Burak Can Kuş Ödev Raporları

Ödev 1: Taksi Durağı Uygulaması

Bu ödevin amacı http://bil4118.somee.com/api/cabstand server'ına erişerek uygun taksi duraklarını sorgulayan ve bu durakları, kullanıcıya mesafeleriyle birlikte haritalar üzerinde gösteren bir Android uygulaması yazmaktır. Taksi duraklarının telefon numaraları da uygulamada gösterilmelidir.

1 Kod Anlatımı

Location.java

```
public class Location implements Serializable {
    public String name;
    public double lat;
    public double lon;
    public String phone;
    public double distance;
}
```

Bu java class'ı dizilerde tutulacak bilgilerin ve bir aktiviteden diğer aktiviteye aktarımın kolaylaştırılması için (Serializable) yazılmıştır.

MapsActivity.java

```
private GoogleMap mMap;
Location stop;

double lat = 999999;
double lon = 999999;

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_maps);
    // Obtain the SupportMapFragment and get notified when the map is ready to be used.
    SupportMapFragment mapFragment = (SupportMapFragment) getSupportFragmentManager()
            .findFragmentById(R.id.map);
    mapFragment.getMapAsync(this);
    Intent intent = this.getIntent();

    Bundle bundle = intent.getExtras();
    stop = (Location)bundle.getSerializable("stop");
    lat = bundle.getDouble("my_lat");
    lon = bundle.getDouble("my_lon");
}
```

stop (durak) isimli Location objesi ve kullanıcının bulunduğu konumun lat ve lon değerleri bundle'dan alınır. Bu bundle `MainActivity.java`'dan gelmektedir.

MapsActivity.java

```
@Override
public void onMapReady(GoogleMap googleMap) {
    mMap = googleMap;

    if (lat != 999999 && lon != 999999) {
        LatLng myloc = new LatLng(lat, lon);
        mMap.addMarker(new MarkerOptions().position(myloc).title("My location"));
    }

    LatLng stoploc = new LatLng(stop.lat, stop.lon);
    mMap.addMarker(new MarkerOptions().position(stoploc).title(stop.name));
    mMap.moveCamera(CameraUpdateFactory.zoomTo(12));
    mMap.moveCamera(CameraUpdateFactory.newLatLng(stoploc));
}
```

Ekranda kullanıcının konumuna bir marker ve durağın konumuna bir marker eklenir.

MainActivity.java

```
@Override
protected void onCreate(Bundle savedInstanceState);
```

MainActivity'de onCreate methodunda, kullanıcıdan konum izni alınmaya çalışılır, eğer başarısız olunursa tekrar sorulur. Başarılı olunduğu takdirde intent bundle'ına konum eklenir ve AsyncWebCall execute edilir.

MainActivity.java

```
private class AsyncWebCall extends AsyncTask<String, List<Location>, List<Location>>

protected List<Location> doInBackground(String... params);
private List<Location> callApi() throws IOException, JSONException;
protected void onPostExecute(final List<Location> stops);
```

Bu class'daki methodlar sırasıyla şu görevleri yaparlar:

doInBackground() methodu bu class execute edildiğinde çalışır. İçerisinde callApi'a bir çağrı vardır.

