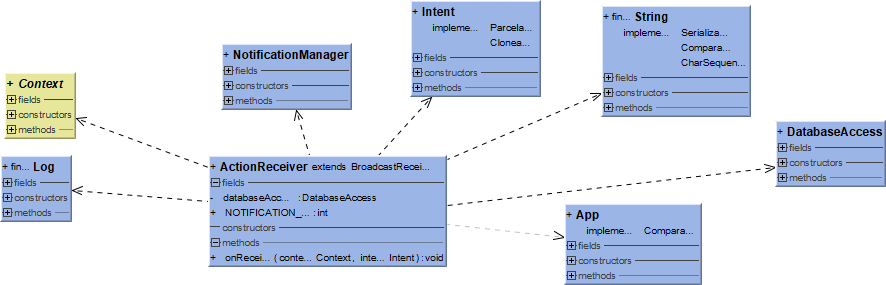
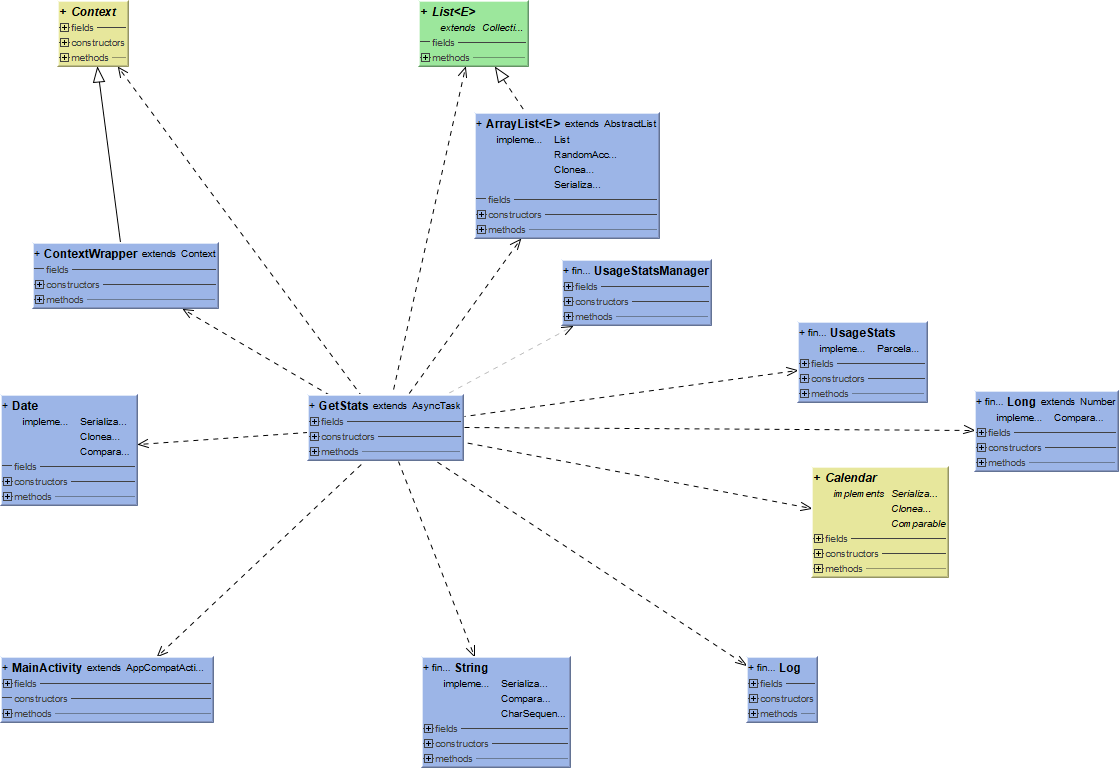
**ActionReceiver( класс отлова нажатия кнопки в уведомления)  
**

