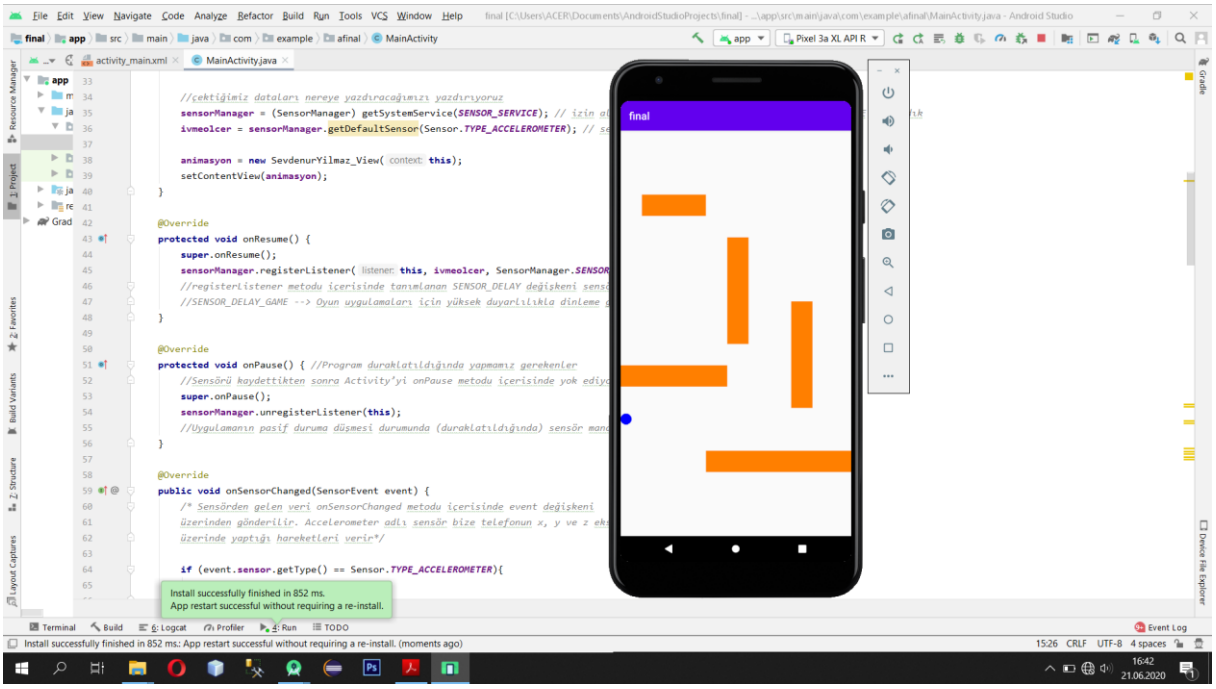
MOBİL PROGRAMLAMA FİNAL ÖDEVİ

SORU1)

```
public class SevdenurYilmaz_Activity extends AppCompatActivity implements SensorEventListener {

    private SensorManager sensorManager; // sensör manager tanımladık
    private Sensor ivmeolcer ; // sensör tanımladık
    private SevdenurYilmaz_View animasyon = null;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        requestWindowFeature(Window.FEATURE_NO_TITLE);

getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,WindowManager.LayoutParam
s.FLAG_FULLSCREEN);
        setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_UNSPECIFIED);

        //çektiğimiz dataları nereye yazdıracağımızı yazdırıyoruz
        sensorManager = (SensorManager) getSystemService(SENSOR_SERVICE); // izin almak için
getSystemService,izni SENSOR_SERVİCE den aldık
        ivmeolcer = sensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER); // sensörümüzün
tipini belirliyoruz -- ivmeölçer

        animasyon = new SevdenurYilmaz_View(this);
        setContentView(animasyon);
    }

    @Override
    protected void onResume() {
        super.onResume();
        sensorManager.registerListener(this, ivmeolcer, SensorManager.SENSOR_DELAY_GAME);
        //registerListener metodu içerisinde tanımlanan SENSOR_DELAY değişkeni sensörün
duyarlılığını belirler.
        //SENSOR_DELAY_GAME --> Oyun uygulamaları için yüksek duyarlılıkla dinleme
gerçekleştirir.
    }

    @Override
    protected void onPause() { //Program duraklatıldığında yapmamız gerekenler
        //Sensörü kaydettikten sonra Activity'yi onPause metodu içerisinde yok ediyoruz
        super.onPause();
        sensorManager.unregisterListener(this);
        //Uygulamanın pasif duruma düşmesi durumunda (duraklatıldığında) sensör managera veri
akışının devamını sağlıyoruz
    }

    @Override
    public void onSensorChanged(SensorEvent event) {
        /* Sensörden gelen veri onSensorChanged metodu içerisinde event değişkeni
        üzerinden gönderilir. Accelerometer adlı sensör bize telefonun x, y ve z ekseni
        üzerinde yaptığı hareketleri verir*/

        if (event.sensor.getType() == Sensor.TYPE_ACCELEROMETER){
            animasyon.onSensorEvent(event);
        }
    }
```

```
    @Override
    public void onAccuracyChanged(Sensor sensor, int accuracy) {
    }

    public static class SevdenurYilmaz_View extends View {
        Context context;
        static final int topR = 26; // top yarıçapını belirledik (2X)
        public Paint top, d1,d2,d3,d4,d5;
        private int x=100; // topun bulunduğu konumu belirliyoruz
        private int y=100; // topun bulunduğu konumu belirliyoruz
        private int viewWidth;
        private int viewHeight;

        public SevdenurYilmaz_View(Context context) {
            super(context);
            this.context=context;
            top=new Paint();
            d1 = new Paint();
            d2 = new Paint();
            d3 = new Paint();
            d4 = new Paint();
            d5 = new Paint();
            top.setColor(Color.BLUE); // renklerini belirliyoruz
            d1.setColor(Color.rgb(255,127,0));
            d2.setColor(Color.rgb(255,127,0));
            d3.setColor(Color.rgb(255,127,0));
            d4.setColor(Color.rgb(255,127,0));
            d5.setColor(Color.rgb(255,127,0));
        }

        @Override