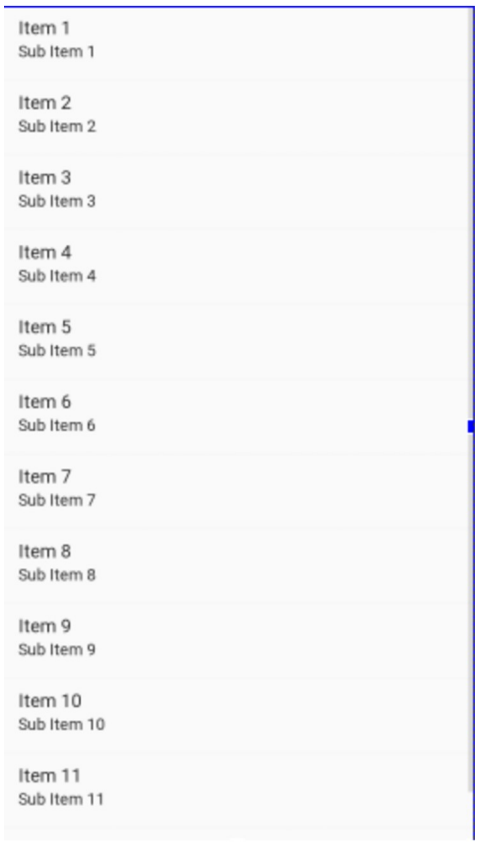
callApi() methodu "http://bil4118.somee.com/api/cabstand" methodu adresinden JSON objesi olarak taksi duraklarını almayı çalışır. Bu noktaya kullanıcının konumundan olan uzaklık `android.location.Location#distanceTo` methodu ile aşağıdaki şekilde bulunur.

```
Location stop = new Location();
stop.name = jsonObj.getString("Name");
stop.phone = jsonObj.getString("Phone");
stop.lat = jsonObj.getDouble("Latitude");
stop.lon = jsonObj.getDouble("Longitude");

android.location.Location stopLoc = new android.location.Location("stopLocation");
stopLoc.setLatitude(stop.lat);
stopLoc.setLongitude(stop.lon);
double distance = Math.round(location.distanceTo(stopLoc) * 100.0) / 100.0;
stop.distance = distance;
stops.add(stop);
```

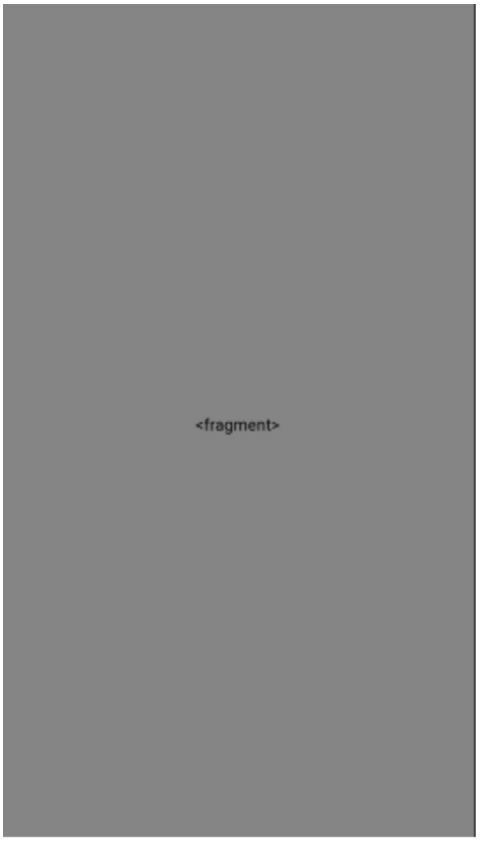
Daha sonra da bu duraklar, kullanıcıya olan mesafelerine göre https://stackoverflow.com/questions/16252269/how-to-sort-an-arraylist/16252290 cevabındaki gibi sıralanır.

onPostExecute methodunda ise bu duraklar bir ListView'a eklenir ve her birisi için onClickListener ayarlanır. Bu onClickListener'lar, ListView item'ı tıklanıldığı halde, bu durağı intent bundle'ına ekleyerek MapsActivity'i çağırır.

2 Ekran Görüntüleri



MainActivity görüntüsü



MapsActivity görüntüsü

3 Geliştirilebilecek Kısımlar

Uygulama şu anda server bağlantısı başarılı olmadığı için çalışamamaktadır. Uygulamada geliştirilecek bir iki şey kolayca bulunabilir. Güzel bir menü eklemek, listedeki durağa uzun süreli basıldığında telefon araması için soru sorulması bunlardan birkaçıdır.

Ödev 2: Telefon Kullanım Süresi Takip

Bu ödevde amaç, ebeveynlerin çocukların gün içerisinde telefonla ne kadar ilgilendiklerini veya vakit geçirdiklerini takip etmelerini sağlayacak bir uygulama geliştirmektir.

1 Kod Anlatımı

Ekranda ilk olarak gözüken Activity LoginActivity'dir. Bu ekran yaratıldığında ScreenListenService servisi başlatılır. Bu servis ekranın açılıp kapanma eventlerini veritabanına kaydeder.

Ekranda 2 radioButton bulunmaktadır. Ebeveyn veya çocuk seçeneği. Ebeveyn seçeneği işaretlenirse, şifre girme ekranı gözükür. Giriş butonuna tıklandığında ise ebeveyn seçiliyse, şifre ancak "admin"'e eşitse bir sonraki ekrana geçilir. Çocuk girişinde ise herhangi bir şifre sorulmaz.

