

برنامه نویسی دستگاه های سیار (CE364)

جلسه نهم: حرکت بین صفحات

سجاد شیرعلی شمرضا

پاییز 1401

شنبه، 12 آذر 1401

- بخشهای مرتبط با این جلسه:

- Unit 3: Navigation:
 - Pathway 1: Navigate between screens



سوال؟

مجموعه در زبان کاتلین

مجموعه (Collection)

- گروهی از اشیاء مرتبط
- با ترتیب یا بدون ترتیب
- امکان داشتن عنصر تکراری

مجموعه (Set)

```
val numbers = listOf(0, 3, 8, 4, 0, 5, 5, 8, 9, 2)
val setOfNumbers = numbers.toSet()
println("set:    ${setOfNumbers}")
```

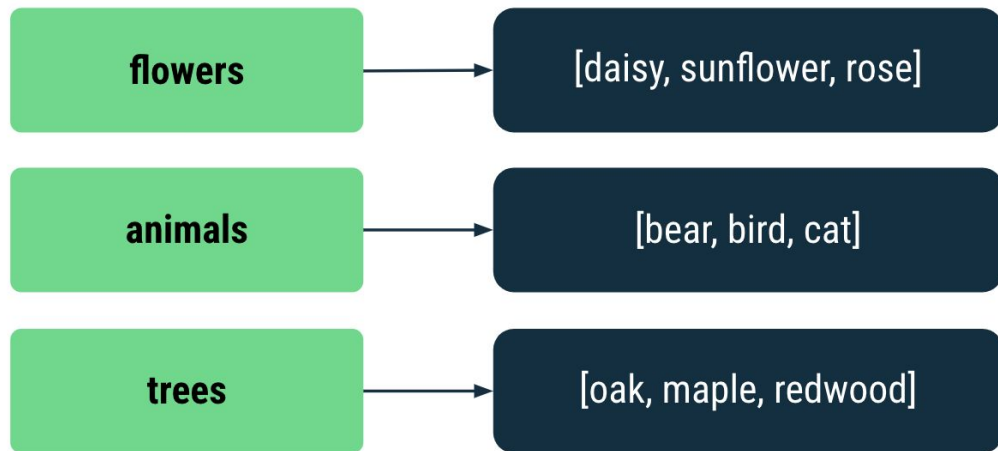
```
val set1 = setOf(1,2,3)
val set2 = mutableSetOf(3,2,1)
println("$set1 == $set2: ${set1 == set2}")

[1, 2, 3] == [3, 2, 1]: true
```

```
println("contains 7: ${setOfNumbers.contains(7)}")
```

- گروهی از اشیاء
- عضو تکراری ندارد
- ترتیبی بین اعضا وجود ندارد
- امکان ادغام و یا اشتراک گیری

نگاشت



- مجموعه ای از کلید-مقدار
- کلید ها غیر تکراری هستند
- نگاشت هر کلید به یک مقدار

```
val peopleAges = mutableMapOf<String, Int>(  
    "Fred" to 30,  
    "Ann" to 23  
)
```

تغییر یک نگاشت

```
val peopleAges = mutableMapOf<String, Int>(
    "Fred" to 30,
    "Ann" to 23
)
peopleAges.put("Barbara", 42)
peopleAges["Joe"] = 51
peopleAges["Fred"] = 31
```


انجام عمل بر روی تمام اعضای یک مجموعه

- یک راه حل دیگر به غیر از حلقه تکرار for
- استفاده از تابع forEach
- اشاره به عنصر فعلی با it

```
peopleAges.forEach { print("${it.key} is ${it.value}, ") }
```

تبدیل یک مجموعه

- انجام یک عمل بر روی تمام اعضا
- تبدیل هر عنصر به یک عنصر جدید
- تابع map
- عنصر فعلی با it

```
println(peopleAges.map { "${it.key} is ${it.value}" }.joinToString(", ") )
```

انتخاب زیر مجموعه

- انتخاب اعضا با ویژگی خاص
- تبدیل آن به یک مجموعه دیگر

```
val filteredNames = peopleAges.filter { it.key.length < 4 }  
println(filteredNames)
```

نوع "تابع"

- عبارت لمبدا: یک تابع تعریف شده که اسم ندارد و بلافاصله استفاده می شود
- امکان ذخیره یک تابع در متغیر و همچنین برگرداندن یک تابع

```
(Int) -> Int
```

```
{ a: Int -> a * 3 }
```

مثال تعریف و ذخیره کردن تابع

```
val number: Int = 5
```

```
val triple: (Int) -> Int = { a: Int -> a * 3 }
```

Function Type

Lambda

```
val triple: (Int) -> Int = { a: Int -> a * 3 }  
println(triple(5))
```

```
val triple: (Int) -> Int = { it * 3 }
```

تابع مقایسه گر برای مرتب سازی

```
val peopleNames = listOf("Fred", "Ann", "Barbara", "Joe")
```

```
peopleNames.sortedWith { str1: String, str2: String -> str1.length - str2.length }
```

```
{ str1: String, str2: String -> str1.length - str2.length }
```

مثال از جلسات قبل

```
calculateButton.setOnClickListener{ calculateTip() }
```

LONG FORM

```
calculateButton.setOnClickListener(object: View.OnClickListener {  
    override fun onClick(view: View?) {  
        calculateTip()  
    }  
})
```

SHORT FORM

```
calculateButton.setOnClickListener { view -> calculateTip() }
```

