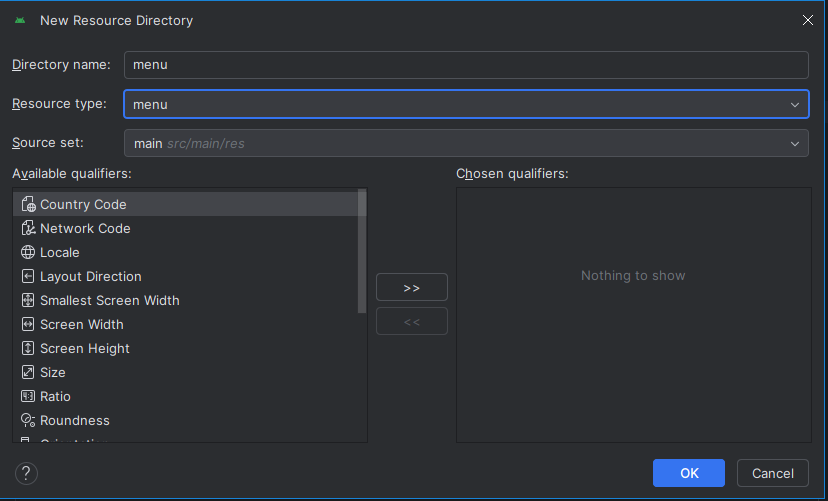
  
Рисунок 2.1.5 – Создание иконки с помощью Asset Studio (часть 2)

По аналогии создадим ещё две иконки. Итоговая структура ресурсов продемонстрирована на рисунке 2.1.6.

  
Рисунок 2.1.6 – Структура ресурсов drawable

Для дальнейшей работы с описанием меню приложения создадим в папке res новую директорию с названием menu и типом ресурсов menu (рисунки 2.1.7-2.1.8).