LoginActivity.java

```
public static final String EXTRA_MESSAGE = "PARENT";
switch (radioGroup.getCheckedRadioButtonId()) {
    case R.id.childRadioBtn:
        intent.putExtra(EXTRA_MESSAGE, false);
        startActivity(intent);
        break;
    case R.id.parentRadioBtn:
        if (!password.getText().toString().equals("admin")){
            break;
        }
        intent.putExtra(EXTRA_MESSAGE, true);
        startActivity(intent);
        break;
}
```

MainActivity, sürelerin gösterimlerinin olduğu ekrandır. İki buton yardımıyla başlangıç ve bitiş zamanları seçilebilir, bu tarihler arasındaki açık olma süresi gösterilebilir ve tüm zamanlarda ekran açık olma süresi gösterilebilir. Eğer ebeveyn giriş yaptıysa, tüm geçmişi temizleme butonu bulunmaktadır. Onun dışında diğer işlevler aynıdır.

MainActivity.java

```
public void btnCalculate_onClick(View view);
```

methodu, tarihler arasındaki ekran açık olma süresini bulur ve tüm tarihlerdeki günlük ortalamayı hesaplayıp ekranda gösterir.

MainActivity.java

```
protected void showList();
```

methodu, veritabanından tarihler arasındaki ekran açılma ve kapanma eventlerini alır ve ekranda RecyclerView içerisinde gösterilmelerini sağlar. Bu method, yalnızca btnCalculate_onClick methodu çağrıldığında çağrılır.

MainActivity.java

```
public void btnRemoveAll_onClick(View view);
```

methodunda, veritabanından tüm verileri silmek için gerekli Query çağrılır.

MainActivity.java

```
protected void bindButtontoPicker(final Button button);
```

methodu, tarih seçme ekranını butona bağlar. onCreate methodunda, startDate ve endDate butonları için çağrılır.

MainActivity.java

```
protected void clearTimesFromCalendar(Calendar calendar) {
    calendar.clear(Calendar.HOUR);
    calendar.clear(Calendar.HOUR_OF_DAY);
    calendar.set(Calendar.HOUR_OF_DAY, 0);
    calendar.clear(Calendar.MINUTE);
    calendar.clear(Calendar.SECOND);
    calendar.clear(Calendar.MILLISECOND);
}
```

methodu, kendisine verilen calendar objesinden zaman bildiren bilgileri siler. Kodda birden fazla yerde kullanılmıştır.

ScreenListenService.java

class'ı, ekranın açılıp kapanma eventlerine, şimdiki zamanı ve hangi event olduğunu veritabanına yazacak işlevsellikleri bağlar. LoginActivity'de çalıştırılmaya başlanır.

TimeTableContract.java

class'ı, veritabanında tutulacak bilgilerin (tablo adı, sütun isimleri) tutulduğu static tablo class'larını içeren class'dır.

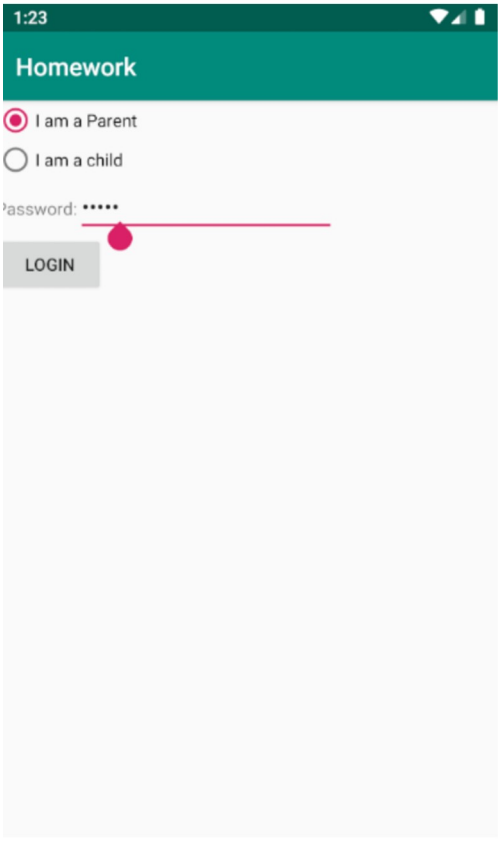
TimeTableContract.java

```
static class TimeEntry implements BaseColumns {
    static final String TABLE_NAME = "times";
    static final String COLUMN_NAME_UNIXTIME = "unixtime";
    static final String COLUMN_NAME_SCREENUNLOCKED = "screenunlocked";
}
```