مثالی دیگر

```
costOfServiceEditText.setOnKeyListener  
{ view, keyCode, event -> handleKeyEvent(view, keyCode) }
```

```
{ view, keyCode, event -> handleKeyEvent(view, keyCode) }
```

```
costOfServiceEditText.setOnKeyListener { view, keyCode, _ -> handleKeyEvent(view, keyCode) }
```


مثال برنامه: چند کلمه با یک حرف خاص

```
val words = listOf("about", "acute", "awesome", "balloon", "best", "brief",  
    "class", "coffee", "creative")  
val filteredWords = words.filter { it.startsWith("b", ignoreCase = true) }  
    .shuffled()  
    .take(2)  
    .sorted()  
println(filteredWords)
```

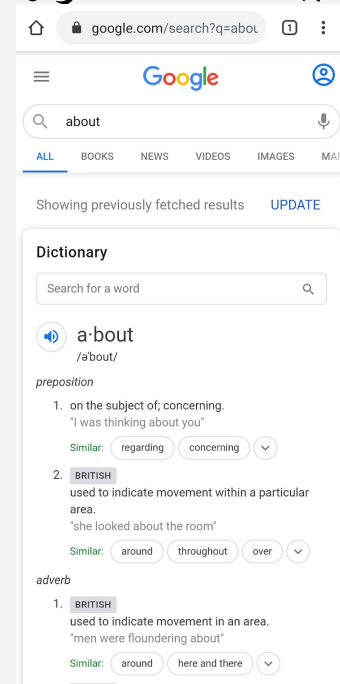
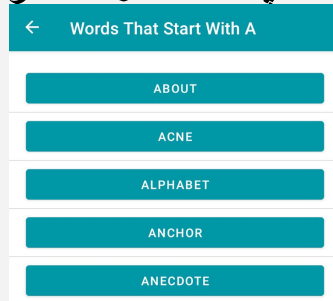
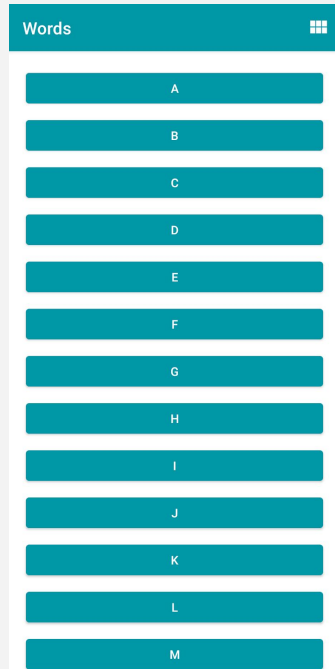


سوال؟

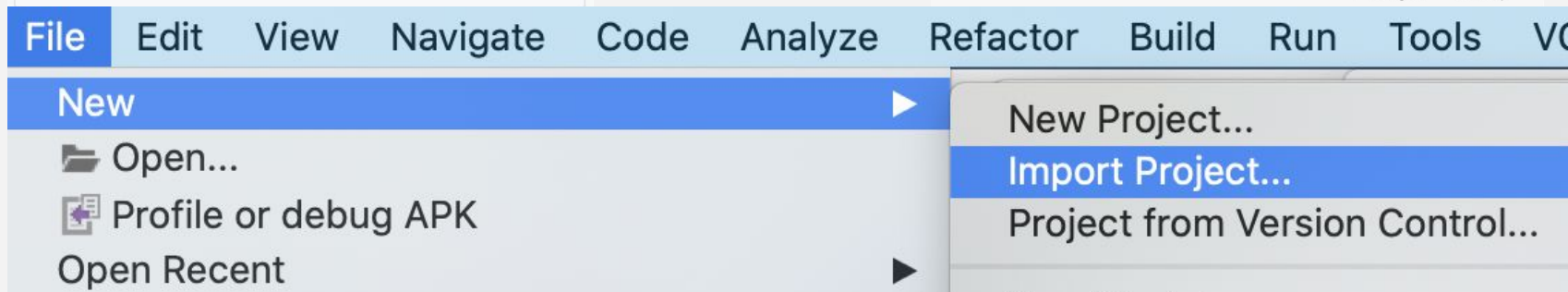
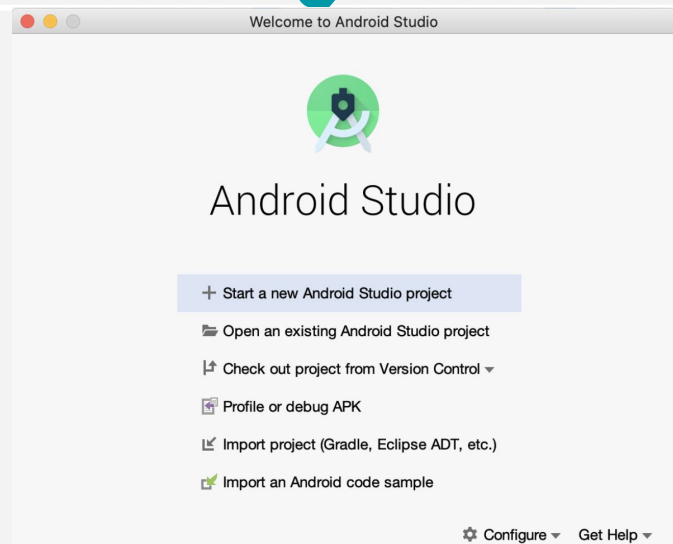
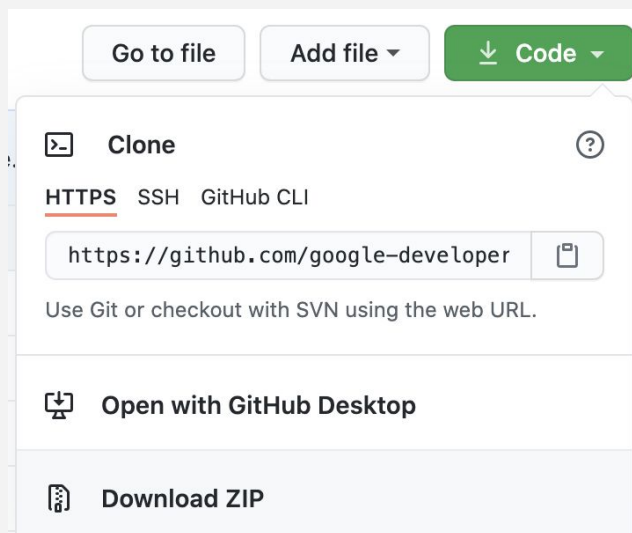
حرکت بین صفحات مختلف