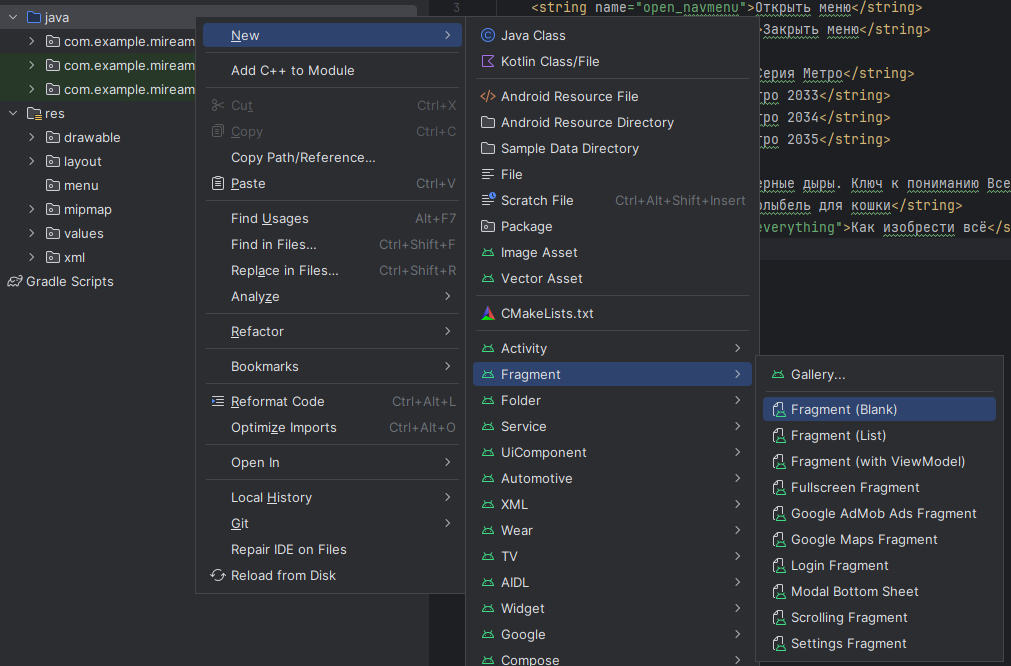
  
Рисунок 2.1.7 – Создание новой директории ресурсов

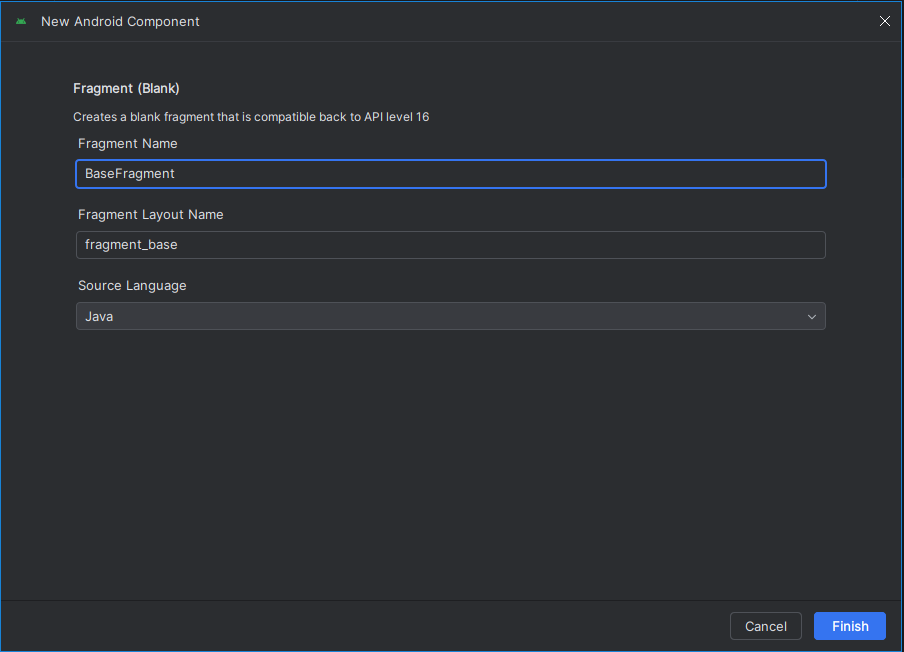
  
Рисунок 2.1.8 – Создание директории ресурсов типа menu

## Реализация фрагментов для демонстрации изображений

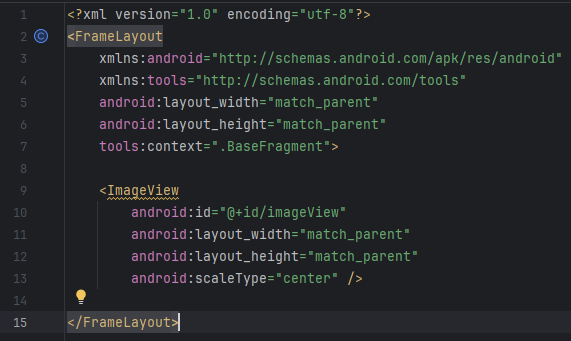
Реализуем изменяемое содержимое наших Activity с помощью фрагментов. Каждый фрагмент будет иметь единообразную структуру и выполнять одну основную функцию — отображение изображения.

Сначала мы создадим новый пустой фрагмент с именем BaseFragment и файлом разметки fragment\_base.xml. Этот фрагмент будет служить шаблоном для всех остальных фрагментов. Android Studio позволяет удобным способом с помощью контекстных меню создать новый фрагмент, для этого нажимаем правой кнопкой мыши на "java"→ "New" → "Fragment" → "Fragment (Blank)" (рисунки 2.2.1-2.2.2).

  
Рисунок 2.2.1 – Создание базового фрагмента BaseFragment (часть 1)

  
Рисунок 2.2.2 – Создание базового фрагмента BaseFragment (часть 2)

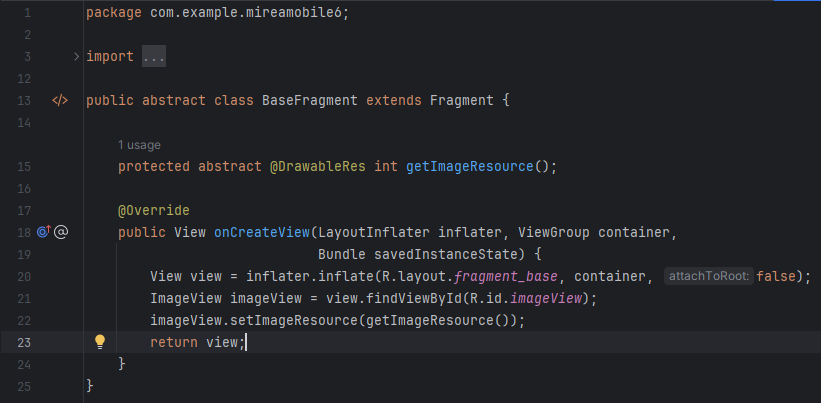
В файле разметки fragment\_base.xml определим в качестве корневого элемента контейнер компоновки FrameLayout, а в него поместим компонент ImageView с идентификатором imageView. Таким образом, получим простейший макет, по которому будут строиться наши фрагменты (рисунок 2.2.3).

