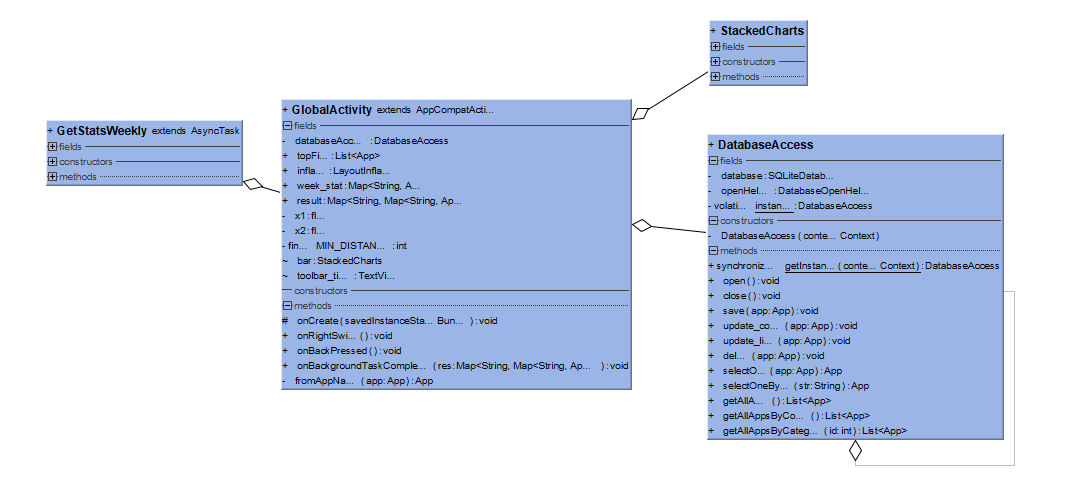
**GlobalActivity (дейятельность окна глобальной статистики)**



private DatabaseAccess databaseAccess;

переменная для работы с бд

public List<App> topFive;

топ 5 частых приложений за неделю

public LayoutInflater inflater;

[LayoutInflater](http://developer.android.com/reference/android/view/LayoutInflater.html) – это класс, который умеет из содержимого layout-файла создать View-элемент. Метод который это делает называется **inflate**. (для вывода топ 5 уведомлений или топ 5 приложений за неделю)

public Map<String,App> week\_stat;

переменная мапа хранит в себе стату приложений за неделю (Название, Приложение)

public Map<String,Map<String,App>> result;

Переменная хранит в себе мапу в виде( Дата, мапа (Название, приложение)ж

private float x1, x2;  
private final int MIN\_DISTANCE = 150;

Переменные которые используются для отслеживания изменения координат( для свайпа)

StackedCharts bar;

Переменная для создания Стакедбара (для вывода статы)

@Override  
protected void onCreate(Bundle savedInstanceState)

Создание самого активити, Получение данных о StackedCharts. Вызов Асинхронного класса для получения статистики за неделю, получение сведений из БД о топ 5 приложений (по уведомлениям), заполнения статистики о топ 5 приложениях (по уведомлениям и по времени использованию), а так же

setOnTouchListener

описание работы свайпа.

public void onRightSwipe() {  
 overridePendingTransition(R.anim.*right\_in*, R.anim.*left\_out*);  
 GlobalActivity.this.finish();  
}  
  
@Override  
public void onBackPressed() {  
 overridePendingTransition(R.anim.*right\_in*, R.anim.*left\_out*);  
 GlobalActivity.this.finish();  
  
}

два метода содержащие в себе переопределения выхода (завершения ) активити …То что активити нас закрывается слайдом с права на лево.

onBackgroundTaskCompleted

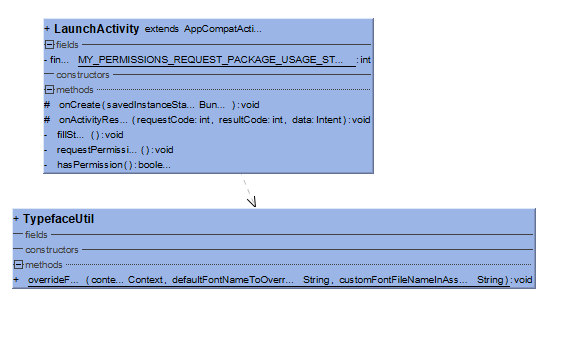
Метод который вызывается по завершению выполенния асинхронного потока (получения статы за неделю).