کد شروع برای یک لغتنامه

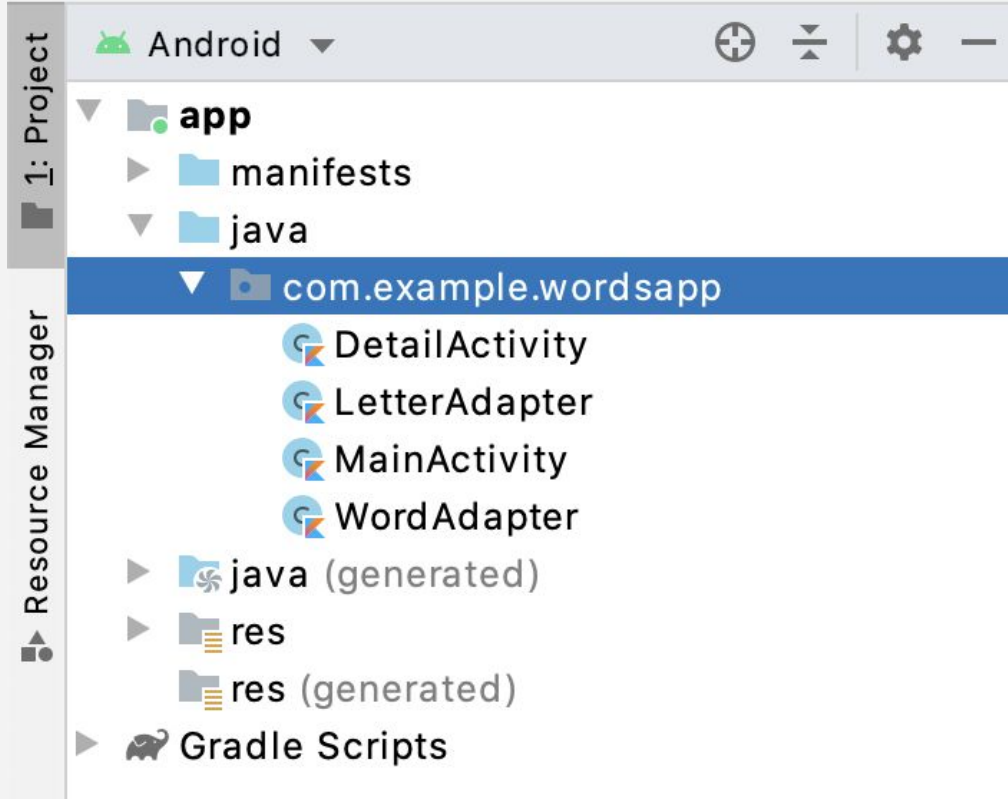
- صفحات مختلف تعریف شده اند
- فقط حرکت بین صفحات قرار است پیاده سازی شود



دریافت و شروع به کار



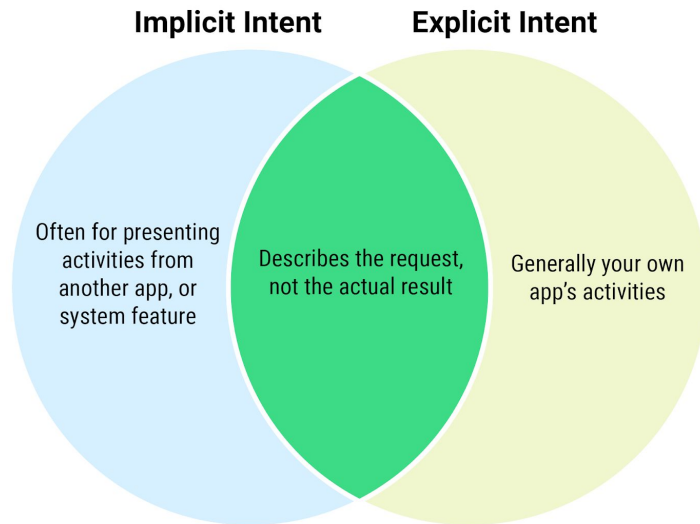
ساختار برنامه



Words			
A	B	C	D
E	F	G	H
I	J	K	L
M	N	O	P
Q	R	S	T
U	V	W	X
Y	Z		

Words			
A			
B			
C			
D			
E			
F			
G			
H			
I			
J			
K			
L			
M			

قصد (Intent)



