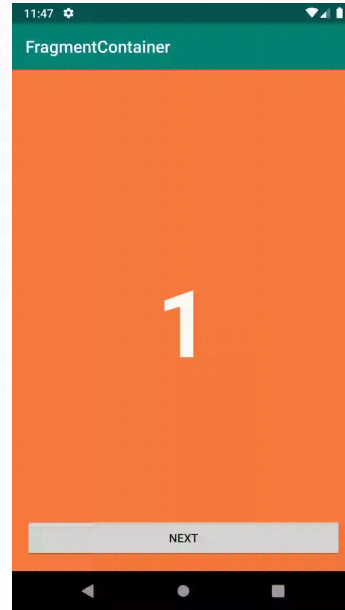


# Aula 21

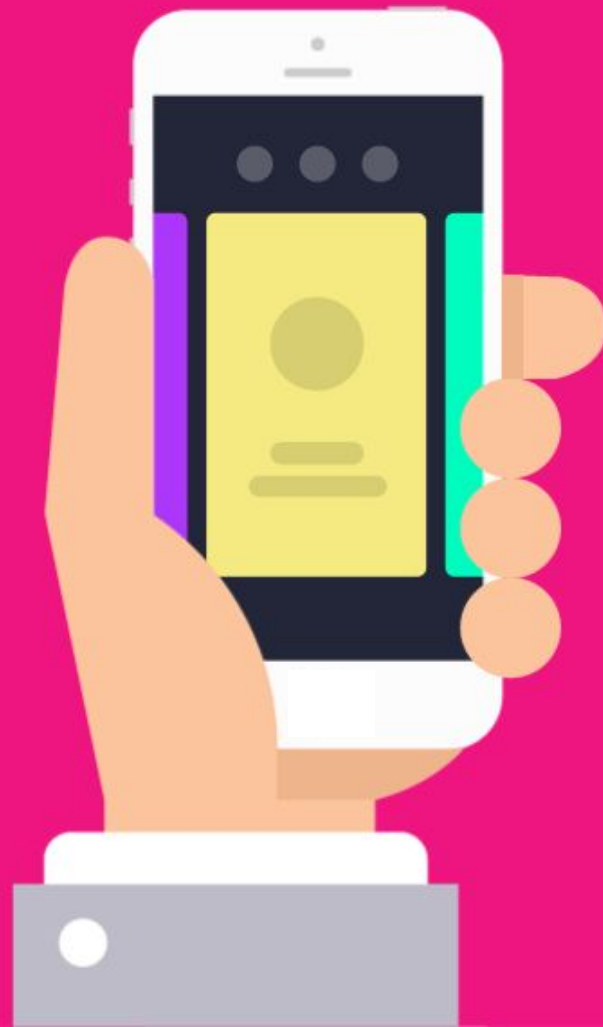
## Fragments

# Fragments

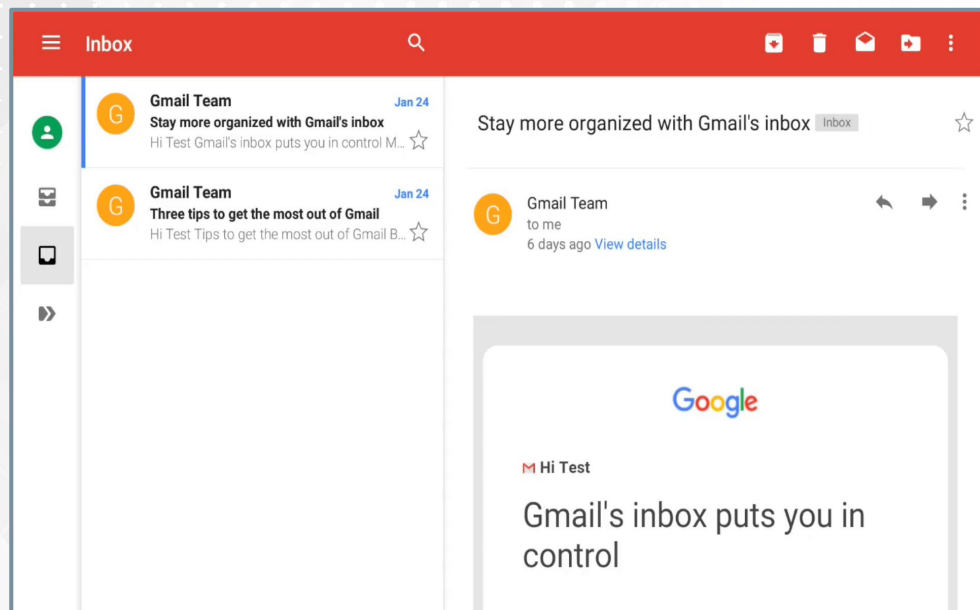


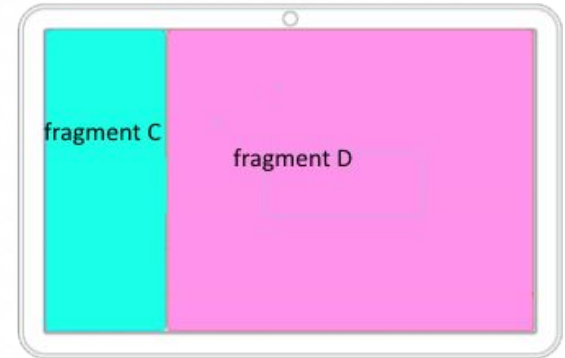
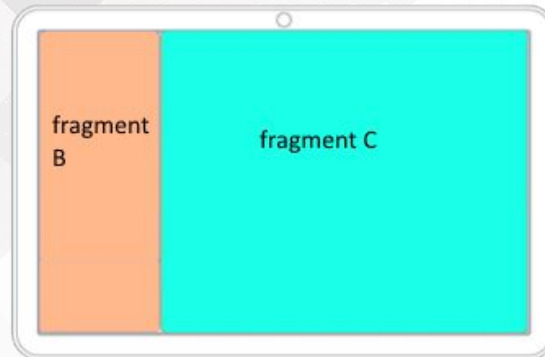
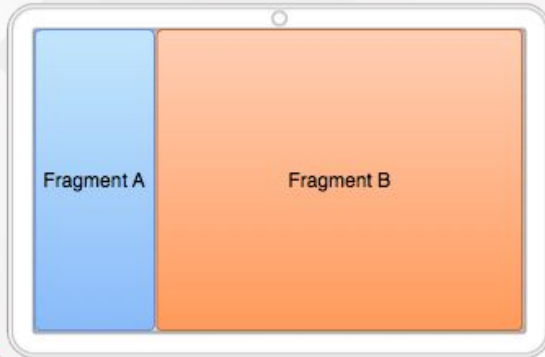
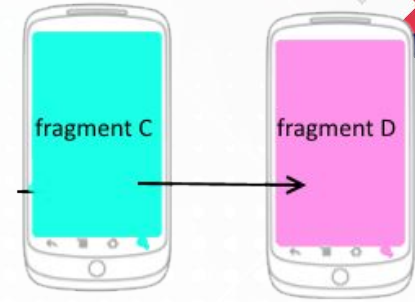
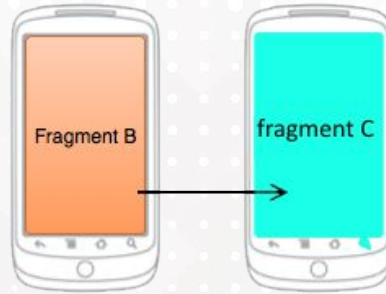
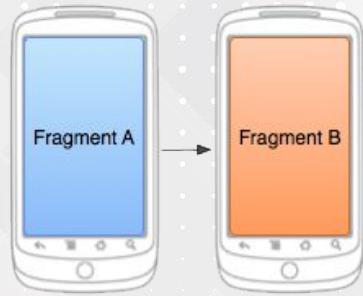
# O que são?

Um Fragment é um “pedaço” de tela que pode ser reutilizado em **outras partes** do seu app.



