



— İSTANBUL —
OKAN ÜNİVERSİTESİ

İSTANBUL OKAN ÜNİVERSİTESİ
MESLEK YÜKSEKOKULU
BİLİŞİM TEKNOLOJİLERİ BİLGİSAYAR
PROGRAMCILIĞI BÖLÜMÜ
ANDROID PROGRAMLAMAYA GİRİŞ
Sudoku Uygulaması
Danışman: NİLGÜN İNCEREİS
18MY03009 – GÖRKEM ACAR

İÇİNDEKİLER

Giriş	2
Genel Bilgiler.....	2
<input type="checkbox"/> Kotlin nedir?	2
<input type="checkbox"/> Kotlin nerelerde kullanılır?	2
<input type="checkbox"/> Kotlin'in avantajları nelerdir?	2
<input type="checkbox"/> Android Studio nedir?	2
<input type="checkbox"/> Emülator nedir?.....	2
<input type="checkbox"/> AVD Manager nedir?.....	2
<input type="checkbox"/> SDK nedir?.....	2
Gelişme	3
<input type="checkbox"/> Projemi nasıl oluşturdum?	3
<input type="checkbox"/> Nelere ihtiyaç duydum?	4
<input type="checkbox"/> Proje algoritması	4
<input type="checkbox"/> Butonları ve tabloyu oluşturma.....	4
<input type="checkbox"/> Hücreler ve seçim ayarları	4
<input type="checkbox"/> Kullanıcı input ayarları	5
Sonuç	7
Zaman Çizelgesi.....	7
Ekler.....	8
Kaynakça.....	9

Giriş

Uygulamaya giren kullanıcı başlangıç olarak yukarıda grid e aktarılmış olan rakamları 9x9 şekilde görecektir. Aşağıdaki kısımda ise butonlar halinde rakamlar bulunacak. Kullanıcı rakamları yukarıdaki duruma göre dizebilir. Eğer işlemini geri almak istiyorsa silebilir. Programdan çıkmak istiyorsa da çarpı tuşuna basarak çıkabilir.

Genel Bilgiler

- **Kotlin nedir?**

Kotlin JVM (Java Virtual Machine) üzerinde çalışan bir programlama dilidir. Kotlin yazılım dili 2011 yılında [JetBrains](#) firması tarafından geliştirilmeye başlandığı duyuruldu ve ilk stabil sürümünü (v1.0) 2016 yılında yayınladı.

- **Kotlin nerelerde kullanılır?**

JVM -> Java kütüphanelerini kullanarak Java sanal makinesi üzerinde çalışacak sunucu tabanlı uygulamalar geliştirebilirsiniz.

Android -> Java ile karşılaştırıldığında hiçbir kısıtlama olmadan Android uygulamaları geliştirebilirsiniz.

Browser -> Kotlin ile yazdığınız kodları Javascript olarak derleyip uygulamalar geliştirebilirsiniz.

- **Kotlin'in avantajları nelerdir?**

Birçok Kotlin özelliği performanstan ve güvenlikden göz ardı etmeden Java'dan daha sade ve anlaşılabilir kodu etkinleştirir.

Kotlin'i avantajlı kılan bir diğer özellik ise, Java'ya göre kod satırlarında göz ile görülür oranda daha kısa kod satırları bulunmaktadır.

- **Android Studio nedir?**

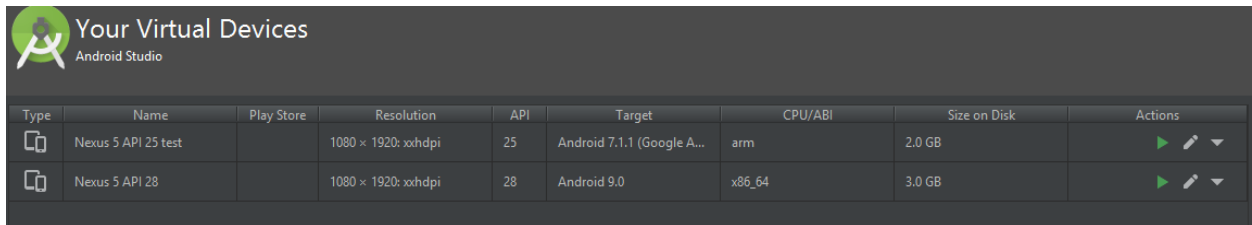
Adından da anlaşılacağı üzere Android işletim sistemli cihazlar telefon, tablet ve giyilebilir cihazlar için uygulama geliştirme platformudur.

- **Emülator nedir?**

Yaptığımız uygulamayı telefon vb. yerlerde çalıştırmamız gerekir. Android Studio içerisinde kendimize emülatör kurarak uygulamamızın nasıl çalıştığını görebiliriz.