**Как только нажата кнопка +5мин…**

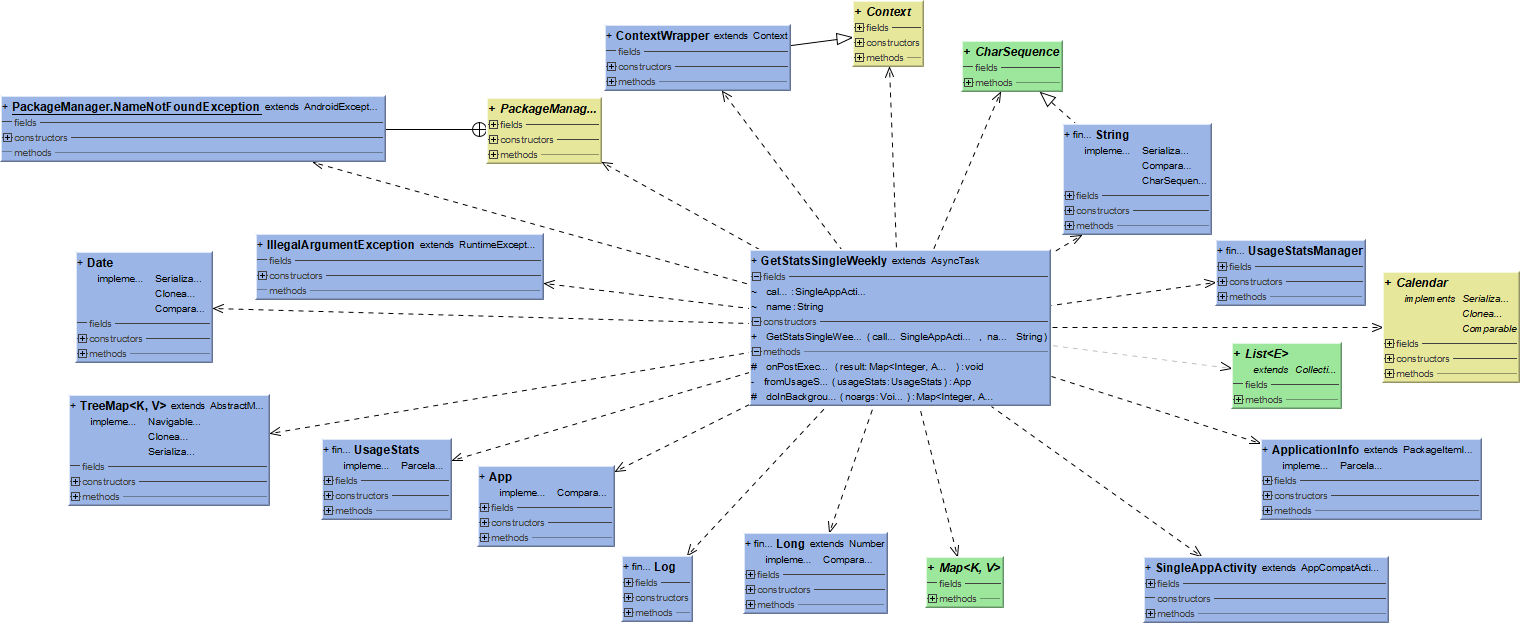
this.databaseAccess = DatabaseAccess.*getInstance*(context);  
String action = intent.getStringExtra("action");  
Log.*d*("onReceive: ", action);  
String[] list = action.split(",");  
for (int i = 0;i<list.length;i++) {  
 databaseAccess.open();  
 App single\_db = databaseAccess.selectOneByStr(list[i]);  
 databaseAccess.close();  
 if (single\_db.getLimit\_m()+5>=60){  
 single\_db.setLimit\_h(single\_db.getLimit\_h()+1);  
 single\_db.setLimit\_m((single\_db.getLimit\_m()+5)-60);  
 }  
 else{  
 single\_db.setLimit\_m((single\_db.getLimit\_m()+5));  
 }  
 databaseAccess.open();  
 databaseAccess.update\_limit(single\_db);  
 databaseAccess.close();  
}  
  
NotificationManager manager = (NotificationManager)context.getSystemService(Context.*NOTIFICATION\_SERVICE*);  
manager.cancel(NOTIFICATION\_ID);