- یک شیء
- نمایش دهنده کاری که باید انجام شود
- یکی از نمونه های رایج: ایجاد یک فعالیت
- دو نوع: صریح (Explicit) و ضمنی (Implicit)

A

B

C

D

E

F

G

H

I

J

K

L

M

ABOUT

ACNE

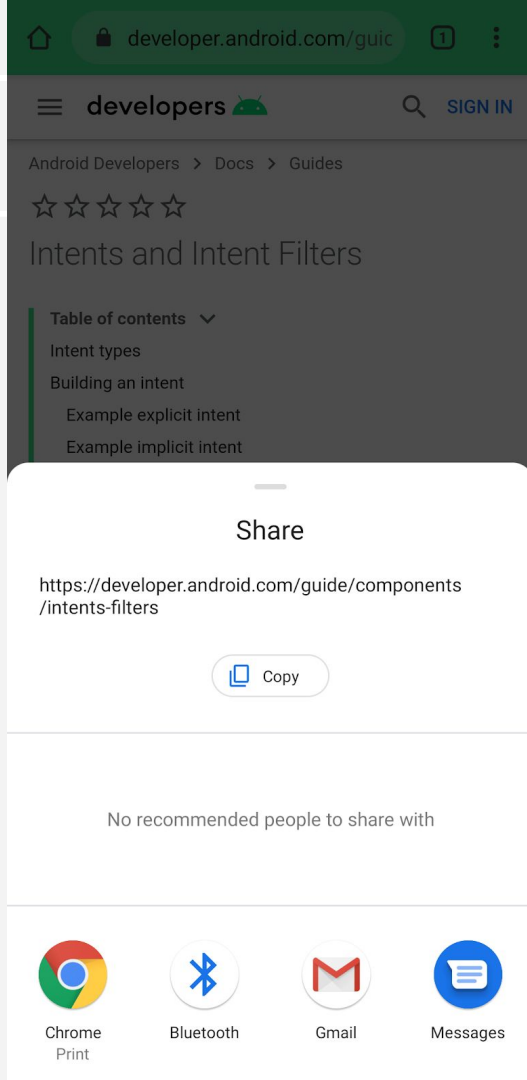
ALPHABET

ANCHOR

ANECDOTE

قصد صریح Explicit Intent

- مشخص است دقیقا چه کاری انجام شود
- مثلا یک فعالیت خاص در برنامه شما



قصد ضمنی Implicit Intent

- درخواست انجام از سیستم
- مثلاً بازکردن یک آدرس

رفتن به صفحات یک حرف

- اضافه کردن OnClickListener برای holder.button در LetterAdapter

```
val context = holder.view.context
```

```
val intent = Intent(context, DetailActivity::class.java)
```

```
intent.putExtra("letter", holder.button.text.toString())
```

```
context.startActivity(intent)
```

پردازش یک قصد در مقصد

- گرفتن اطلاعات در تابع onCreate از DetailActivity

```
val letterId = intent?.extras?.getString("letter").toString()
```

- استفاده از عملگر ?. به خاطر امکان تعریف نشدن مقدار (nullable)

intent?.extra

v extras (from get... Bundle?)

تعریف کلاس همراه (Companion)

```
companion object {  
  
}
```

- فقط یک نمونه از آن ساخته خواهد شد
- به نام یگانه (Singleton) نیز شناخته می شود

```
const val LETTER = "letter"
```

```
intent.putExtra(DetailActivity.LETTER, holder.button.text.toString())
```

```
val letterId = intent?.extras?.getString(LETTER).toString()
```

اضافه کردن یک مقدار دیگر به کلاس همراه

```
companion object {  
    const val LETTER = "letter"  
    const val SEARCH_PREFIX = "https://www.google.com/search?q="
```

```
}
```

ایجاد آدرس برای نمایش

```
val queryUrl: Uri = Uri.parse("${DetailActivity.SEARCH_PREFIX}${item}")
```

URI

Uniform Resource Identifier

URL

Uniform Resource Locator

<https://developer.android.com>
<mailto:someone@gmail.com>

URN

Uniform Resource Name

tel:+1-555-867-5309

ایجاد قصد برای نمایش آدرس

```
val intent = Intent(Intent.ACTION_VIEW, queryUrl)
context.startActivity(intent)
```

- چند نمونه دیگر از قصد ضمنی:
- نمایش یک محل: CATEGORY_APP_MAPS
- ایجاد یک ایمیل: CATEGORY_APP_EMAIL
- نمایش گالری تصاویر: CATEGORY_APP_GALLERY
- تنظیم یک زنگ: ACTION_SET_ALARM
- ایجاد یک تماس: ACTION_DIAL