        protected void onSizeChanged(int w, int h, int oldw, int oldh){
            super.onSizeChanged(w,h,oldw,oldh);
            viewWidth=w;
            viewHeight=h;
        }

        public void onSensorEvent(SensorEvent event) {
            //float [] values = event.values; //Sensörden gelen değerler values dizisi içinde gönderilir
            //float x = values[0]; //dizinin 1. elemanı
            //float y = values[1];
            //float z = values[2];
            //Log.d("SevdenurYilmaz_Activity", String.format("x : %f y : %f z : %f", x, y, z)); //Her eksende yapılan
hareketi LogCat içerisinde ekrana yazdırıyoruz

            x = x - (int) event.values[0];
            y = y + (int) event.values[1];

            if (x <= 0 + topR) {
                x = 0 + topR;
            }
            if (x >= viewWidth - topR) {
                x = viewWidth - topR;
            }
            if (y <= 0 + topR) {
                y = 0 + topR;
            }
            if (y >= viewHeight - topR) {
                y = viewHeight - topR;
```

```
        }
    }


    protected void onDraw(Canvas canvas){
        canvas.drawCircle(x,y,topR, top); // topu çizdik
        Rect r1 = new Rect(400,300,100,400); //dikdörtgen tanımladı ve boyutları ayarlandı
        Rect r2 = new Rect(0,1200,500,1100);
        Rect r3 = new Rect(500,500,600,1000);
        Rect r4 = new Rect(800,800,900,1300);
        Rect r5 = new Rect(1600,1500,400,1600);
        canvas.drawRect(r1,d1);
        canvas.drawRect(r2,d2);
        canvas.drawRect(r3,d3);
        canvas.drawRect(r4,d4);
        canvas.drawRect(r5,d5);
        if (carpisma(x,y, topR,r1)||carpisma(x,y, topR,r2)||carpisma(x,y, topR,r3)||carpisma(x,y,
topR,r4)||carpisma(x,y, topR,r5)){//kesişip kesişmediğine bakıyorum
            Log.v("ÇARPIŞMA", "Çarpışma oldu");
            canvas.drawCircle(x,y,topR,top);
        }
        invalidate();
    }

    public boolean carpisma(int x, int y, int r, Rect r1){
        boolean intersects = false;
        if(x+r >= r1.left && x-r <= r1.right){
            if(y+r >= r1.top && y-r <= r1.bottom){
                intersects = true;
            }
        }

        return intersects;

        // çarpışma olması durumu r1+r2>=aradaki mesafe
        //çarpışma olmaması durumu r1+r2<aradaki mesafe

        //Dörtgenin çemberin merkezine en yakın olan mesafesinden çemberin yarı çapı çıkartılır. Elde edilen
mesafe yarıçaptan büyük ise çarpışma gerçekleşmiş olur.


        /* if (Math.abs(y-r1.top) < r){
            if (x>=r1.left){
                if (y<=r1.right){
                    intersects=true;
                }else{
                    intersects=true;
                }
            }else{
                intersects=true;
            }
        } */

    }
  }
}
```



SORU2)

a)

TYPE_ACCELEROMETER : İvmeölçer
TYPE_AMBIENT_TEMPERATURE : Ortam Sıcaklığı
TYPE_GRAVITY : Yerçekimi
TYPE_GYROSCOPE : Jiroskop(Denge)
TYPE_LIGHT : Işık
TYPE_LINEAR_ACCELERATION : Doğrusal İvme
TYPE_MAGNETIC_FIELD : Manyetik alan
TYPE_ORIENTATION : Yön belirleme
TYPE_PRESSURE : Basınç
TYPE_PROXIMITY : Yakınlık
TYPE_RELATIVE_HUMIDITY : Nem
TYPE_ROTATION_VECTOR : Dönme Vektörü
TYPE_TEMPERATURE : Sıcaklık

b)

b.1) TYPE_ACCELEROMETER (İVMEÖLÇER) : Yerçekimi kuvveti de dahil olmak üzere her üç fiziksel eksende (x, y ve z) bir cihaza uygulanan m / s ^{2} cinsinden ivme kuvvetini ölçer.