# Como surgiu a necessidade?



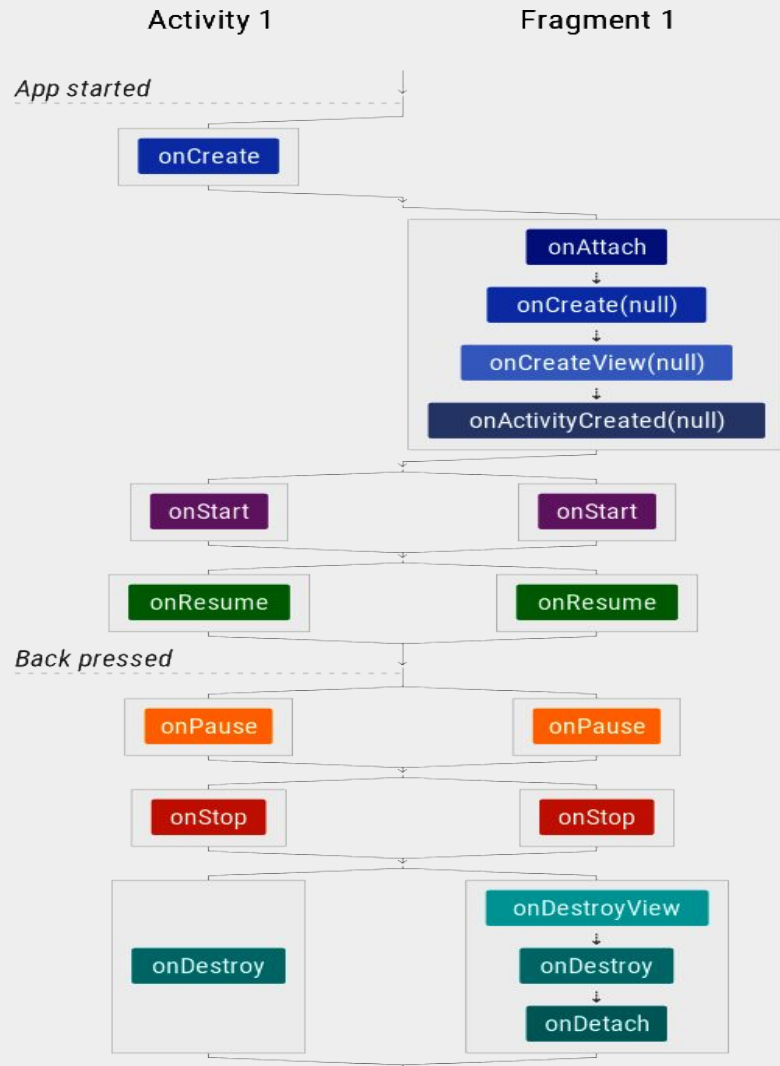


# Vantagens

- Reutilização de telas
- Diminui a duplicação de código
- Permite modularizar uma Activity
- Facilitação de designs dinâmicos e flexíveis para telas grandes

# Ciclo de vida

- Ciclo de vida da **Fragment** depende do ciclo da **Activity**
- O ciclo começa quando a **Activity** está **ativa**
- É obrigatório a implementação do método **onCreateView**. Pois é invocado pelo sistema assim que a tela renderiza pela primeira vez





# Ciclo de vida

**onAttach** - O onAttach é onde podemos obter uma referência para a Activity pai.

**onCreate()** - O sistema o chama ao criar o fragmento.

**onCreateView()** - O onCreateView é onde você constrói ou infla sua interface, faz conexão com alguma fonte de dados e retorna à Activity pai para poder integrá-lo em sua hierarquia de Views.

**onActivityCreated()** - Isso notifica nosso Fragment que a Activity pai completou seu ciclo no onCreate e é aqui que podemos interagir com segurança com a interface de usuário.



# Ciclo de vida

**onStart()** - torna o fragmento visível para o usuário

**onResume()** - faz com que o fragmento comece a interagir com o usuário

**onPause()** - O sistema chama esse método como o primeiro indício de que o usuário está saindo do fragmento (embora não seja sempre uma indicação de que o fragmento está sendo destruído).