TimeTableDbHelper.java

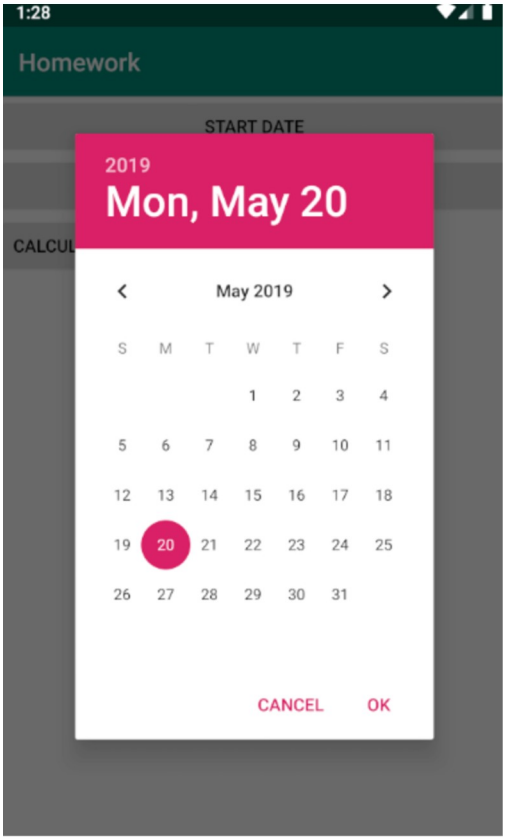
son olarak TimeTableDbHelper class'ı ise veritabanı versiyonunu, ismini, tablo oluşturma query'sini, tabloyu boşaltma query'sini, upgrade ve downgrade durumunda ne olacağı methodlarını barındıran bir class'dır. Veritabanına bağlantıyı yapan class'dır.