Данный метод разбирает всю информацию из мапы, сортирует ее и обновлет экран(диаграмму и все списки с инфой, перерасчет времени в Тоолбаре)

fromAppName

Получения сведения о приложении по его имени пакета

**LaunchActivity (дейятельность экрана заставки)**



protected void onCreate(Bundle savedInstanceState) {

TypefaceUtil.*overrideFont*(getApplicationContext(), "SANS\_SERIF", "font.ttf");  
fillStats();

Установление пользовательских шрифтов и проверка разрешения на чтение инфы об использованных приложениях.

private void fillStats() {  
 if (hasPermission()){  
 new Handler().postDelayed(new Runnable() {  
 @Override  
 public void run() {  
 Intent intent = new Intent(LaunchActivity.this, MainActivity.class);  
 startActivity(intent);  
 overridePendingTransition(R.anim.*right\_in*, R.anim.*left\_out*);  
 LaunchActivity.this.finish();  
 }  
 }, 2000);  
 }else{  
 requestPermission();  
 }  
}

Если разрешение получено, запускаем экран с заставкой с задержкой 2сек..

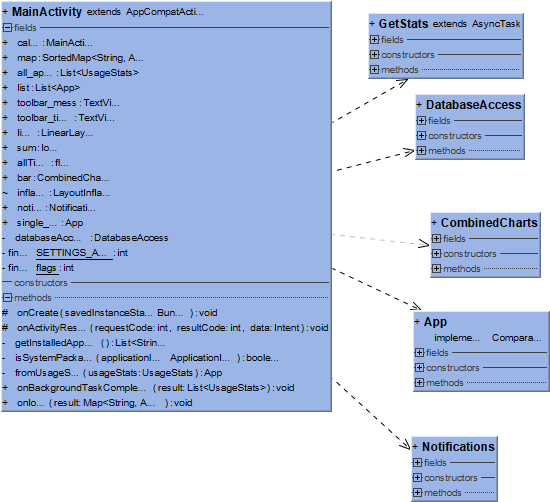
Если нет то вызываем метод для получения разрешения

private void requestPermission() {  
 Toast.*makeText*(this, "Need to request permission", Toast.*LENGTH\_SHORT*).show();  
 startActivityForResult(new Intent(Settings.*ACTION\_USAGE\_ACCESS\_SETTINGS*), *MY\_PERMISSIONS\_REQUEST\_PACKAGE\_USAGE\_STATS*);  
}

Показываем внизу экрана уведомление, что нужно разрешение

И открываем окно для его получения.

**MainActivity (дейятельность главного экрана)**

****

В данном классе описана логика работы главного экрана, все глобальные переменные я описывать не буду …опишу ключевые моменты только.

notific = new Notifications(this);

инициализируем уведомления, теперь как только будет достингнут лимит времени..мы об этом узнаем)

ImageButton setBut = (ImageButton) findViewById(R.id.*optBut*);  
  
setBut.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 Intent intent = new Intent(MainActivity.this, SettingsActivity.class);  
 startActivityForResult(intent, *SETTINGS\_ACT*);  
 }  
});  
  
ImageButton globalBut = (ImageButton) findViewById(R.id.*globalBut*);  
globalBut.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 Intent intent = new Intent(MainActivity.this, GlobalActivity.class);  
 startActivityForResult(intent, *SETTINGS\_ACT*);  
 }  
});

Описание действий кнопок ( глобальная статистика и настройки)

if (Build.VERSION.*SDK\_INT* >= Build.VERSION\_CODES.*O*) {  
 Timer myTimer = new Timer(); // Создаем таймер  
 final Handler uiHandler = new Handler();  
  
 myTimer.schedule(new TimerTask() { // Определяем задачу  
 @Override  
 public void run() {  
 uiHandler.post(new Runnable() {  
 @Override  
 public void run() {  
 new GetStats(caller).execute();  
  
 }  
 });  
 }  
 }, 0L, 300L \* 1000);  
}

Стата за день собирается по таймеру раз в 30 секунд.(вызывается асинхронный класс для сбора инфы)

@Override  
protected void onActivityResult(int requestCode, int resultCode, Intent data) {  
 switch (requestCode) {  
 case *SETTINGS\_ACT*: {  
 this.onload(new App().sort(map));  
 notific.sendAllLimitNotification(allTime);  
 notific.sendAppLimitNotification(map);  
 }  
 break;  
 }  
}

Метод который по завершению активити( Настроек, глобальных,По приложению) , обновляем стату на экране и отправляет сведения в класс уведомлений , для их вывода…если лимит достигнут).

private List<String> getInstalledAppList() {  
 List<ApplicationInfo> infos = getPackageManager().getInstalledApplications(*flags*);  
 List<String> installedApps = new ArrayList<>();  
 for (ApplicationInfo info : infos) {  
 if (!isSystemPackage(info))  
 installedApps.add(info.packageName);  
 }  
 return installedApps;  
}

Метод, для получения приложений установленных на данный момент, чтобы отсеивать удаленные, например 5 минут назад удалил..чтобы не отобрать его в стате.

private boolean isSystemPackage(ApplicationInfo applicationInfo) {  
 return ((applicationInfo.flags & ApplicationInfo.*FLAG\_SYSTEM*) != 0);  
}

Метод, для отсеивания системных приложений.

private App fromUsageStat(UsageStats usageStats) throws IllegalArgumentException {  
 try {  
 ApplicationInfo ai = getPackageManager().getApplicationInfo(usageStats.getPackageName(), 0);  
 return new App(usageStats, getPackageManager().getApplicationIcon(ai), getPackageManager().getApplicationLabel(ai).toString(), usageStats.getTotalTimeInForeground(), ai.category);  
 } catch (PackageManager.NameNotFoundException e) {  
 throw new IllegalArgumentException(e);  
 }  
}

Получение инфы о приложении по его UsageStats

public void onBackgroundTaskCompleted(final List<UsageStats> result) throws IllegalArgumentException {

Метод который автоматом вызывается по завершению асинхронного потока..