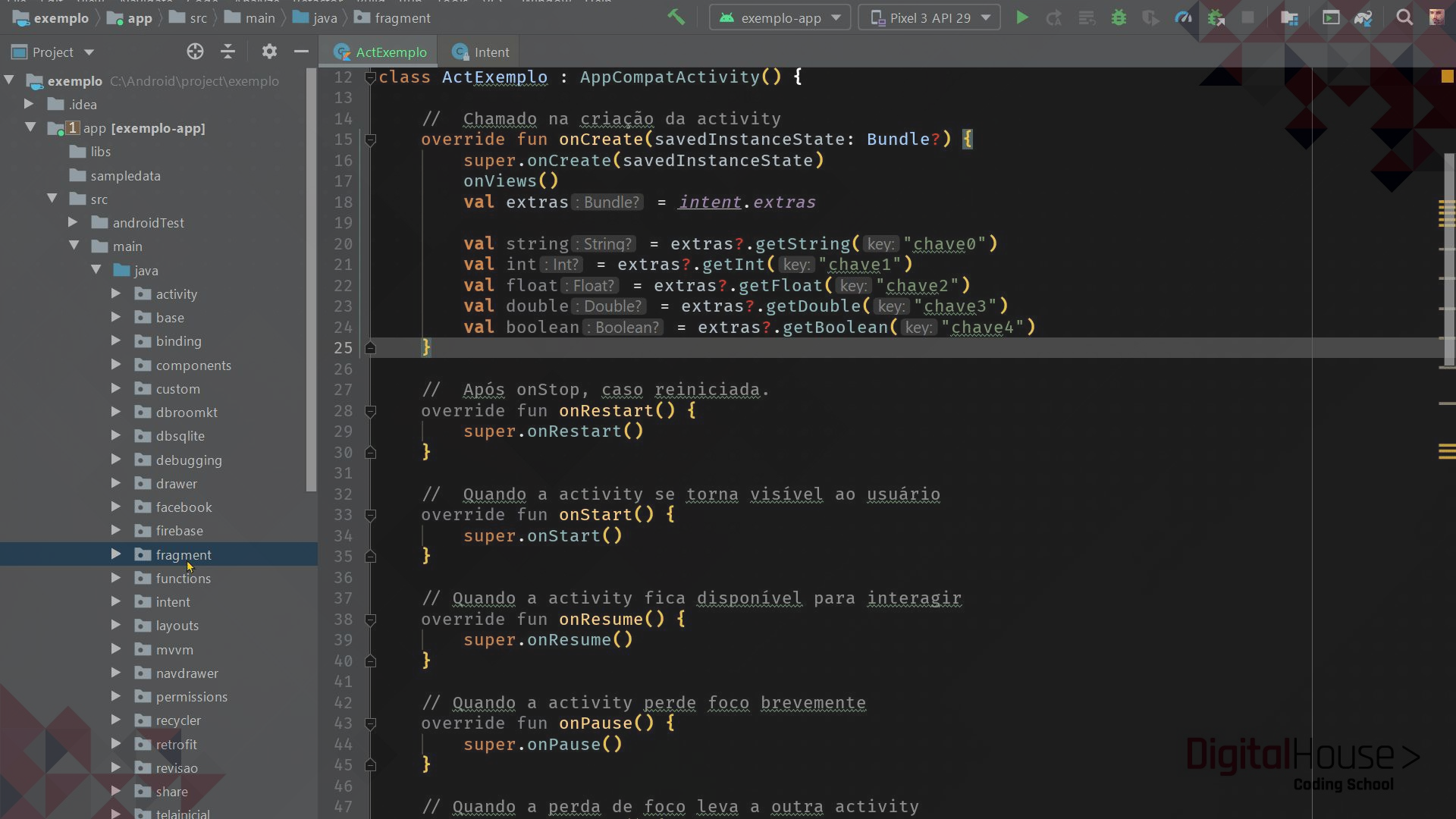
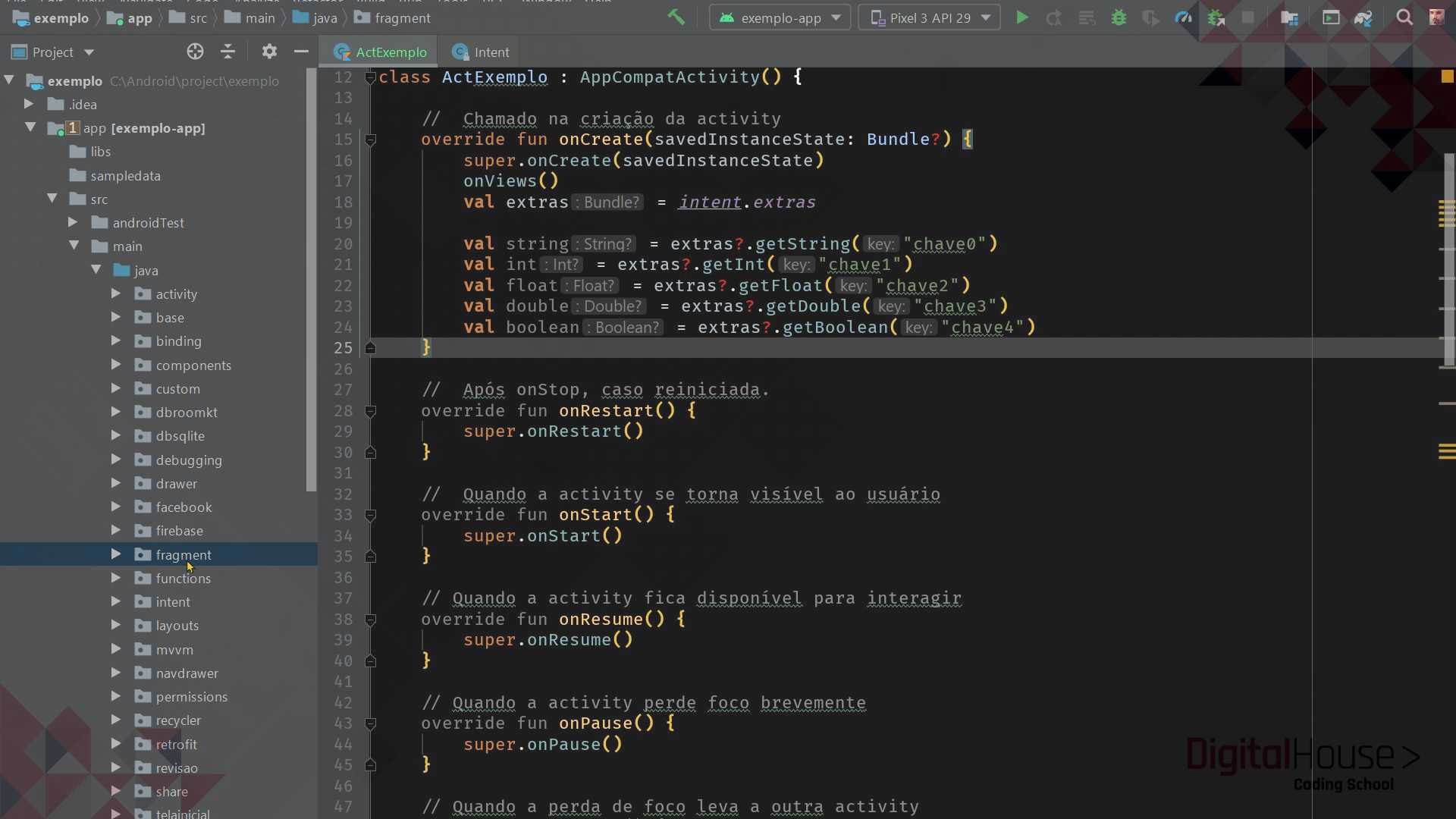