2 Ekran Görüntüleri

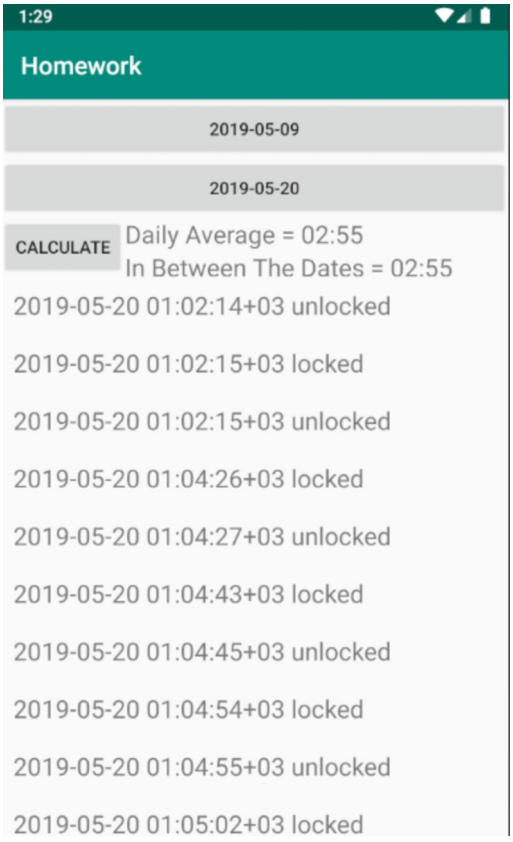


LoginActivity görüntüsü

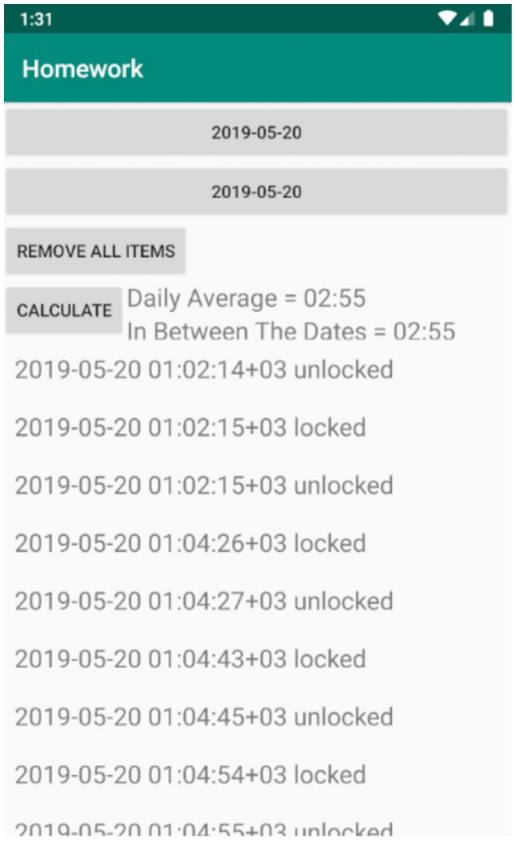
Çocuk girişi yapılmış ekran görüntüsü



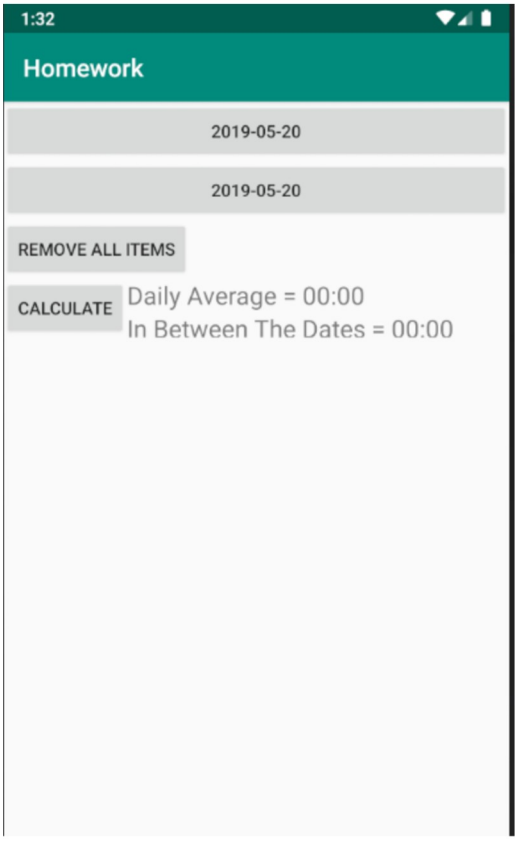
Tarih seçme ekranı



Calculate butonuna basılmış ekran görüntüsü



Ebeveyn ekranı görüntüsü



Tüm veriler silinmiş ekran görüntüsü

3 Geliştirilebilecek Kısımlar

Bu programın bu haliyle gayet iyi bir durumda olduğunu düşünüyorum. Geliştirmek için, UI düzeyinde düzenlemeler yapılabilir, ekran açma kapatmak için daha iyi bir yönteme geçilebilir (bu yöntemin pek iyi olmadığını bir yerlerden okuduğumu hatırlıyorum.

Ödev 3: Akselerometre Oyunu

Bu ödevde amaç kullanıcının akselerometre sensörü ile oyun içindeki topu yönlendirerek belirli bir hedefe ulaştırmaya çalıştığı bir oyun geliştirmektir.

1 Kod Anlatımı

MainActivity.java

```
public static final String EXTRA_MESSAGE = "hardMode";
```

MainActivity, oyunun giriş ekranıdır. Ekranda zor mod için bir Switch bulunmaktadır. Eğer zor mod seçilirse ekstra bir CheckBox gözükmektedir. Bu CheckBox insane mod için buradadır. Bunlar dışında Oyna ve Skorları Görüntüle butonları bulunmaktadır. İkisi de farklı birer Activity başlatır. Eğer hard mod seçili değilse, oyna butonuna basıldığında intente 0 extrası konulur. Hard mod seçiliyse 1 ve insane mod seçiliyse 2 değerleri konulur.