**увеличиваем счетчики в бд и закрываем уведомление**

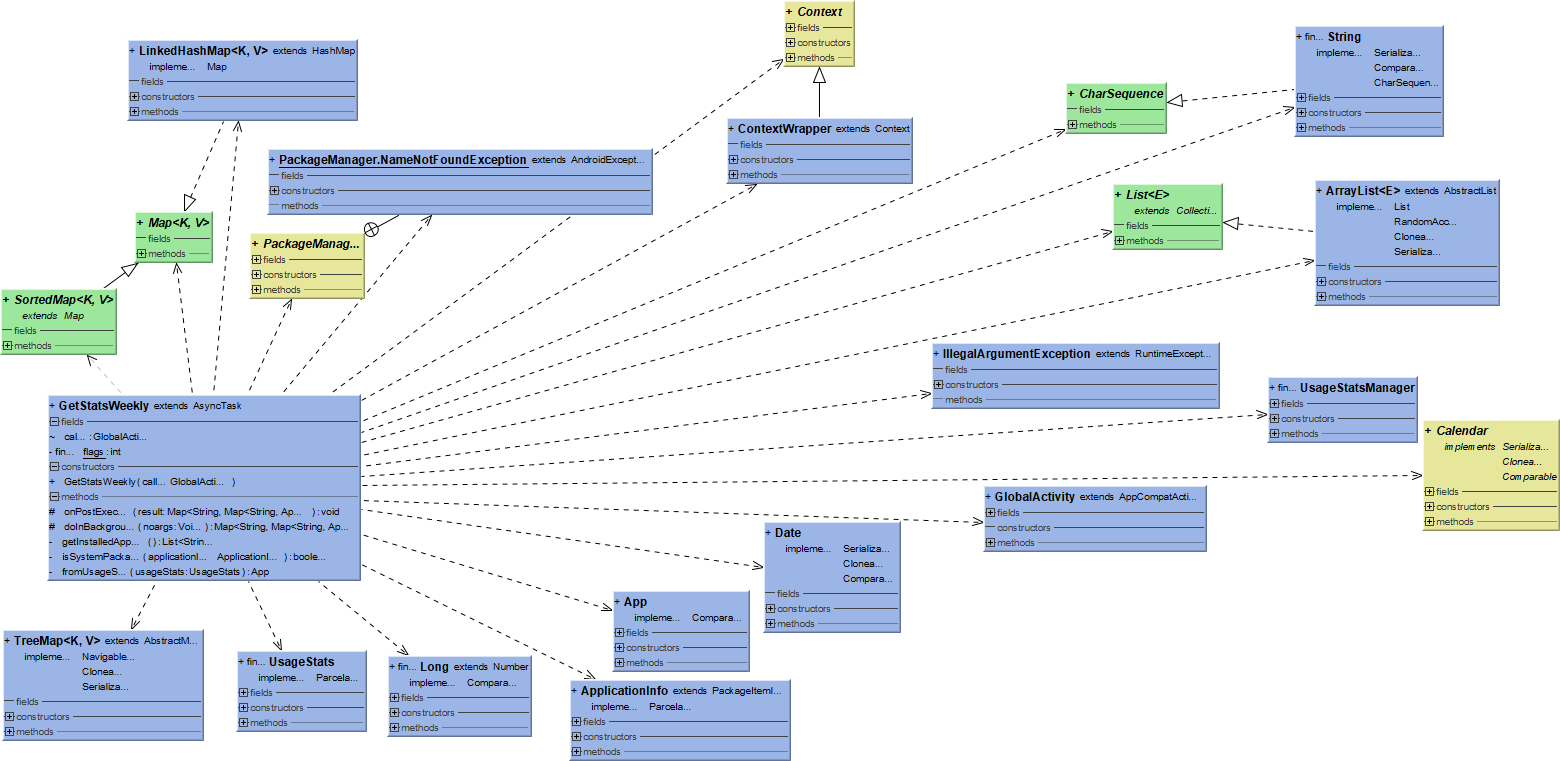
**GetStats(класс создающий асинхронный поток для сбора общей инфы за день)**



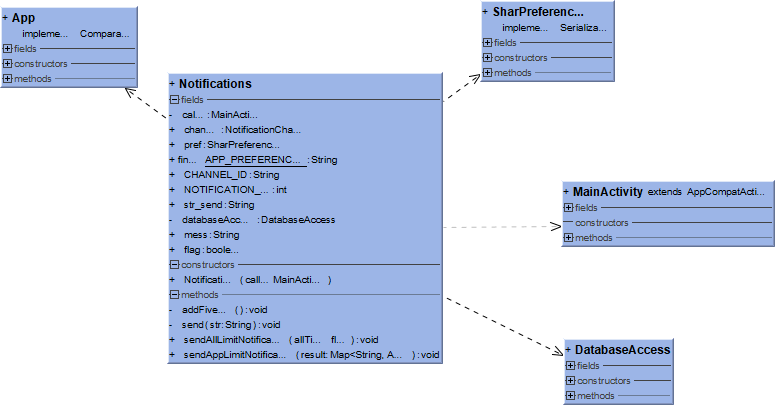
**GetStatsSingleWeekly(класс создающий асинхронный поток для сбора инфы по конкретному приложению за неделю)**