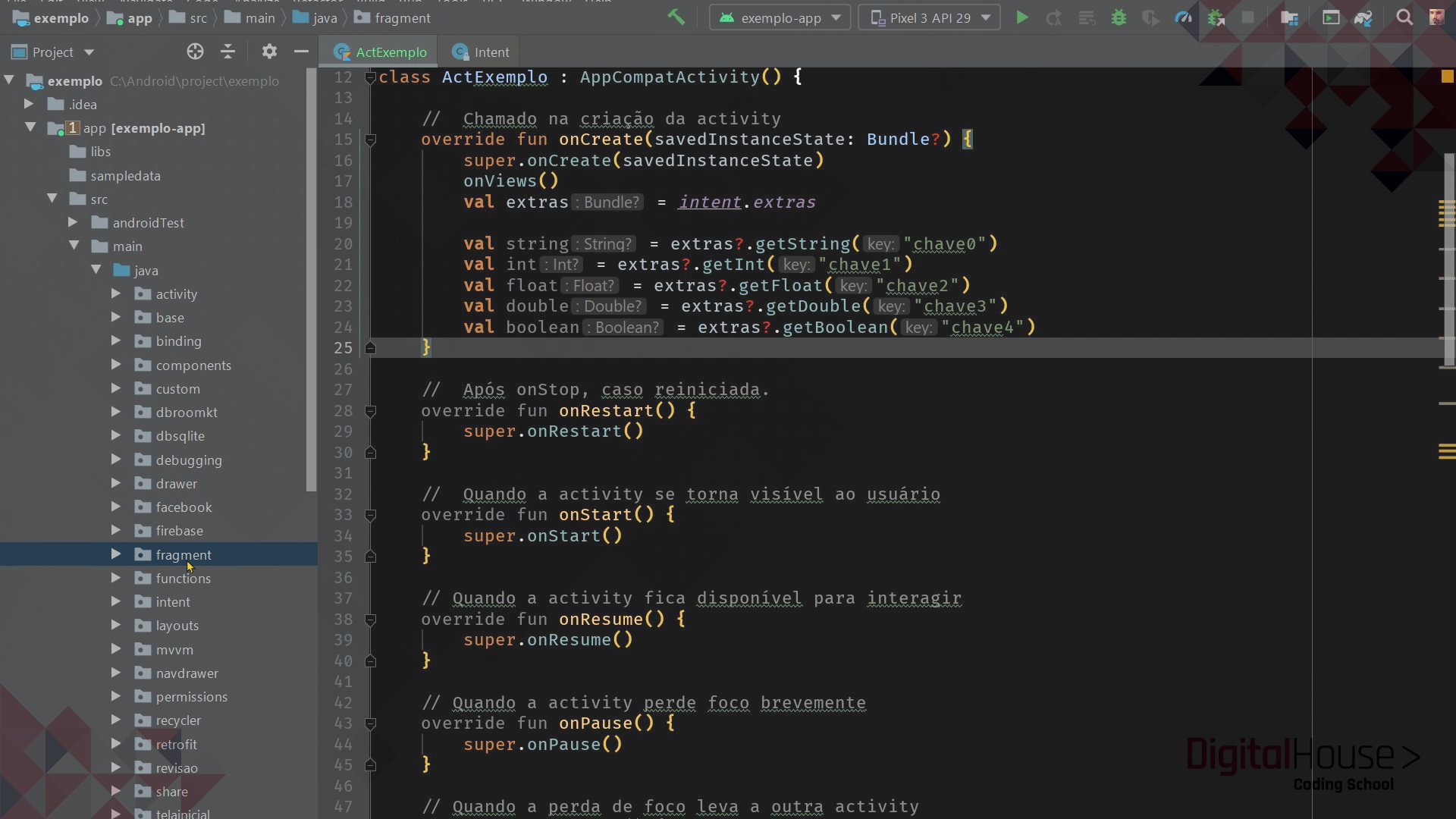
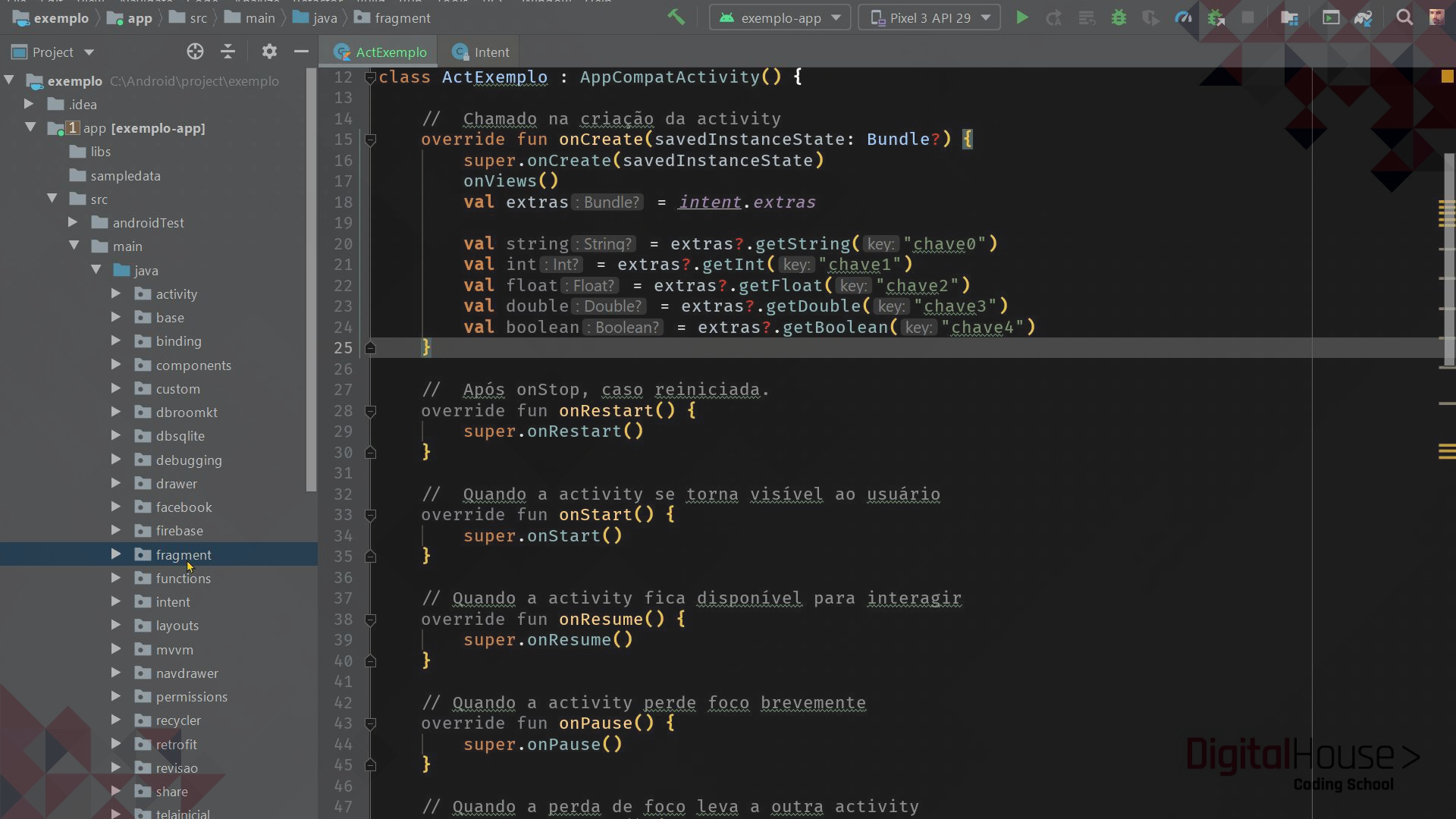
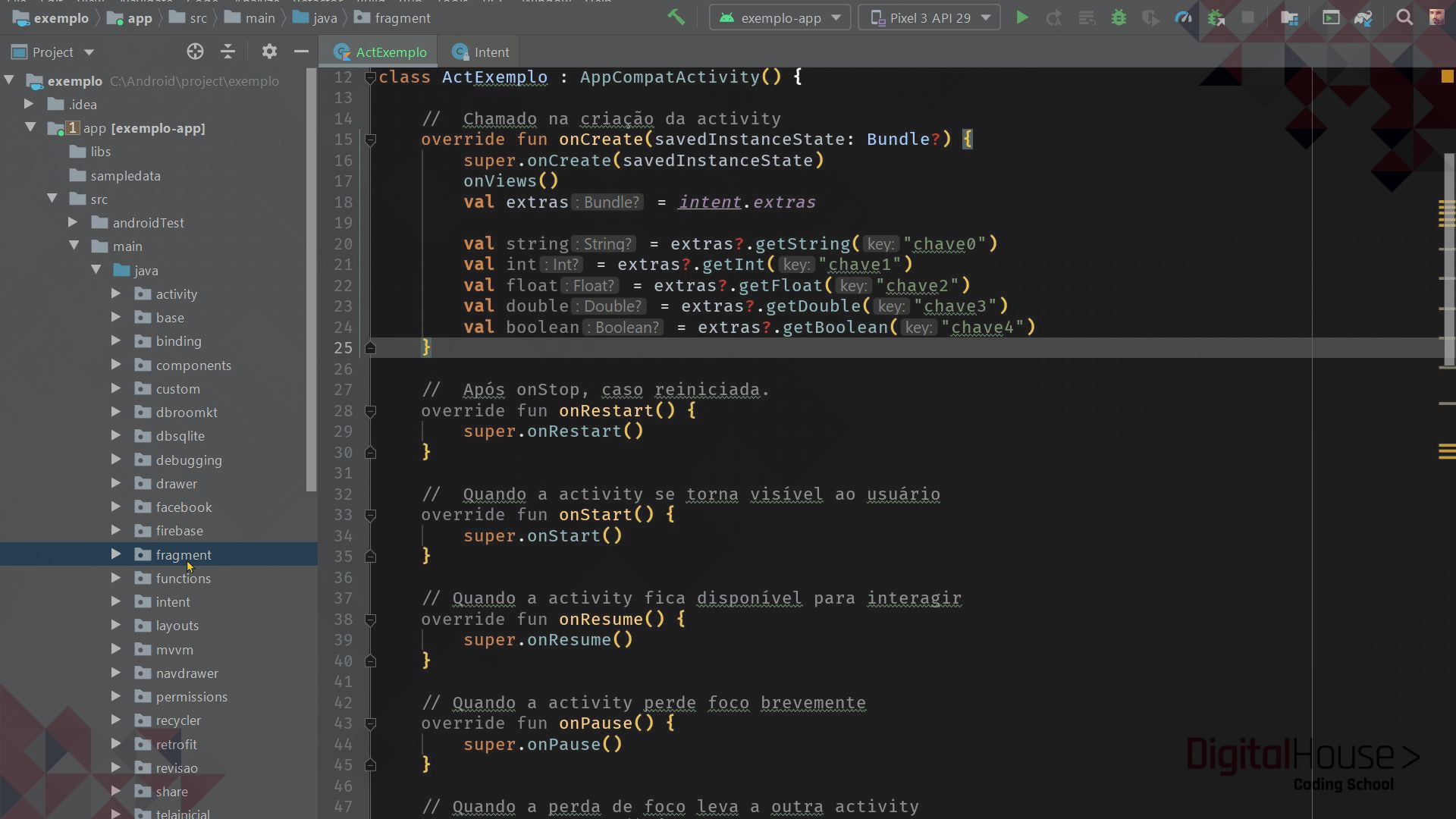