GameActivity.java

```
// Heavily updated from SensorApp example from Moodle
// and https://androidkennel.org/android-sensors-game-tutorial/

public class GameActivity extends AppCompatActivity implements SensorEventListener
```

Bu activity, oyunun oynandığı aktivitedir. onCreate methodunda veritabanına bir bağlantı açılır, gelen intentten oyunun modunun ne olduğu bulunur (mod kolay ise 2 engel, zor ise 4 engel, çok zor ise tüm engeller ekranda belirir), sensorler okunmaya başlanır. Sensorler her değiştiğinde onSensorChanged methodu çalışır ve X ve Y eksenlerinde telefonun ne kadar oynadığını (yerçekimi ivmesi) alır.

GameActivity.java

```
// Sensor Operations
@Override
public void onSensorChanged(SensorEvent event) {
    if (startGame) {
        if (event.sensor.getType() == Sensor.TYPE_ACCELEROMETER) {
            xAccel = event.values[0];
            yAccel = -event.values[1];
            updateBall();
        }
    }
}
```

updateBall methodu ekrandaki topun hareketlerini, colision detection'ı yapan methoddur. Maksimum velocity (hız) burada belirlenebilir. Oyun tamamlandıysa, yani top hedef noktaya dokunduysa, skorun, oyunun zorluğunun ve oyunun oynanma zamanının veritabanına yazılması da bu methodda olur. Ekranda 3.5 saniye boyunca bir Toast gösterildikten sonra bu aktivite finish() ile kapatılır.

GameActivity.java

```
// Ball update, this also checks for collisions
private void updateBall() {
    ...
    ...
    ...
    ...
    // Game Completed, write to database
    if (Math.pow((xMax / 2) - xPos, 2) + Math.pow(ballRadius - yPos, 2) <= Math.pow(2 *
ballRadius, 2)) {
        ContentValues values = new ContentValues();
        Date endDate = Calendar.getInstance().getTime();
        long diffInMs = endDate.getTime() - startDate.getTime();
        long diffInSec = TimeUnit.MILLISECONDS.toSeconds(diffInMs);
        Log.i("Time", " " + diffInSec);
        values.put(TimeTableContract.TimeEntry.COLUMN_NAME_TIMETAKENINSECONDS,
diffInSec);
        values.put(TimeTableContract.TimeEntry.COLUMN_NAME_HARDMODE, hardMode);
        db.insert(TimeTableContract.TimeEntry.TABLE_NAME, null, values);

        sensorManager.unregisterListener(this);
        Toast.makeText(this, "Congratulations! Game completed in " + diffInSec + "
seconds!", Toast.LENGTH_LONG).show();

        // https://stackoverflow.com/a/10032406/6077951
        new CountDownTimer(3500, 1000) {
            public void onTick(long millisUntilFinished) {
            }

            public void onFinish() {
                setResult(Activity.RESULT_OK);
                finish();
            }
        }.start();
    }
}
```

GameActivity.java

```
// Sensor Operations
class MyDraw extends View
```

GameActivity class'ı, MyDraw isimli bir başka class da içerir. Bu class, ekranda oyun alanı canvasını göstermekle görevlidir. onSizeChanged methodunun içinde ekran boyutları belli olduğu için, obstacle üretme kısmı da burada yapılmaktadır. onDraw methodunda sadece çizim yapılmaktadır, herhangi bir işlem yapılmaz.

ResultsActivity.java

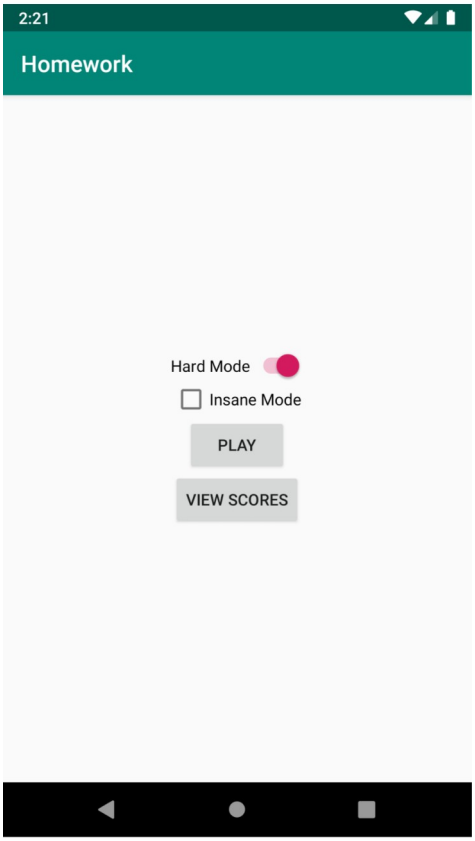
ResultsActivity, "Sonuçları Göster" butonuna basıldığında ekrana gelecek olan Activity'dir. Oyunun oynanma zamanına göre sıralı şekilde veritabanından skorları (oyunu tamamlama süresini) alıp, ekranda ListView'da bu bilgiyi ve hangi zorlukta olduğunu gösterir.