**GetStatsWeekly(класс создающий асинхронный поток для сбора инфы общей за неделю)**

****

**Notification(класс для работы с уведомлениями)**



public NotificationChannel channel;  
public String CHANNEL\_ID = "5MM";  
public int NOTIFICATION\_ID = 1515;

Переменные хранящие в себе описание создаваемого канала, для отправки уведомлений.

public String mess;

Данная текстовая переменная содержит уведомление с названиями приложений, которые превысили лимит и общего лимита , если он был достигнут.

public Notifications(MainActivity caller) {  
 this.caller = caller;  
 this.databaseAccess = DatabaseAccess.*getInstance*(caller);  
 pref = new SharPreferences(caller.getSharedPreferences(*APP\_PREFERENCES*, Context.*MODE\_PRIVATE*));  
 pref.loadLimitSwitch(caller.getIntent());  
 pref.loadLimit(caller.getIntent());  
 pref.loadNotific(caller.getIntent());  
 pref.loadPopNotific(caller.getIntent());  
 if (Build.VERSION.*SDK\_INT* >= Build.VERSION\_CODES.*O*) {  
 CharSequence name = "5MM";  
 String description = "description\_5mm";  
 int importance = 0;  
 if (pref.isPopNotific())  
 importance = NotificationManager.*IMPORTANCE\_MAX*;  
 else  
 importance = NotificationManager.*IMPORTANCE\_DEFAULT*;  
 channel = new NotificationChannel(CHANNEL\_ID, name, importance);  
 channel.setDescription(description);  
 channel.enableLights(true);  
 channel.setLightColor(Color.*RED*);  
 channel.setShowBadge(true);  
 channel.setLockscreenVisibility(Notification.*VISIBILITY\_PUBLIC*);  
 NotificationManager notificationManager = caller.getSystemService(NotificationManager.class);  
 notificationManager.createNotificationChannel(channel);  
 }  
}

Конструктор класса, создающий канал для отправки уведомлений.(а также, указывающий, включение индикации при уведомлении, задание иконки, и установления приоритет отображение (пробудить экран и показать/ или просто отобразить ))

public void sendAppLimitNotification(Map<String,App> result){  
 pref.loadLimitSwitch(caller.getIntent());  
 pref.loadLimit(caller.getIntent());  
 pref.loadNotific(caller.getIntent());  
 pref.loadPopNotific(caller.getIntent());  
 flag = false;  
 str\_send="";  
 if ((pref.isNotific())||(pref.isPopNotific())) {  
 if (result.size() > 0) {  
 String string = "";  
 databaseAccess.open();  
 List<App> apps = databaseAccess.getAllApps();  
 databaseAccess.close();  
 if (apps.size() > 0) {  
 for (Map.Entry<String, App> app : result.entrySet()) {  
 for (App list\_app : apps) {  
 if (list\_app.getAppName().equals(app.getValue().getAppName())) {  
 float limit = (list\_app.getLimit\_h() \* 60) + list\_app.getLimit\_m();  
 if (limit < app.getValue().CalcTimeCompare()) {  
 string += list\_app.getAppName() + ", ";  
 databaseAccess.open();  
 databaseAccess.update\_count(list\_app);  
 databaseAccess.close();  
 }  
 }  
 }  
 }  
 if (string.length() > 0) {  
 flag = true;  
 string = string.substring(0, string.length() - 2);  
 str\_send = string;  
 mess+="\n"+string+" limit is exceeded";  
 Class clazz = R.string.class;  
 int rnd = (int) (30 \* Math.*random*());  
 try {  
 Field field = clazz.getField("notifi\_"+rnd);  
 mess+="\n" + caller.getString(field.getInt(null));  
 }  
 catch (Exception ex){  
  
 }  
  
 }  
 }  
 }  
 if (mess.length()>0)  
 send(mess);  
 }  
}

Данный метод получает список все приложений, которые превысили лимит и подготавляивает текстовое сообщение об этом.