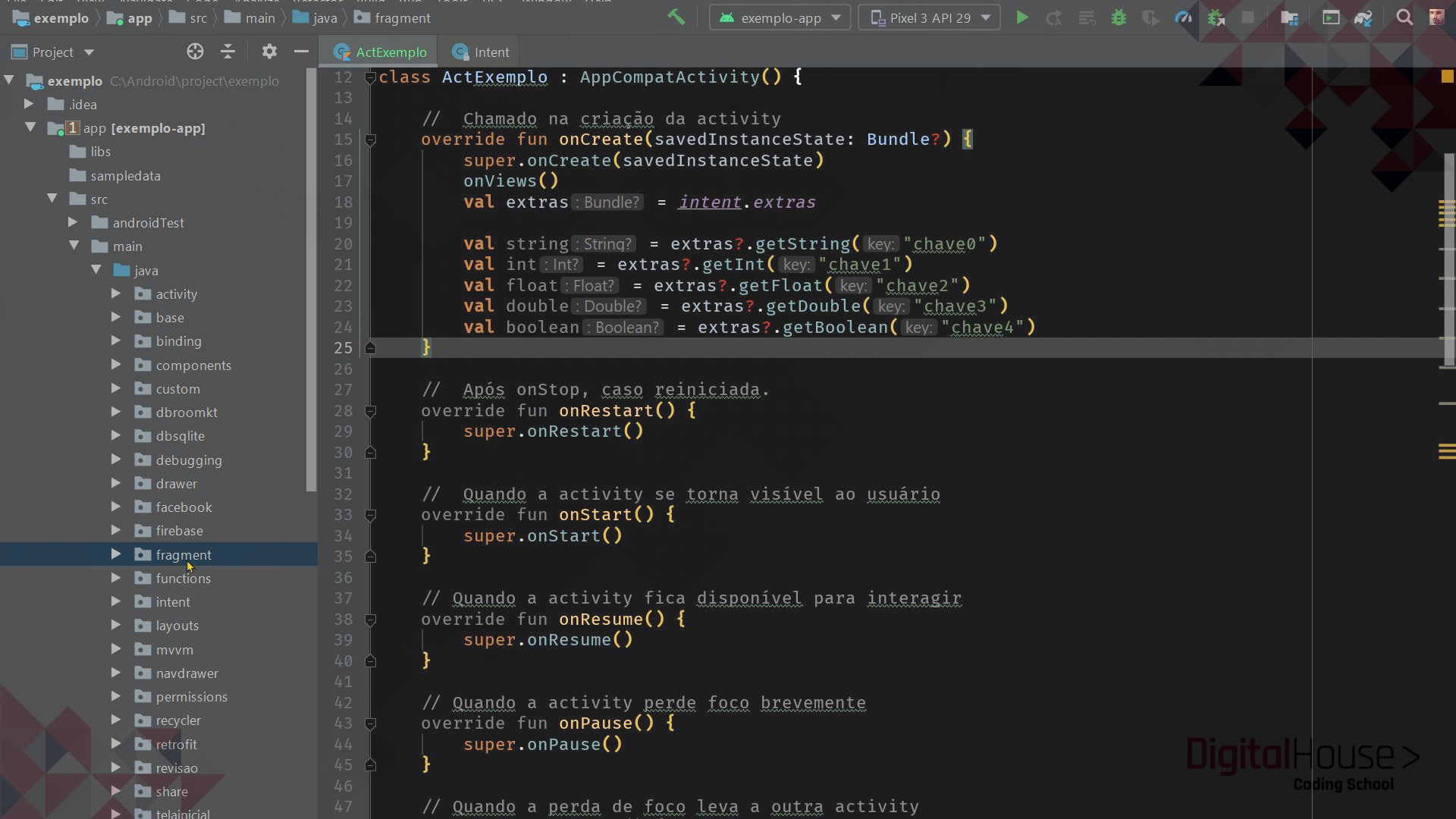
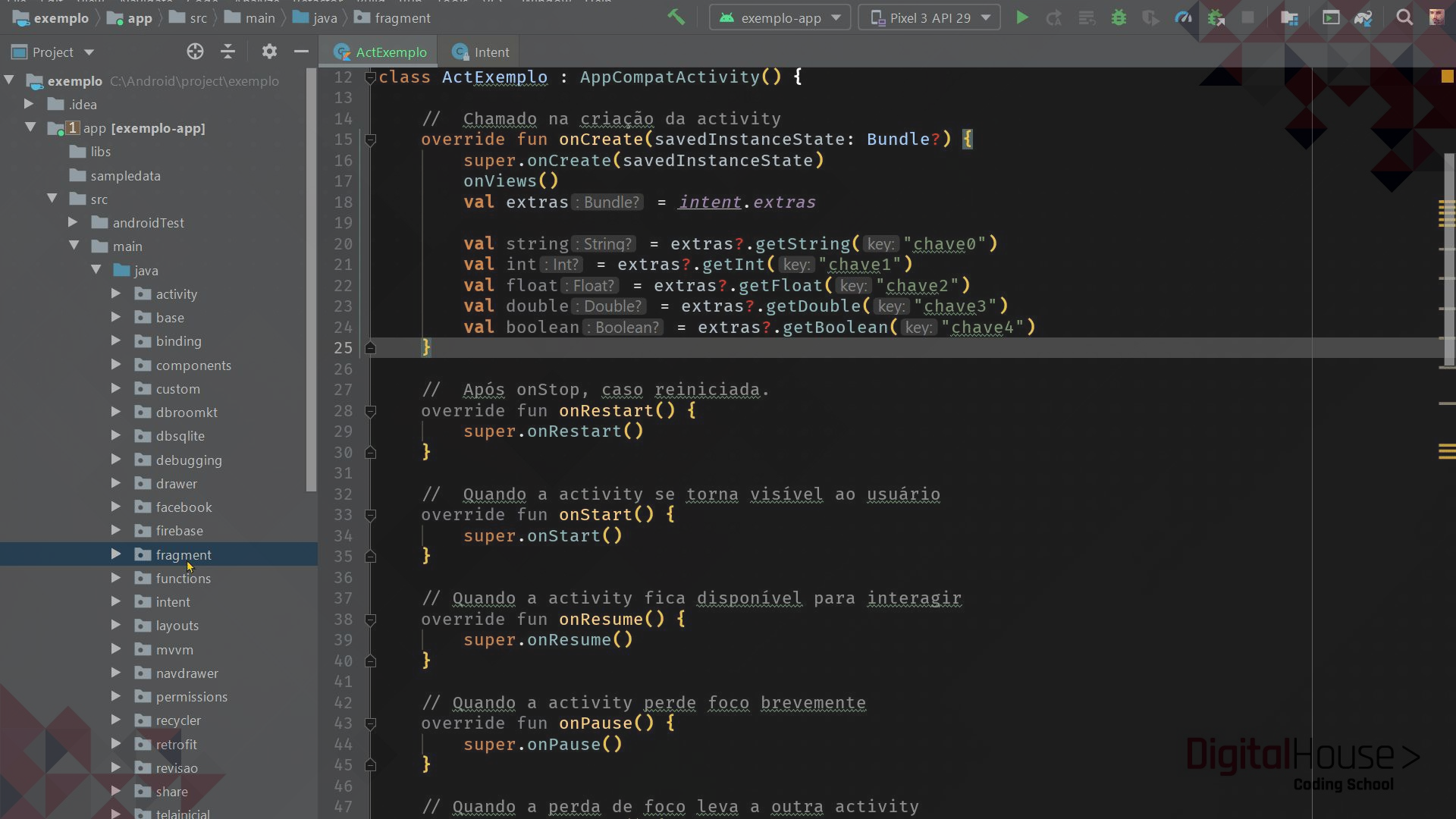
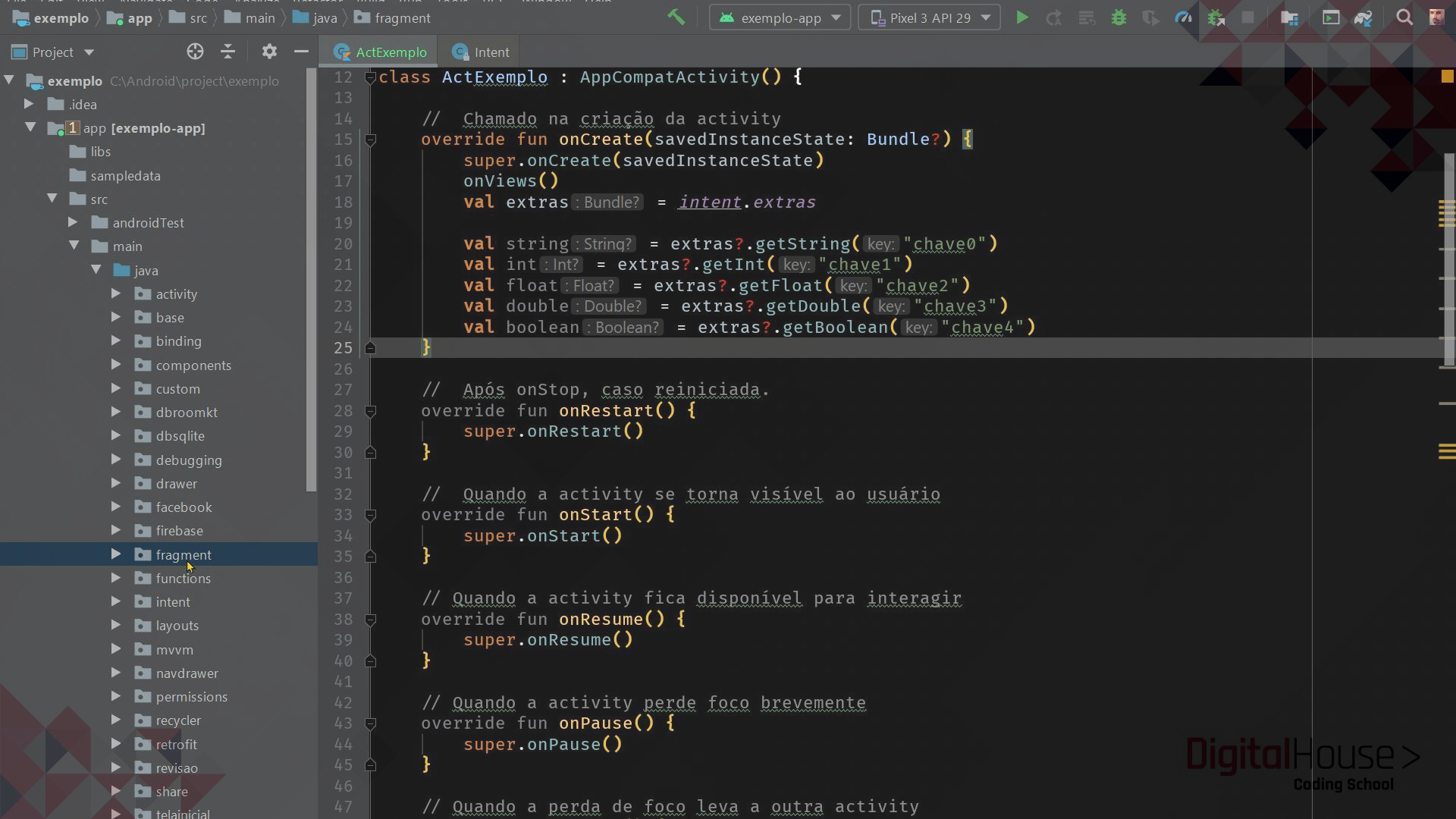