- **AVD Manager nedir?**

Android Studio programında AVD Manager bölümüne giderek kendimize sanal cihaz oluşturabiliriz ve uygulamamızı görüntüleyebiliriz.



Type	Name	Play Store	Resolution	API	Target	CPU/ABI	Size on Disk	Actions
	Nexus 5 API 25 test		1080 x 1920: xhdpi	25	Android 7.1.1 (Google A...	arm	2.0 GB	
	Nexus 5 API 28		1080 x 1920: xhdpi	28	Android 9.0	x86_64	3.0 GB	

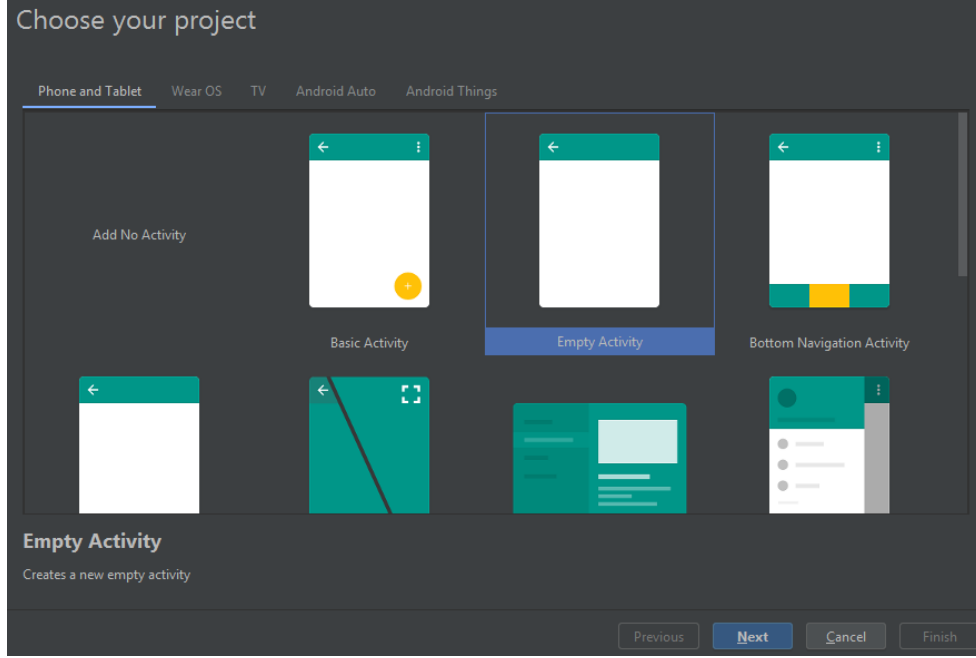
- **SDK nedir?**

SDK (software development kit) veya **yazılım geliştirme kiti**, geliştiricilerin belirli işletim sistemleri için uygulamalar oluşturmak için kullandıkları indirilebilir bir araç setidir. Genel olarak, bir SDK bir uygulama içinde belirli bir modül oluşturmak için ihtiyacınız olan her şeyden oluşur ve kütüphaneler, araçlar, örnek kod, ilgili belgeler ve çoğu zaman API'ler içerebilir.

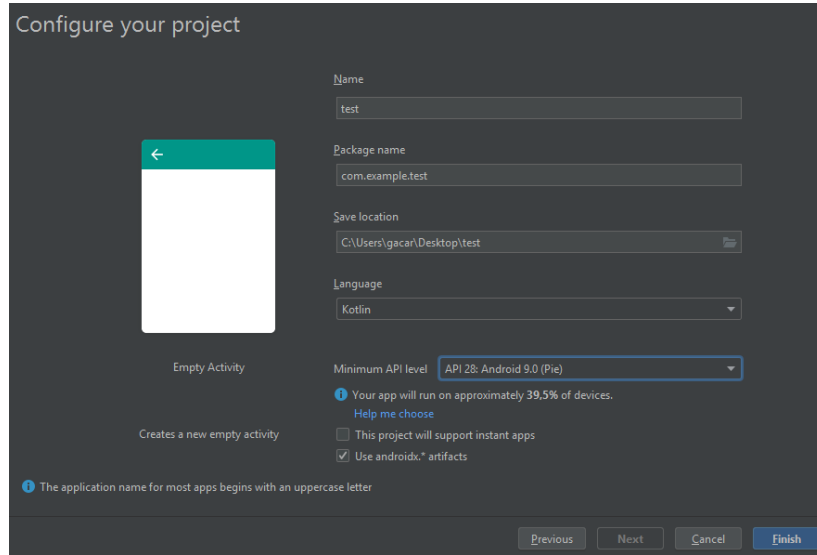
Gelişme

- **Projemi nasıl oluşturdum?**

Android Studio programında File>New>New Project>Empty Activity diyerek ilerliyoruz.