b.2) TYPE_AMBIENT_TEMPERATURE (ORTAM SICAKLIĞI) : Ortam oda sıcaklığını Santigrat (° C) cinsinden ölçer.

b.3) TYPE_GRAVITY (YERÇEKİMİ) : Üç fiziksel eksenin (x, y, z) üzerindeki bir aygıta uygulanan m / s ^{2} cinsinden yerçekimi kuvvetini ölçer.

b.4) TYPE_GYROSCOPE (JİROSKOP-DENGE) : Bir cihazın üç fiziksel eksenin (x, y ve z) her biri etrafında rad / s cinsinden dönme hızını ölçer.

b.5) TYPE_LIGHT (IŞIK) : Ortam ışığı seviyesini (aydınlatma) lx cinsinden ölçer.

b.6) TYPE_LINEAR_ACCELERATION (DOĞRUSAL İVME) : Yerçekimi kuvveti hariç üç fiziksel eksenin (x, y ve z) her birindeki cihaza uygulanan m / s^2 cinsinden ivme kuvvetini ölçer.

b.7) TYPE_MAGNETIC_FIELD (MANYETİK ALAN) : MT cinsinden her üç fiziksel eksen (x, y, z) için ortam jeomanyetik alanını ölçer.

b.8) TYPE_ORIENTATION (YÖN BELİRLEME) : Bir cihazın üç fiziksel eksenin (x, y, z) etrafında yaptığı dönüş derecelerini ölçer.

b.9) TYPE_PRESSURE (BASINÇ) : HPa veya mbar cinsinden ortam hava basıncını ölçer.

b.10) TYPE_PROXIMITY (YAKINLIK) : Bir nesnenin aygıtın görüntüleme ekranına göre cm cinsinden yakınlığını ölçer. Bu sensör tipik olarak bir ahizenin bir kişinin kulağına kadar tutulup tutulmadığını belirlemek için kullanılır.
b.11) TYPE_RELATIVE_HUMIDITY (NEM) : Bağıl ortam nemini yüzde (%) olarak ölçer.

b.12) TYPE_ROTATION_VECTOR (DÖNME VEKTÖRÜ) : Aygıtın döndürme vektörünün üç öğesini sağlayarak aygıtın yönünü ölçer.

b.13) TYPE_TEMPERATURE (SICAKLIK) : Cihazın sıcaklığını Santigrat (° C) cinsinden ölçer.

c)

c.1) https://github.com/serkankaya/Android_Sensor_Example
Uygulamada kullanılan sensör: TYPE_ACCELEROMETER
Kullanım Amacı : Mobil cihazın fiziksel hareketlerine bağlı olarak ekranda görünecek nesnenin hareket etmesi

c.2) https://github.com/dipanjal/ProfileManager-sensor
Uygulamada kullanılan sensörler: TYPE_ACCELEROMETER (İvmeölçer) / TYPE_PROXIMITY(Yakınlık) / TYPE_LIGHT (Işık)
Kullanım Amacı : Cihazın X, Y, Z koordinatlarındaki bilgisini verir / Herhangi bir fiziksel temas olmadan yakın nesnelerin varlığını tespit etmeye yarayan sensördür. Telefonu kulağınıza götürdüğünüzde ekranın sönmesini sağlar / Ortam aydınlatmasını sağlar

c.3) https://github.com/kishan2612/Proxsor
Uygulamada kullanılan sensör: TYPE_PROXIMITY (Yakınlık sensörü)

Kullanım Amacı: Arama yaparken ve telefon arama yapmak veya almak için yüze tutulduğunda, sensör onu algılar ve ciltten istenmeyen girişleri önlemek için dokunmatik ekranı devre dışı bırakır.

c.4) https://github.com/frybitsinc/scroll_by_accelerometer
Uygulamada kullanılan sensör: TYPE_ACCELEROMETER (İvmeölçer)
Kullanım Amacı: Üç eksen boyunca hareketi ve tam oryantasyonu belirlemek için hızlanma, titreşim ve eğimi algılar. Bu uygulamada hızlanmayı algılıyor.

c.5) https://github.com/Iamsdt/CameraandSensorDemo
Uygulamada kullanılan sensör: TYPE_LIGHT (Işık)
Kullanım Amacı: ekran parlaklığını buna göre ayarlamak için yakındaki ışık düzeylerini algılar. Otomatik Parlaklık Ayarlayıcısında, ışığın kullanılabilirliğine bağlı olarak akıllı telefon ekranının parlaklığını azaltmak veya artırmak için kullanılır.