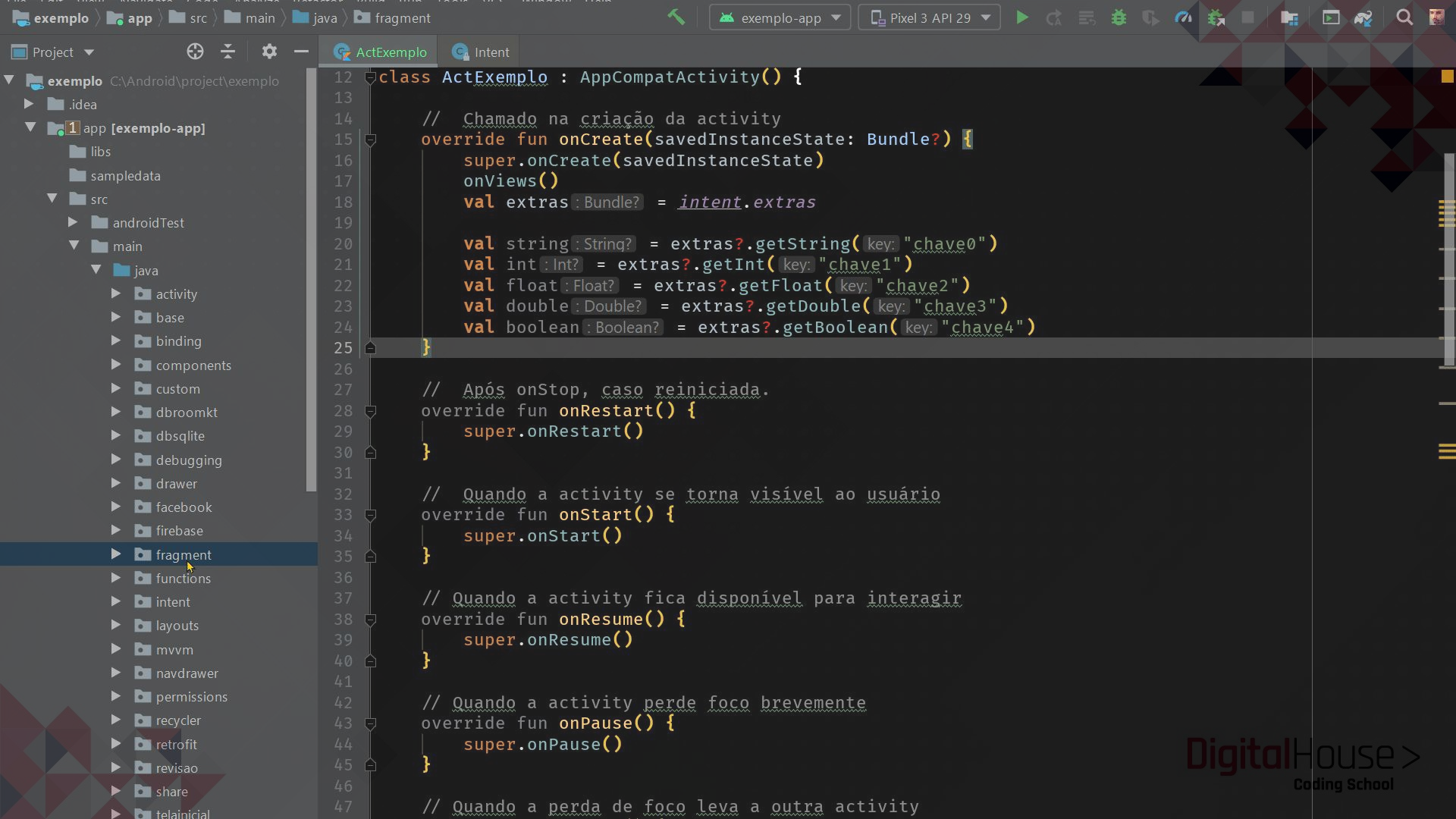
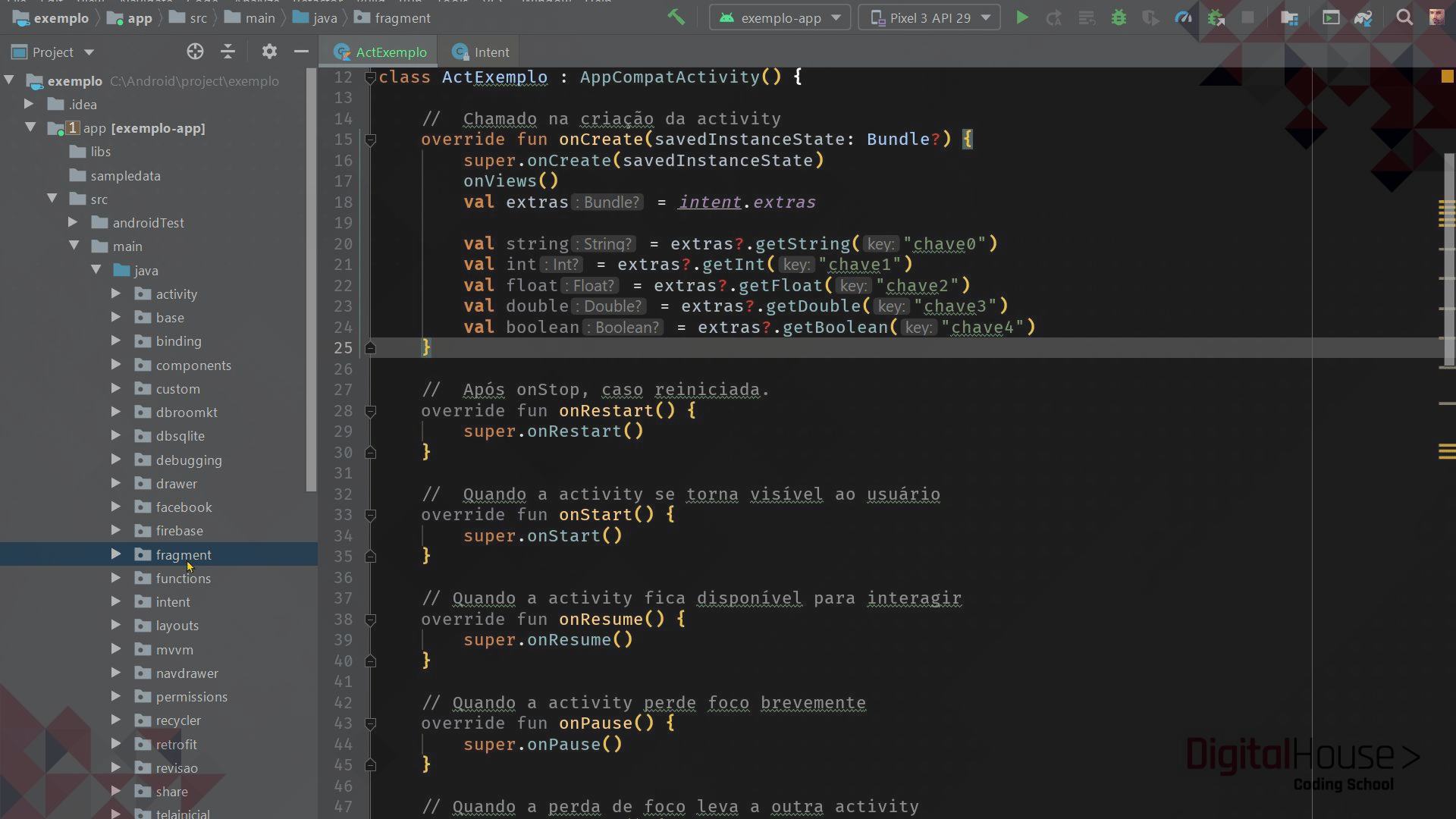