  
Рисунок 2.2.3 – Разметка базового фрагмента для демонстрации изображения

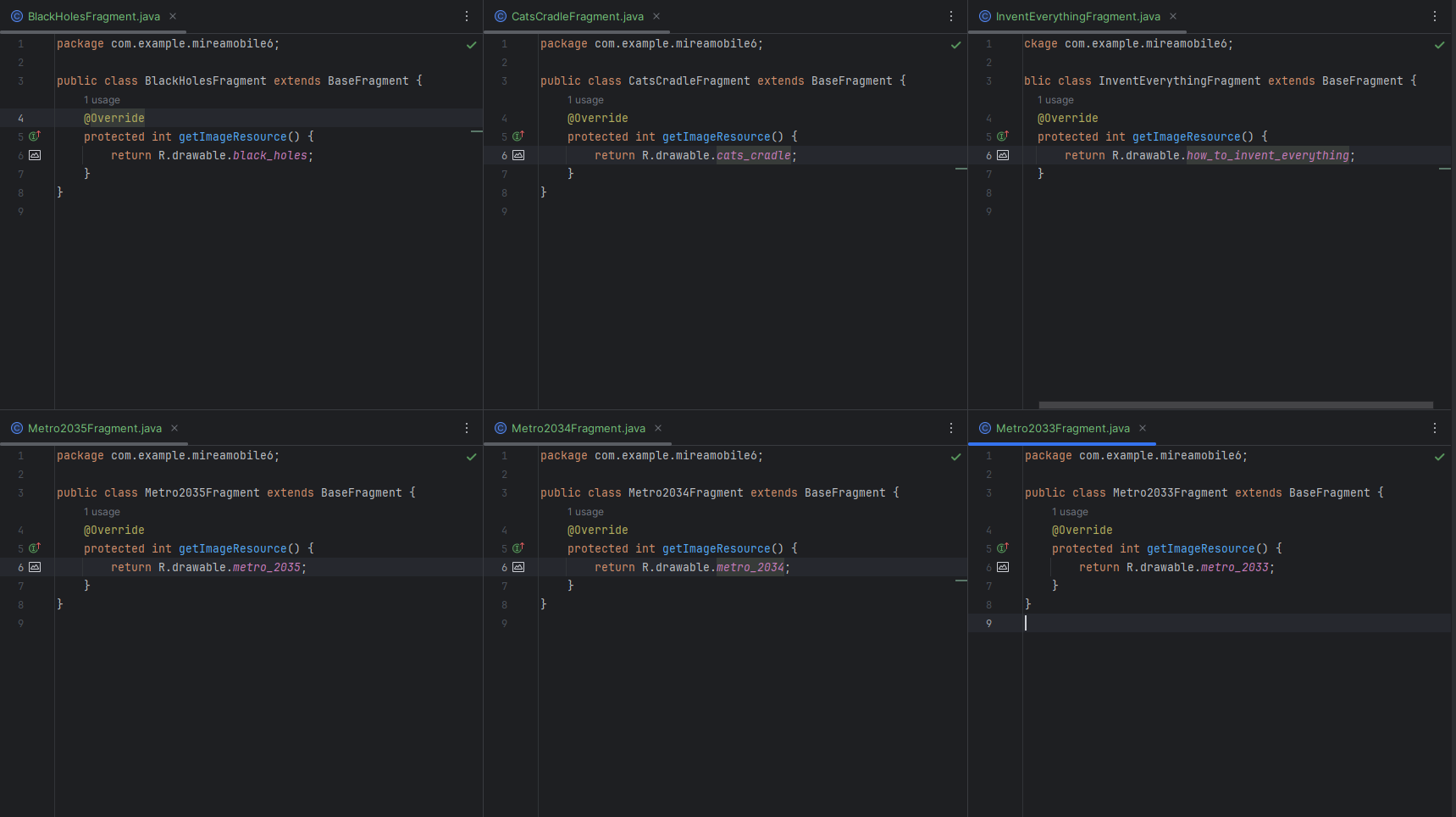
Определим абстрактный класс BaseFragment, выступающий базой для специализированных фрагментов с изображениями. В классе определяется абстрактный метод getImageResource(), возвращающий идентификатор ресурса изображения – его реализация потребуется в дочерних классах.

В методе onCreateView() происходит загрузка макета fragment\_base.xml через inflate(). Затем находится элемент ImageView по идентификатору, и ему устанавливается изображение методом setImageResource() с использованием ресурса из getImageResource().

Такой подход создаёт универсальную основу: общая логика отображения сохраняется в базовом классе, а конкретные изображения определяются в наследниках (рисунок 2.2.4).

  
Рисунок 2.2.4 – Определение абстрактного класса базового фрагмента

Далее создадим конкретные реализации фрагментов для отображения различных изображений в зависимости от разделов нашего приложения. Каждый фрагмент наследуется от базового класса BaseFragment и переопределяет метод getImageResource(), возвращая соответствующий идентификатор ресурса изображения из папки res/drawable (рисунок 2.2.5).

  
Рисунок 2.2.5 – Создание фрагментов для демонстрации конкретных изображений

## Реализация Activity с боковой панелью

# ЗАКЛЮЧЕНИЕ

В ходе данной практической работы были получены знания связанные с NavigationBar, ActionBar и BottomBar. Полученные знания были закреплены путём выполнения практического задания.