Сортирует полученную инфу и обновляет экран.

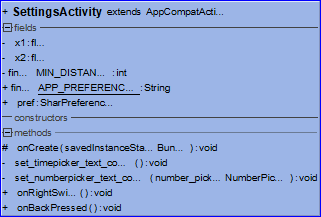
public void onload(final Map<String,App> result) {

Обновляет экран..

single\_app.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View view) {  
 Intent intent = new Intent(caller, SingleAppActivity.class);  
 intent.putExtra("US", single.getUsageStats());  
 intent.putExtra("time", single.getTimeInforground());  
 intent.putExtra("all", app\_all.getTimeInforground());  
 startActivityForResult(intent, *SETTINGS\_ACT*);  
 }  
});

на каждое окошко приложение мы сосздаем событие, т.е. при нажатии на окно приложение мы перейдем в активити которое содержит только инфу о нем.(графики (кольцевые/столбчатые) и настройку лимитов по конкретному приложению.

**SettingsActivity (дейятельность экрана настроек)**



**SharedPreferences** — постоянное хранилище на платформе Android, используемое приложениями для хранения своих настроек, например. **Это** хранилище является относительно постоянным, пользователь может зайти в настройки приложения и очистить данные приложения, тем самым очистив все данные в хранилище.

Опять же ключевые моменты.

public static final String *APP\_PREFERENCES* = "mysettings";

обьявление имени нашего хранилища с настройками.

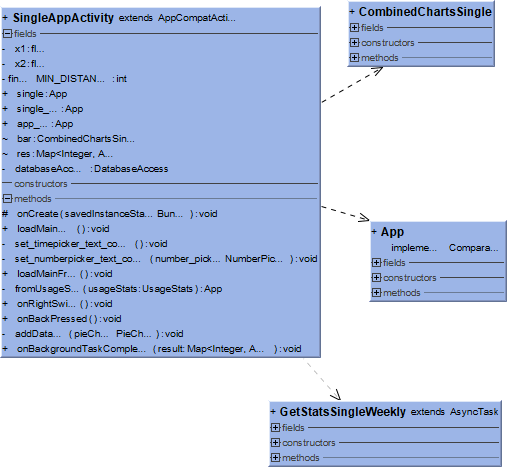
pref = new SharPreferences(getSharedPreferences(*APP\_PREFERENCES*, Context.*MODE\_PRIVATE*));  
pref.loadLimitSwitch(getIntent());  
pref.loadLimit(getIntent());  
pref.loadNotific(getIntent());  
pref.loadPopNotific(getIntent());

загружаем нужные сведения из хранилища. Логика загрузки будет описана чуть позже

final Switch notific\_swtch = (Switch) findViewById(R.id.*notific\_swtch*);  
notific\_swtch.setChecked(pref.isNotific());  
notific\_swtch.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 pref.saveNotific(getIntent(),notific\_swtch.isChecked());  
 }  
});  
  
  
final Switch push\_swtch = (Switch) findViewById(R.id.*push\_swtch*);  
push\_swtch.setChecked(pref.isPopNotific());  
push\_swtch.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 pref.savePopNotific(getIntent(),push\_swtch.isChecked());  
 }  
});

Обработчик кнопок переключений,при переключении сохраням состояние в SharedPreference.

**SingleAppActivity (дейятельность экрана настроек)**



public void loadMainStat()

Загрузка фрейма с настройками лимитов приложения

public void loadMainFrag()

Загрузка фрейма с диаграммами по приложению (кольцевая и столбцатая)

public void onRightSwipe() {  
 overridePendingTransition(R.anim.*right\_in*, R.anim.*left\_out*);  
 SingleAppActivity.this.finish();  
}  
  
@Override  
public void onBackPressed() {  
 overridePendingTransition(R.anim.*right\_in*, R.anim.*left\_out*);  
 SingleAppActivity.this.finish();  
  
}

Обраотка выхода с активити

private void addDataSet(PieChart pieChart) {

Внесение данных в кольцевую диаграмму

public void onBackgroundTaskCompleted(final Map<Integer,App> result){

При завершении асинхронного потока, получает данные (Map<Integer,App> result)

bar.mainLoad(result);

и передает их в график столбчатый( недельный по приложению)