ResultsActivity.java

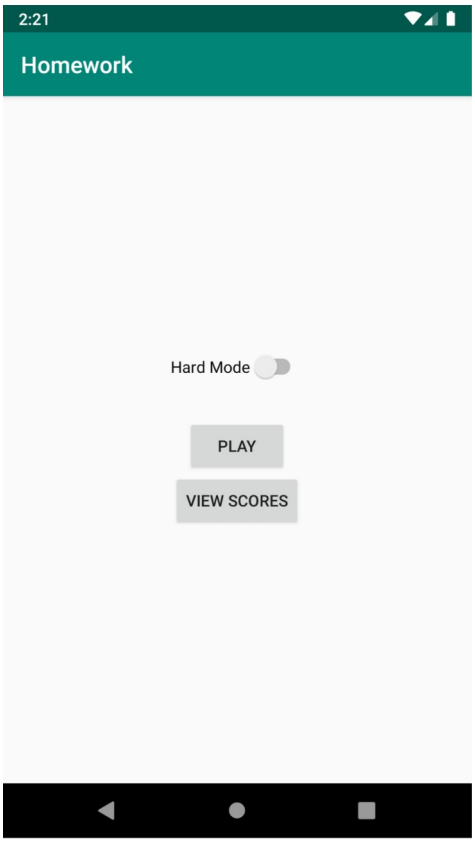
```
class Score {
    String date;
    String text;

    Score(String date, String text) {
        this.date = date;
        this.text = text;
    }
}
```

iki satırlı gösterimin ListView'da kolay yapılması için bu Activity class'ı Score adında başka bir class içerir. Text, ekranda gösterilecek zamanı, date ise oyunun oynanıldığı tarihi gösterir.