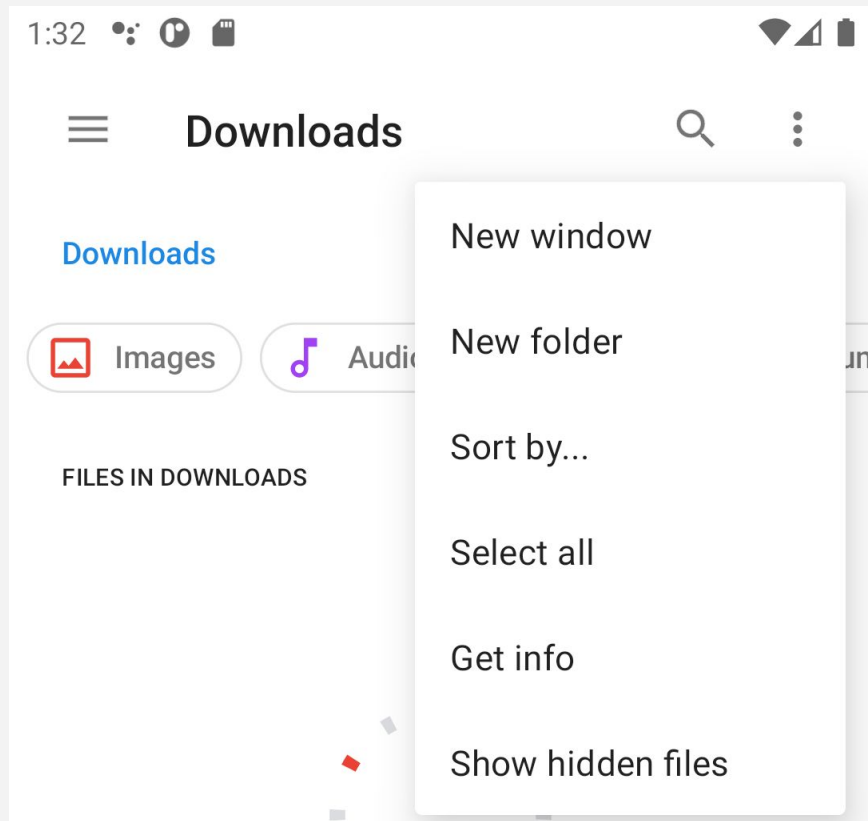


سوال؟

توسعه نوار برنامه

نوار برنامه

- امکان انجام امور مختلف
- نمایش موارد دیگر غیر از نام برنامه





A

B

C

D

E

F

G

H

I

J

K

L

M



A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

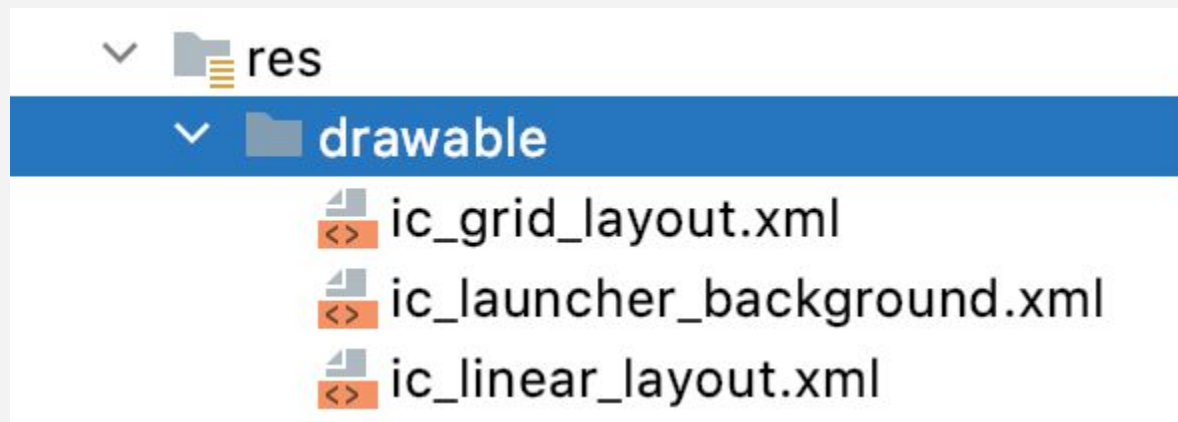
Y

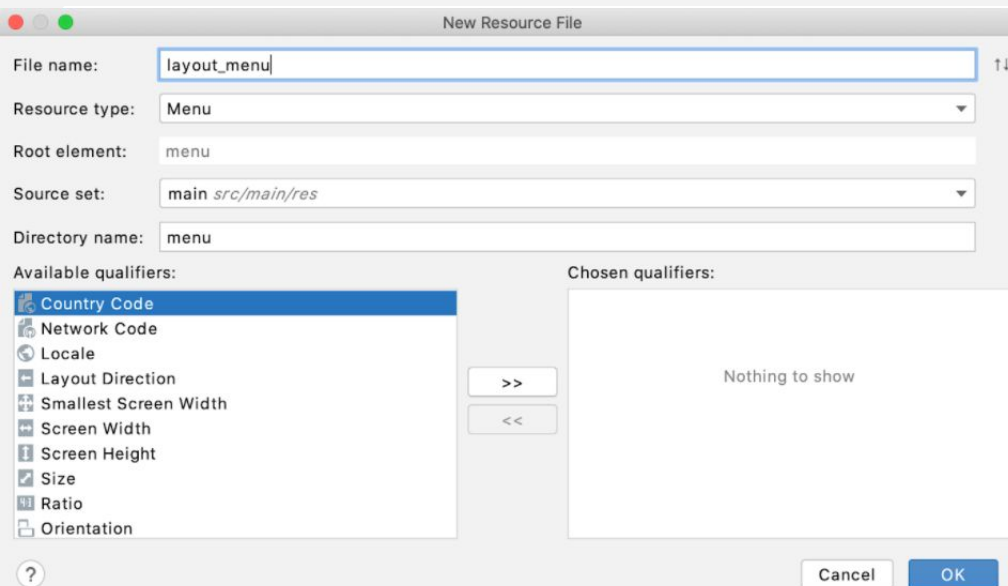
Z

هدف برای این برنامه

- یک دکمه جدید برای تغییر نحوه چیدمان

افزافه کردن آیکن های جدید





تعریف منو برای نوار برنامه

- اضافه کردن در بخش res
res/Menu/layout_menu.xml ○
- تعیین نوع منو ●

```
<menu xmlns:android="http://schemas.android.com/apk/res/android"
      xmlns:app="http://schemas.android.com/apk/res-auto">
  <item android:id="@+id/action_switch_layout"
        android:title="@string/action_switch_layout"
        android:icon="@drawable/ic_linear_layout"
        app:showAsAction="always" />
</menu>
```