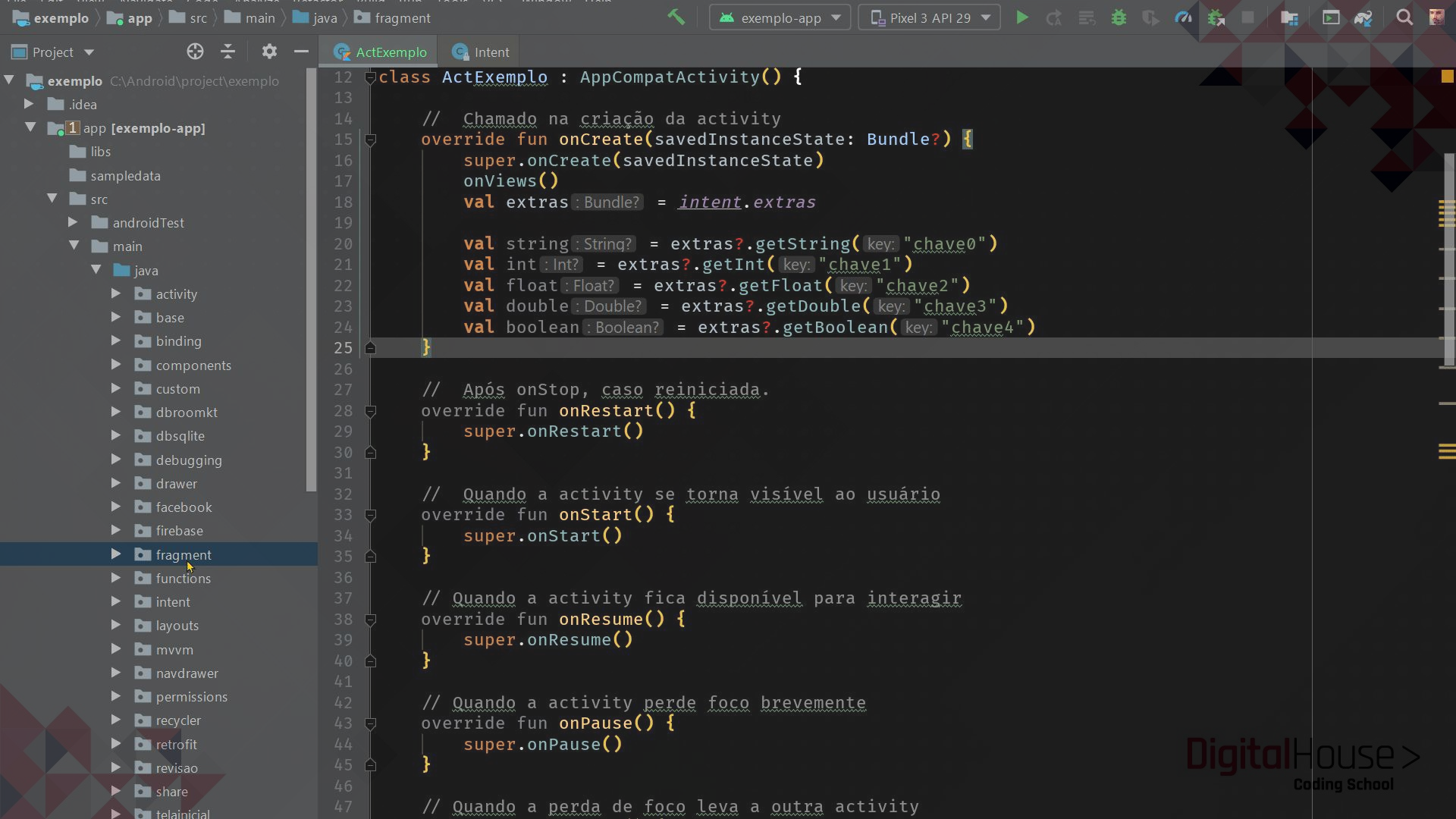
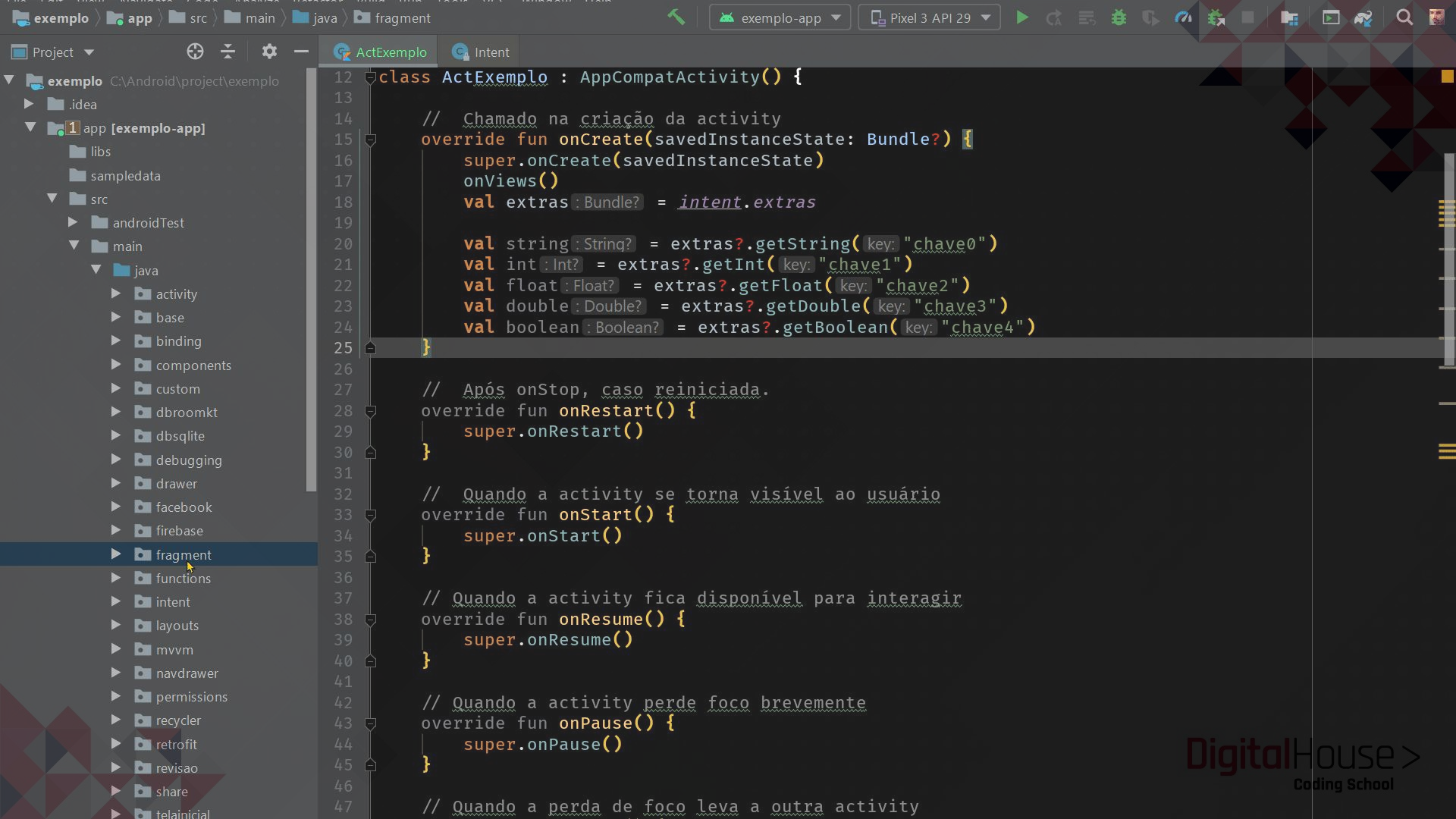