Sonrasında gelen ekranda; proje adımızı, projenin hangi dilde yazılacağını seçiyoruz (kotlin). API level kısmındaysa en düşük sürümü seçiyoruz ve projemiz oluşuyor.



- **Nelere ihtiyaç duydum?**

Projemi oluşturmak için öncelikle Android Studio programına sonrasında bazı kurulumları yapmamız gerekli. Örneğin emülatör kurulumu, JDK, SDK kurulumları...

Benim kullandığım emülatör: Nexus API 28

- **Proje algoritması**

9 kutu ve her kutuda 3 satır, 3 sütun var.

Bir dizi olacak ve bu dizide 1'den 9'a kadar sayılar olacak.

Sudoku üzerinde sırayla boş hücreler seçilecek.

Seçilen hücreyle aynı kutuda olan sayılar ihtimaller dizisinden çıkarılacak.

Sonra aynı satırdaki hücrelere bakılacak ve bu satırdaki sayılar da diziden çıkarılacak. Sonra aynı sütundaki sayılar da diziden çıkarılacak.

- **Butonları ve tabloyu oluşturma**

Öncelikle yukarıda bir kapsayıcı alana ihtiyacımız var bu alanda rakamlar listelenecek. Aşağıda ise 1 2 3 4 5 6 7 8 9 rakamları buton şeklinde listelenecek.

```
<GridLayout
    android:id="@+id/buttonsLayout"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentBottom="true"
    android:layout_centerHorizontal="true"
    android:columnCount="3"
    android:rowCount="3">

<Button
    android:id="@+id/oneButton"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_row="0"
    android:layout_column="0"
    android:text="1" />
```

Görüldüğü üzere bir grid içerisine butonlarımızı attık. Bu butondan toplam 9 tane bulunmakta. Tasarımı ise şöyle görünmekte.



- **Hücreler ve seçim ayarları**

Bu bölümde hücrelerden birisine tıklanıldığında o hücreye bakılacak sonrasında satır ve sütunlara bakılacak arka planında ise bir renk değişmesi olacak.

```

class PlaySudokuActivity : AppCompatActivity(), SudokuBoardView.OnTouchListener {

    private lateinit var viewModel: PlaySudokuViewModel
    private lateinit var numberButtons: List<Button>

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_play_sudoku)

        sudokuBoardView.registerListener(this)

        viewModel = ViewModelProviders.of(this).get(PlaySudokuViewModel::class.java)
        viewModel.sudokuGame.selectedCellLiveData.observe(this, Observer { updateSelectedCellUI(it) })
        viewModel.sudokuGame.cellsLiveData.observe(this, Observer { updateCells(it) })
        viewModel.sudokuGame.isTakingNotesLiveData.observe(this, Observer { updateNoteTakingUI(it) })
        viewModel.sudokuGame.highlightedKeysLiveData.observe(this, Observer { updateHighlightedKeys(it) })

        numberButtons = listOf(oneButton, twoButton, threeButton, fourButton, fiveButton, sixButton,
            sevenButton, eightButton, nineButton)

        numberButtons.forEachIndexed { index, button ->
            button.setOnClickListener { viewModel.sudokuGame.handleInput(index + 1) }
        }

        notesButton.setOnClickListener { viewModel.sudokuGame.changeNoteTakingState() }
        deleteButton.setOnClickListener { viewModel.sudokuGame.delete() }
    }

    private fun updateCells(cells: List<Cell>?) = cells?.let {
        sudokuBoardView.updateCells(cells)
    }

    private fun updateSelectedCellUI(cell: Pair<Int, Int>?) = cell?.let {
        sudokuBoardView.updateSelectedCellUI(cell.first, cell.second)
    }

    private fun updateNoteTakingUI(isNoteTaking: Boolean?) = isNoteTaking?.let {
        val color = if (it) ContextCompat.getColor(this, R.color.colorPrimary) else Color.LTGRAY
        notesButton.background.setColorFilter(color, PorterDuff.Mode.MULTIPLY)
    }
}

```

- **Kullanıcı input ayarları**

Bu bölümde kullanıcılar rakamları değiştirebilecek. Sonrasında aşağıdaki butonlara basarak yukarıda rakamlar gözükecek.

```

class SudokuBoardView(context: Context, attributeSet: AttributeSet) : View(context, attributeSet) {

    private var sqrtSize = 3
    private var size = 9

    // these are set in onDraw
    private var cellSizePixels = 0F
    private var noteSizePixels = 0F

    private var selectedRow = 0
    private var selectedCol = 0
}