پیاده سازی دکمه تغییر ساختار

```
private var isLinearLayoutManager = true
```

```
private fun chooseLayout() {  
    if (isLinearLayoutManager) {  
        recyclerView.layoutManager = LinearLayoutManager(this)  
    } else {  
        recyclerView.layoutManager = GridLayoutManager(this, 4)  
    }  
    recyclerView.adapter = LetterAdapter()  
}
```

تغییر آیکن در نوار برنامه

```
private fun setIcon(menuItem: MenuItem?) {  
    if (menuItem == null)  
        return  
  
    // Set the drawable for the menu icon based on which LayoutManager is currently in use  
  
    // An if-clause can be used on the right side of an assignment if all paths return the same value  
    // The following code is equivalent to  
    // if (isLinearLayoutManager)  
    //     menu.icon = ContextCompat.getDrawable(this, R.drawable.ic_grid_layout)  
    // else menu.icon = ContextCompat.getDrawable(this, R.drawable.ic_linear_layout)  
    menuItem.icon =  
        if (isLinearLayoutManager)  
            ContextCompat.getDrawable(this, R.drawable.ic_grid_layout)  
        else ContextCompat.getDrawable(this, R.drawable.ic_linear_layout)  
}
```

نمایش دکمه تغییر ساختار

```
override fun onCreateOptionsMenu(menu: Menu?): Boolean {  
    menuInflater.inflate(R.menu.layout_menu, menu)  
  
    val layoutButton = menu?.findItem(R.id.action_switch_layout)  
    // Calls code to set the icon based on the LinearLayoutManager of the RecyclerView  
    setIcon(layoutButton)  
  
    return true  
}
```


صدا زدن تابع در صورت فشرده شدن دکمه تغییر ساختار

```
override fun onOptionsItemSelected(item: MenuItem): Boolean {  
    return when (item.itemId) {  
        R.id.action_switch_layout -> {  
            // Sets isLinearLayoutManager (a Boolean) to the opposite value  
            isLinearLayoutManager = !isLinearLayoutManager  
            // Sets layout and icon  
            chooseLayout()  
            setIcon(item)  
  
            return true  
        }  
        // Otherwise, do nothing and use the core event handling  
  
        // when clauses require that all possible paths be accounted for explicitly,  
        // for instance both the true and false cases if the value is a Boolean,  
        // or an else to catch all unhandled cases.  
        else -> super.onOptionsItemSelected(item)  
    }  
}
```