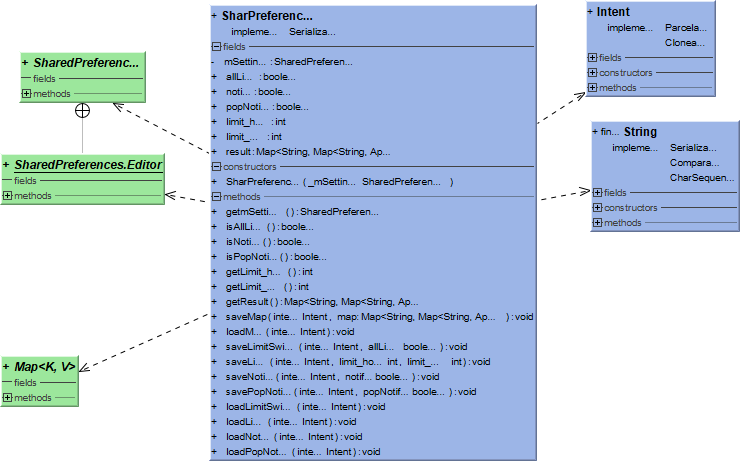
public void sendAllLimitNotification(float allTime) {  
 mess="";  
 pref.loadLimitSwitch(caller.getIntent());  
 pref.loadLimit(caller.getIntent());  
 pref.loadNotific(caller.getIntent());  
 pref.loadPopNotific(caller.getIntent());  
 if (((pref.isNotific())||(pref.isPopNotific())) && (pref.isAllLimit())) {  
 float limit\_all = (pref.getLimit\_hour() \* 60) + pref.getLimit\_min();  
 if (allTime > limit\_all) {  
 mess+="Total application limit is exceeded";  
 }  
 }  
}

Данный метод анализирует был ли превышен общий лимит приложений.Если да подготавливает текстовое сообщение об этом.

private void send(String str){  
 Intent intent = new Intent(caller, MainActivity.class);  
 intent.setFlags(Intent.*FLAG\_ACTIVITY\_NEW\_TASK* | Intent.*FLAG\_ACTIVITY\_CLEAR\_TASK*);  
 PendingIntent pendingIntent = PendingIntent.*getActivity*(caller, 0, intent, 0);  
  
  
 Intent addIntent = new Intent(caller, ActionReceiver.class);  
 addIntent.putExtra("action",str\_send);  
 PendingIntent addPendingIntent = PendingIntent.*getBroadcast*(caller,0,addIntent,0);  
  
 NotificationCompat.Builder mBuilder = new NotificationCompat.Builder(caller, CHANNEL\_ID)  
  
 .setSmallIcon(R.drawable.*ic\_timer\_black\_24dp*)  
 .setContentTitle("5 Minute More")  
 .setContentText("Limit is exceeded")  
 .setAutoCancel(true)  
 .setContentIntent(pendingIntent)  
 .setLargeIcon(BitmapFactory.*decodeResource*(caller.getApplicationContext().getResources(), R.drawable.*ic\_vector\_name*))  
 .setStyle(new NotificationCompat.BigTextStyle().bigText(str));  
  
if (flag)  
 mBuilder.addAction(R.drawable.*ic\_add\_alert\_black\_24dp*, "Add 5 more minutes", addPendingIntent);  
  
 mBuilder.setDefaults(Notification.*DEFAULT\_SOUND* | Notification.*DEFAULT\_VIBRATE* | Notification.*DEFAULT\_LIGHTS* );  
 NotificationManagerCompat notificationManager = NotificationManagerCompat.*from*(caller);  
 notificationManager.notify(NOTIFICATION\_ID, mBuilder.build());  
 }

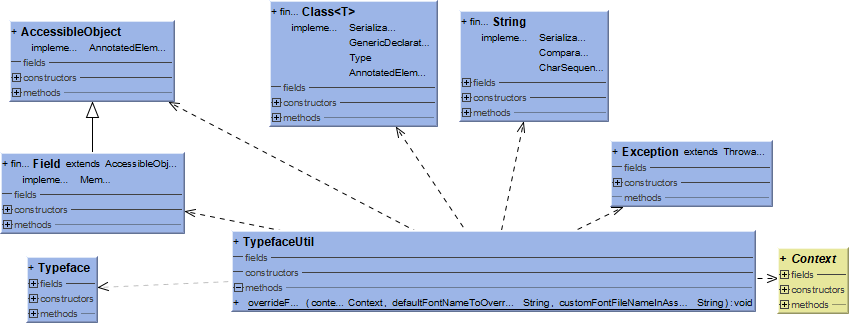
Данный метод, формирует и отображает уведомление.

**SharPreferences(класс которые сохраняет и извлекает данные из хранилища)**

****

Класс содержит методы извлечения инфы и ее сохранения.

**TypefaceUtil( для установки пользовательского шрифта)**

****