```

```
private var listener: SudokuBoardView.OnTouchListener? = null

private var cells: List<Cell>? = null

private val thickLinePaint = Paint().apply {
    style = Paint.Style.STROKE
    color = Color.BLACK
    strokeWidth = 4F
}

private val thinLinePaint = Paint().apply {
    style = Paint.Style.STROKE
    color = Color.BLACK
    strokeWidth = 2F
}

private val selectedCellPaint = Paint().apply {
    style = Paint.Style.FILL_AND_STROKE
    color = Color.parseColor("#6ead3a")
}

private val conflictingCellPaint = Paint().apply {
    style = Paint.Style.FILL_AND_STROKE
    color = Color.parseColor("#efedef")
}

private val textPaint = Paint().apply {
    style = Paint.Style.FILL_AND_STROKE
    color = Color.BLACK
}

private val startingCellTextPaint = Paint().apply {
    style = Paint.Style.FILL_AND_STROKE
    color = Color.BLACK
    typeface = Typeface.DEFAULT_BOLD
}

private val noteTextPaint = Paint().apply {
    style = Paint.Style.FILL_AND_STROKE
    color = Color.BLACK
}

private val startingCellPaint = Paint().apply {
    style = Paint.Style.FILL_AND_STROKE
    color = Color.parseColor("#acacac")
}

override fun onMeasure(widthMeasureSpec: Int, heightMeasureSpec: Int) {
    super.onMeasure(widthMeasureSpec, heightMeasureSpec)
    val sizePixels = min(widthMeasureSpec, heightMeasureSpec)
    setMeasuredDimension(sizePixels, sizePixels)
}

override fun onDraw(canvas: Canvas) {
    updateMeasurements(width)
}
```

```

        fillCells(canvas)
        drawLines(canvas)
        drawText(canvas)
    }

    private fun updateMeasurements(width: Int) {
        cellSizePixels = width / size.toFloat()
        noteSizePixels = cellSizePixels / sqrtSize.toFloat()
        noteTextPaint.textSize = cellSizePixels / sqrtSize.toFloat()
        textPaint.textSize = cellSizePixels / 1.5F
        startingCellTextPaint.textSize = cellSizePixels / 1.5F
    }

    private fun fillCells(canvas: Canvas) {
        cells?.forEach {
            val r = it.row
            val c = it.col

            if (it.isStartingCell) {
                fillCell(canvas, r, c, startingCellPaint)
            } else if (r == selectedRow && c == selectedCol) {
                fillCell(canvas, r, c, selectedCellPaint)
            } else if (r == selectedRow || c == selectedCol) {
                fillCell(canvas, r, c, conflictingCellPaint)
            } else if (r / sqrtSize == selectedRow / sqrtSize && c / sqrtSize == selectedCol / sqrtSize) {
                fillCell(canvas, r, c, conflictingCellPaint)
            }
        }
    }
}

```

Sonuç

Android Studio uygulaması kullanılarak Kotlin dilinde Sudoku Uygulaması tam olarak bitmese de genel hatları oluşturularak yapıldı. Gelen ekranda üst kısımda 9x9 luk satır sütunlar bulunuyor alt kısımda ise yerleştireceğimiz 1-9 arası rakamlar bulunuyor.

Zaman Çizelgesi

Proje başlangıç tarihi: 1 Nisan

Proje bitiş tarihi: 25 Nisan

Toplam süre: 15-20 saat.

Ekler

Tıklanılan sütunu güncelleme kodu

```
fun updateSelectedCell(row: Int, col: Int) {  
    val cell = board.getCell(row, col)  
    if (!cell.isStartingCell) {  
        selectedRow = row  
        selectedCol = col  
        selectedCellLiveData.postValue(Pair(row, col))  
  
        if (isTakingNotes) {  
            highlightedKeysLiveData.postValue(cell.notes)  
        }  
    }  
}
```

Belirli bir sütuna not bırakmak istersek

```
fun changeNoteTakingState() {  
    isTakingNotes = !isTakingNotes  
    isTakingNotesLiveData.postValue(isTakingNotes)  
  
    val curNotes = if (isTakingNotes) {  
        board.getCell(selectedRow, selectedCol).notes  
    } else {  
        setOf<Int>()  
    }  
    highlightedKeysLiveData.postValue(curNotes)  
}
```

Yaptığımız işlemi silmek için ise şu kodları yazmamız gerekiyor.

```
fun delete() {  
    val cell = board.getCell(selectedRow, selectedCol)  
    if (isTakingNotes) {  
        cell.notes.clear()  
        highlightedKeysLiveData.postValue(setOf())  
    } else {  
        cell.value = 0  
    }  
    cellsLiveData.postValue(board.cells)  
}
```

Kaynakça

<https://stackoverflow.com/>

<https://medium.com/>

<https://developer.android.com/studio>

<https://www.youtube.com/user/pj64444>

<https://kotlinlang.org/>