سوال؟

چرخه عمر یک فعالیت

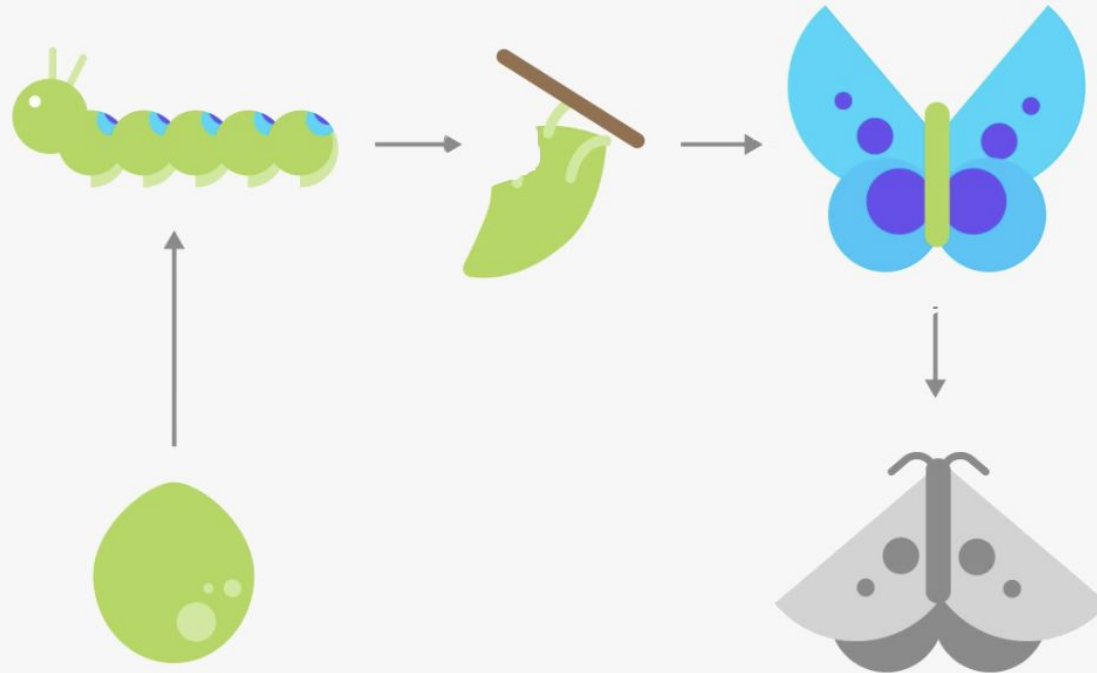


برنامه اولیه مورد استفاده

• خرید دسر

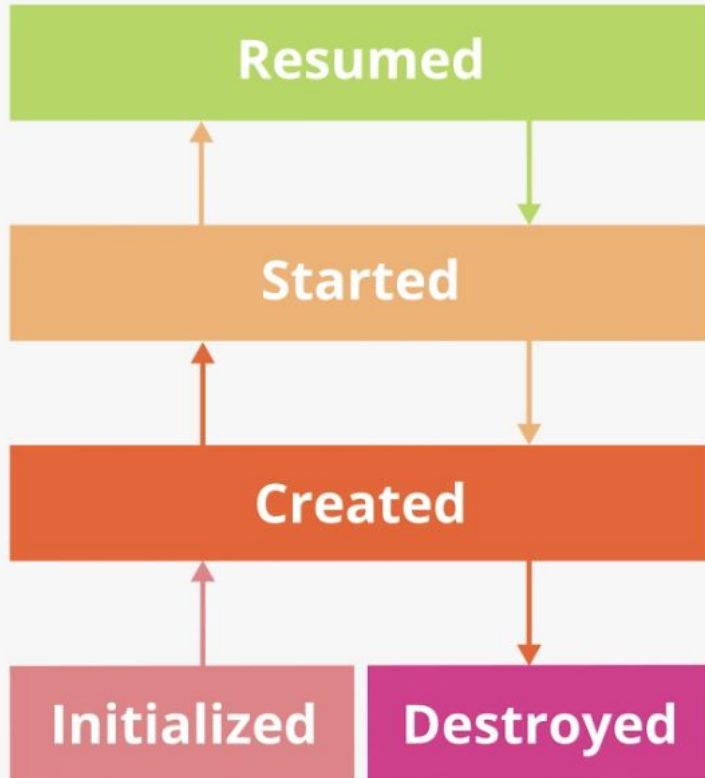
چرخه عمر یک پروانه

Butterfly Lifecycle



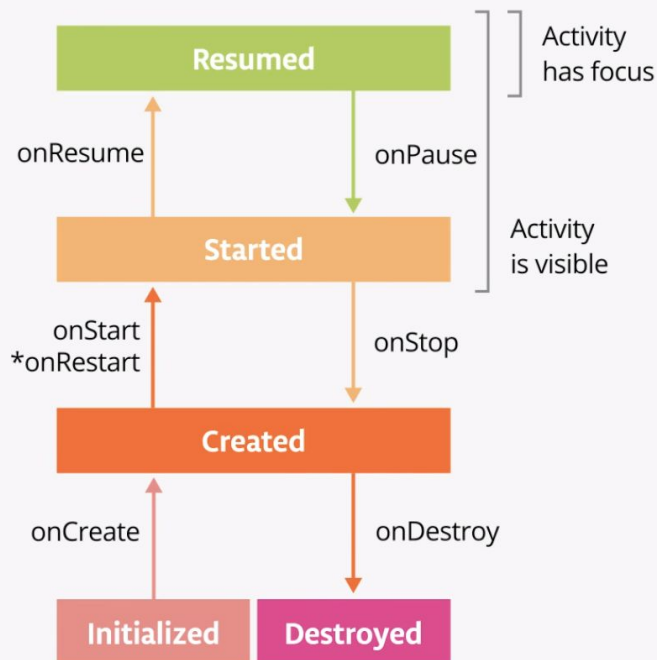
The Activity Lifecycle

چرخه عمر یک فعالیت



چگونگی پردازش تغییرات در حالت فعالیت

The Activity Lifecycle



نوشتن در سیاهه (log)

```
import android.util.Log  
  
Log.d("MainActivity", "onCreate Called")
```

- تعیین اهمیت یک پیغام
 - خطا: Log.e
 - هشدار: Log.w
 - اطلاعات: Log.i
 - عیب یابی: Log.d
 - اضافه: Log.v

استفاده از مقدار ثابت برای برچسب

```
const val TAG = "MainActivity"
```

```
Log.d(TAG, "onCreate Called")
```

مشاهده موارد ثبت شده

The screenshot shows the Android Studio interface. The top toolbar includes icons for Build Variants, Run, and other development tools. The main toolbar at the bottom contains buttons for Run, Logcat, TODO, Terminal, Version Control, Build, and Profiler. The Logcat window is open, displaying a log message from the 'Emulator Pixel_2_API_29' application. The log message is: '2020-10-13 15:50:14.095 30812-30812/com.example.android.dessertclicker D/MainActivity: onCreate Called'. The search filter 'D/MainActivity' is entered in the Logcat search bar. The status bar at the bottom shows 'Gradle build finished in 2 s 666 ms (moments ago)', '2:1 LF', 'UTF-8', 'Git: master', and 'Context: <no context>'.