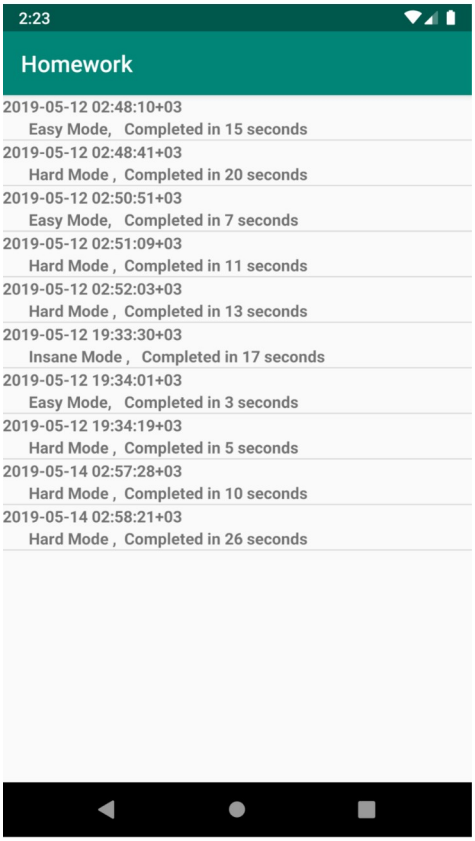
2 Ekran Görüntüleri



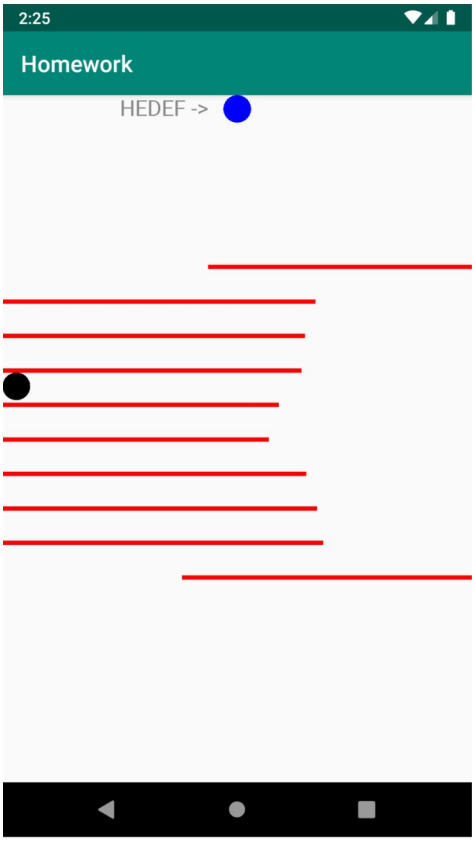
Ana ekran görüntüsü (zor)



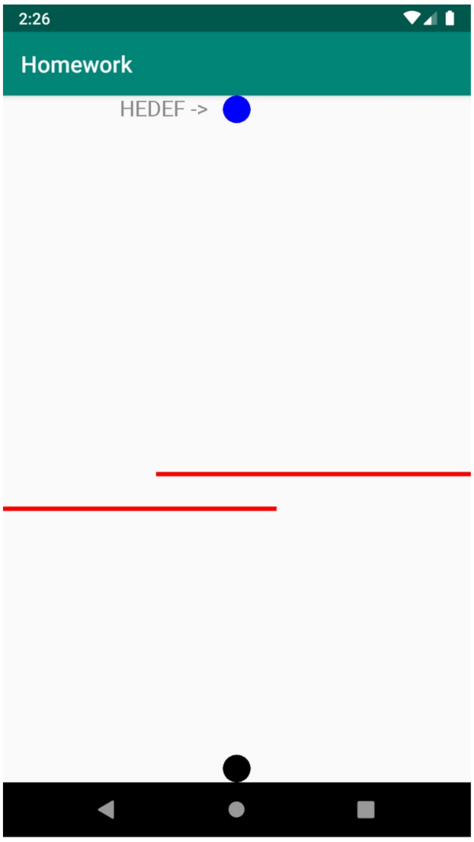
Ana ekran görüntüsü (kolay)



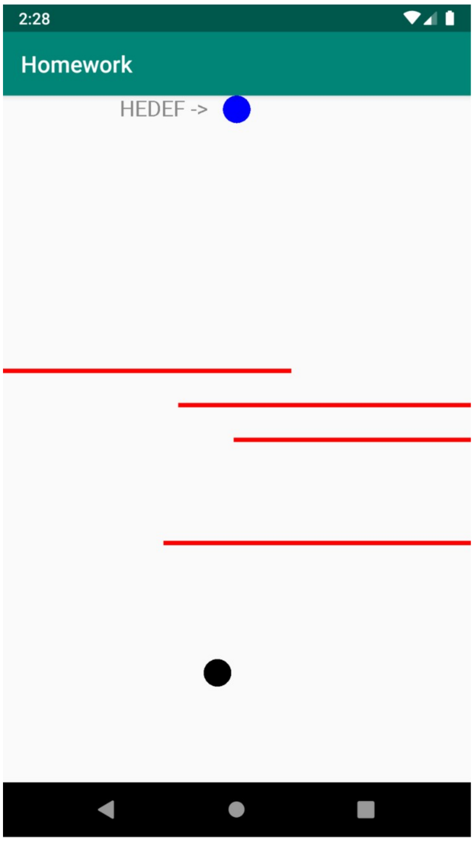
Skorlar ekranı görüntüsü



Insane mod oyun örneği



Easy mod oyun örneği



Hard mod oyun örneği

3 Geliştirilebilecek Kısımlar

Oyun bu haliyle oynanılabilir bir oyun, fakat değişken bir şey pek yok. Farklı şekilde engeller, ölümcül engeller, sürekli yer değiştiren hedefler veya uzun bir platformda sonsuza kadar devam eden şekilde bir ekran tasarımı yapılabilir. Görsellik daha güzel hale getirilebilir, dokunma; hızlanma gibi olaylar için ses efektleri veya titreşim özelliği eklenebilir. Çarpışma olduğunda kıvılcım görsel efektleri gibi eğlenceli özellikler eklenerek daha iyi bir hale getirilebilir.