Build Variants

Run

Logcat

TODO

Terminal

Version Control

Build

Profiler

Event Log

Gradle build finished in 2 s 666 ms (moments ago)

2:1 LF UTF-8 Git: master Context: <no context>

Logcat

Emulator Pixel_2_API_29 An

com.example.android.dessert

Debug

D/MainActivity

Regex

Show only selected applicatio

logcat

2020-10-13 15:50:14.095 30812-30812/com.example.android.dessertclicker D/MainActivity: onCreate Called

Database Inspector

Git

Run

Profiler

Logcat

Build

TODO

Terminal

Device File Explorer

Search for: onStart

m onContentChanged(): Unit
 m onCreateSupportNavigateUpTaskStack(builder: TaskSta
 m onDestroy(): Unit
 m onKeyDown(keyCode: Int, event: KeyEvent!): Boolean
 m onMenuOpened(featureId: Int, menu: Menu!): Boolean
 m onPanelClosed(featureId: Int, menu: Menu!): Unit
 m onPostCreate(savedInstanceState: Bundle?): Unit
 m onPostResume(): Unit
 m onPrepareSupportNavigateUpTaskStack(builder: TaskStackE
 m onSaveInstanceState(outState: Bundle!): Unit
 m **onStart(): Unit**
 m onStop(): Unit
 m onSupportActionModeFinished(mode: ActionMode): Uni
 m onSupportActionMode**Started**(mode: ActionMode): Unit
 m onSupportContentChanged(): Unit
 m onSupportNavigateUp(): Boolean
 m onTitleChanged(title: CharSequence!, color: Int): Unit
 m onWindow**Starting**SupportActionMode(callback: ActionM
 m openOptionsMenu(): Unit
 m setContentView(layoutResID: Int): Unit
 m setContentView(view: View!): Unit
 m setContentView(view: View!, params: ViewGroup.Layout
 m setSupportActionBar(toolbar: Toolbar?): Unit

☐ Copy JavaDoc

Cancel

Select None

OK

تعریف دوباره یک تابع

• نمایش لیست توابع

Code > Override Methods ○

```

override fun onStart() {
    super.onStart()
    Log.d(TAG, "onStart Called")
}
  
```

نمونه تغییر حالت های برنامه

- اجرای برنامه:
- onCreate() -> onStart() -> onResume()
- زدن کلید برگشت به عقب (باعث بسته شدن برنامه می شود):
- onPause() -> onStop() -> onDestroy()
- حالت های دیگری که باعث بسته شدن کامل برنامه میشود:
 - صدا زدن finish
 - بستن اجباری برنامه توسط کاربر (force-quit)
 - عدم استفاده کاربر از برنامه برای مدت طولانی
- بازگشت به برنامه از بین برنامه های اجرا شده:
- onCreate() -> onStart() -> onResume()

نمونه متوقف کردن برنامه

- فشار دادن دکمه خانه هنگام اجرای برنامه
- بازگشت به برنامه
- فشار دادن دکمه به اشتراک گذاری
- بازگشت از منوی به اشتراک گذاری
- onPause() -> onStop()
- onRestart() -> onStart()
- onPause()
- onResume()

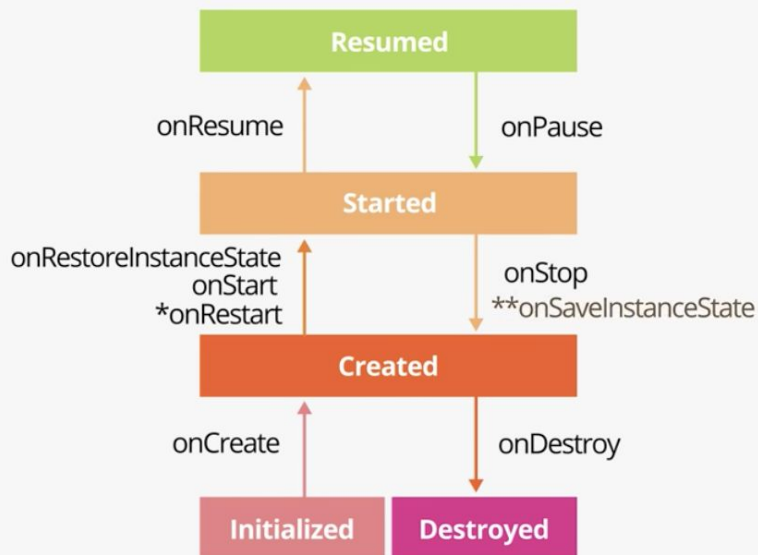
تغییر تنظیمات کلی

- به واسطه تغییرات کلی در ساختار اجرای برنامه
- سیستم برنامه را خاتمه داده و دوباره ایجاد میکند
 - تغییر زبان کل سیستم
 - وصل و یا قطع یک وسیله خارجی مانند صفحه کلید خارجی
 - چرخیدن و تغییر گوشی از افقی به عمودی یا برعکس
- onCreate() -> onStart() -> onResume() -> onPause() -> onStop() -> onDestroy()

ذخیره و بازیابی حالت

- امکان انجام ذخیره حالت قبل از تعلیق برنامه و بازیابی آن

The Activity Lifecycle



نمونه ذخیره حالت

```
override fun onSaveInstanceState(outState: Bundle) {  
    super.onSaveInstanceState(outState)  
  
    Log.d(TAG, "onSaveInstanceState Called")  
}
```

```
const val KEY_REVENUE = "revenue_key"  
const val KEY_DESSERT_SOLD = "dessert_sold_key"
```

```
outState.putInt(KEY_REVENUE, revenue)  
outState.putInt(KEY_DESSERT_SOLD, dessertsSold)
```


نمونه بازیابی حالت

```
override fun onCreate(savedInstanceState: Bundle?) {
```

```
    if (savedInstanceState != null) {  
        revenue = savedInstanceState.getInt(KEY_REVENUE, 0)  
        dessertsSold = savedInstanceState.getInt(KEY_DESSERT_SOLD, 0)  
        showCurrentDessert()  
    }
```



سوال؟