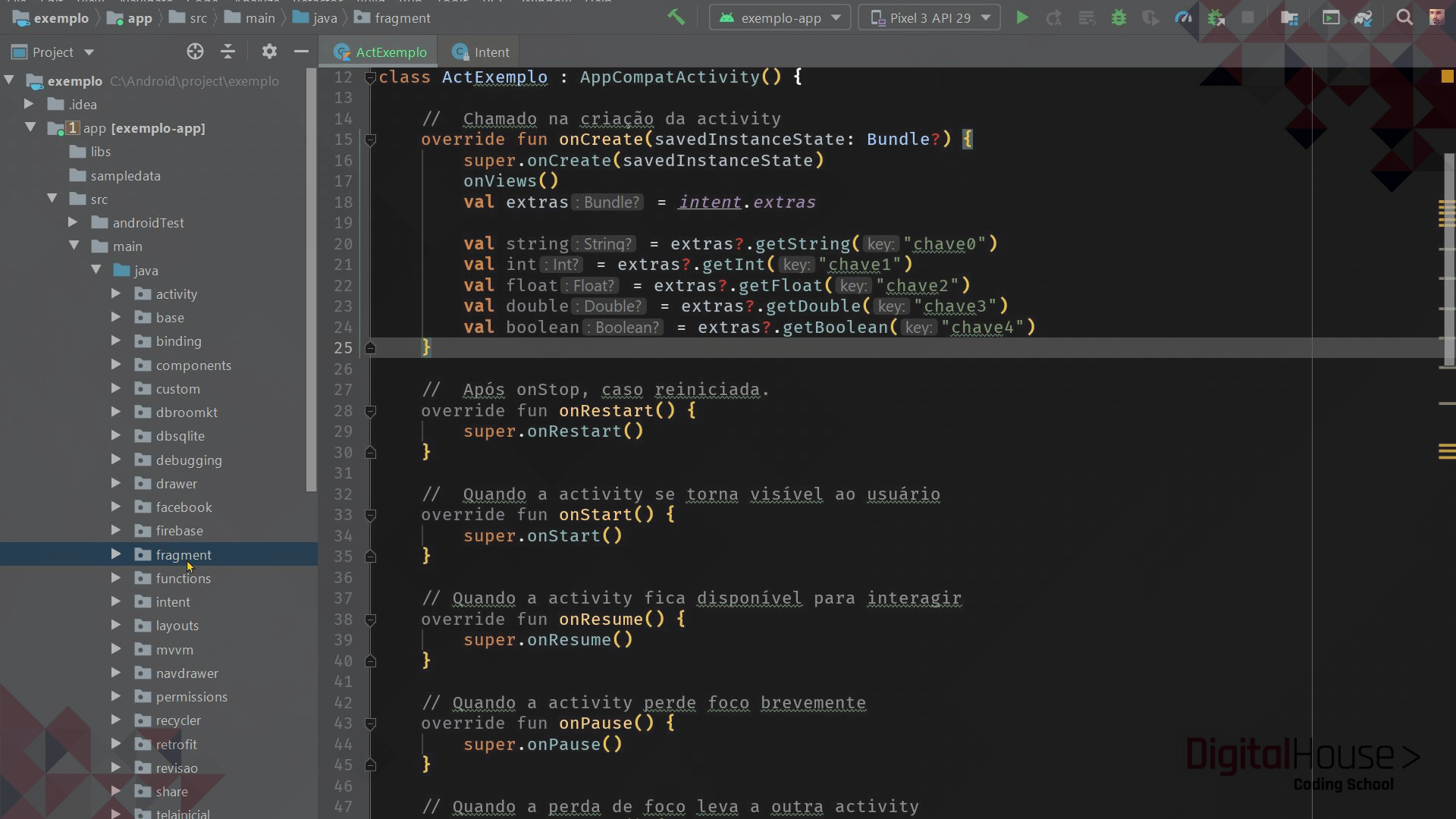
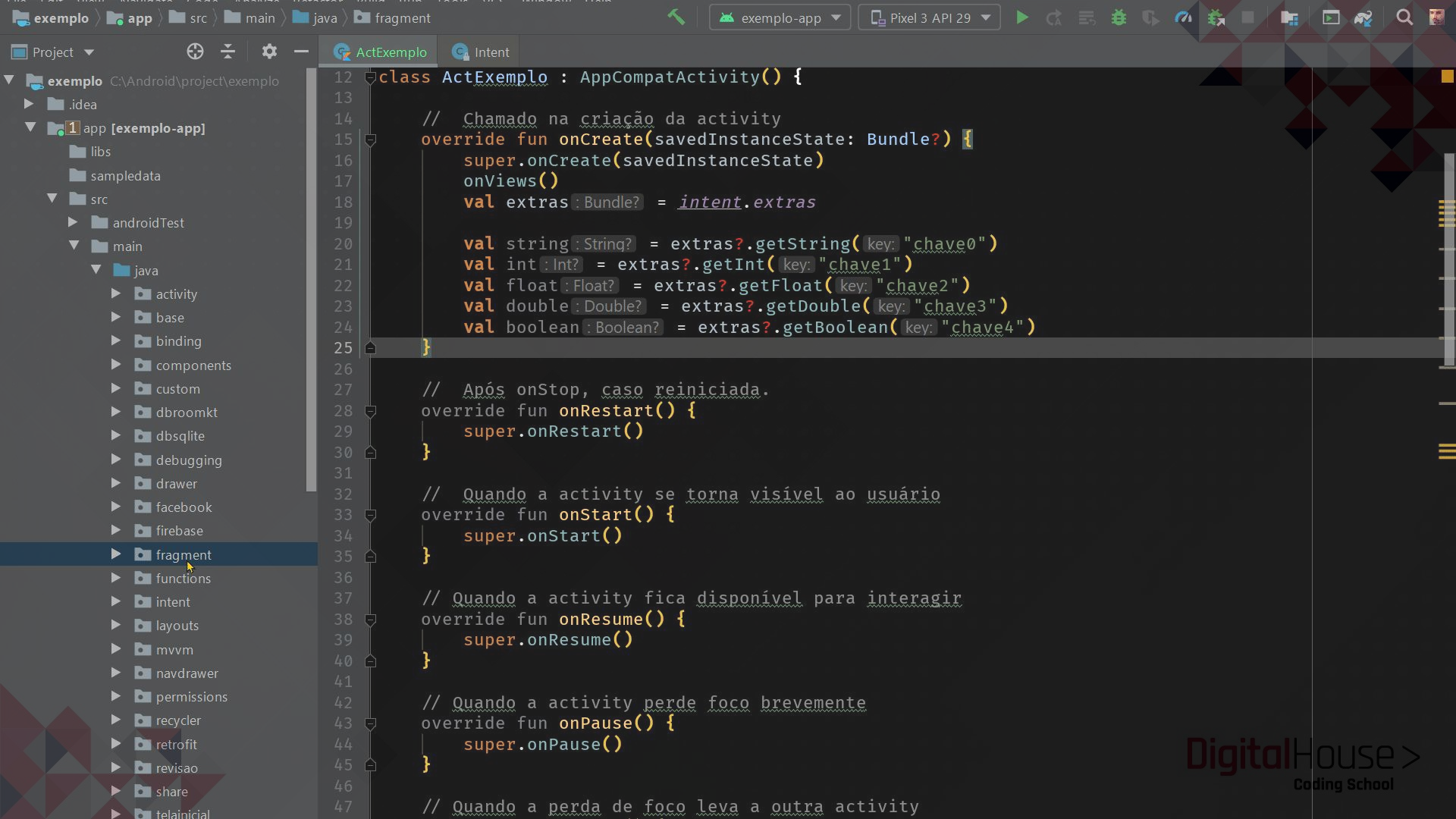
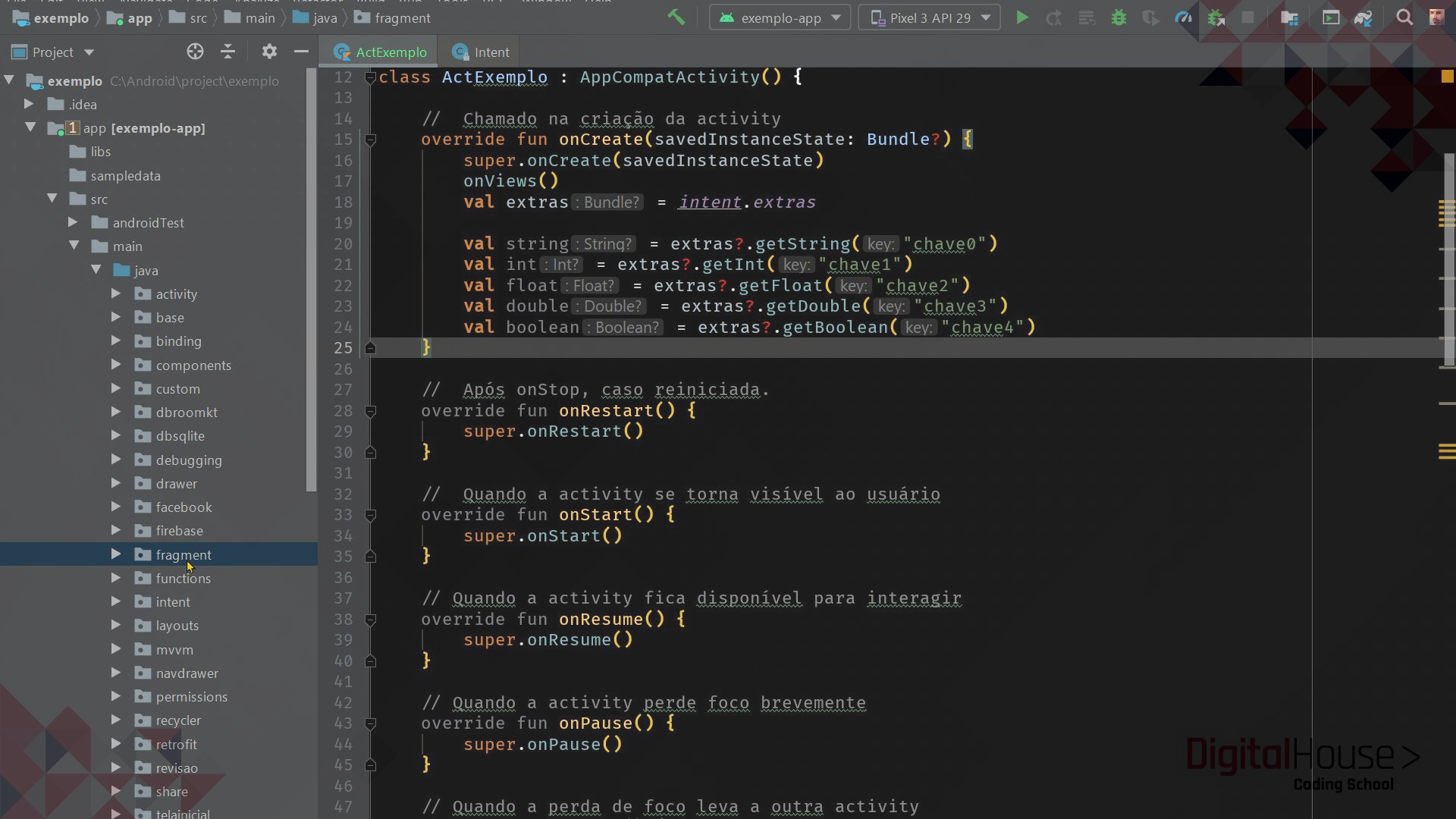
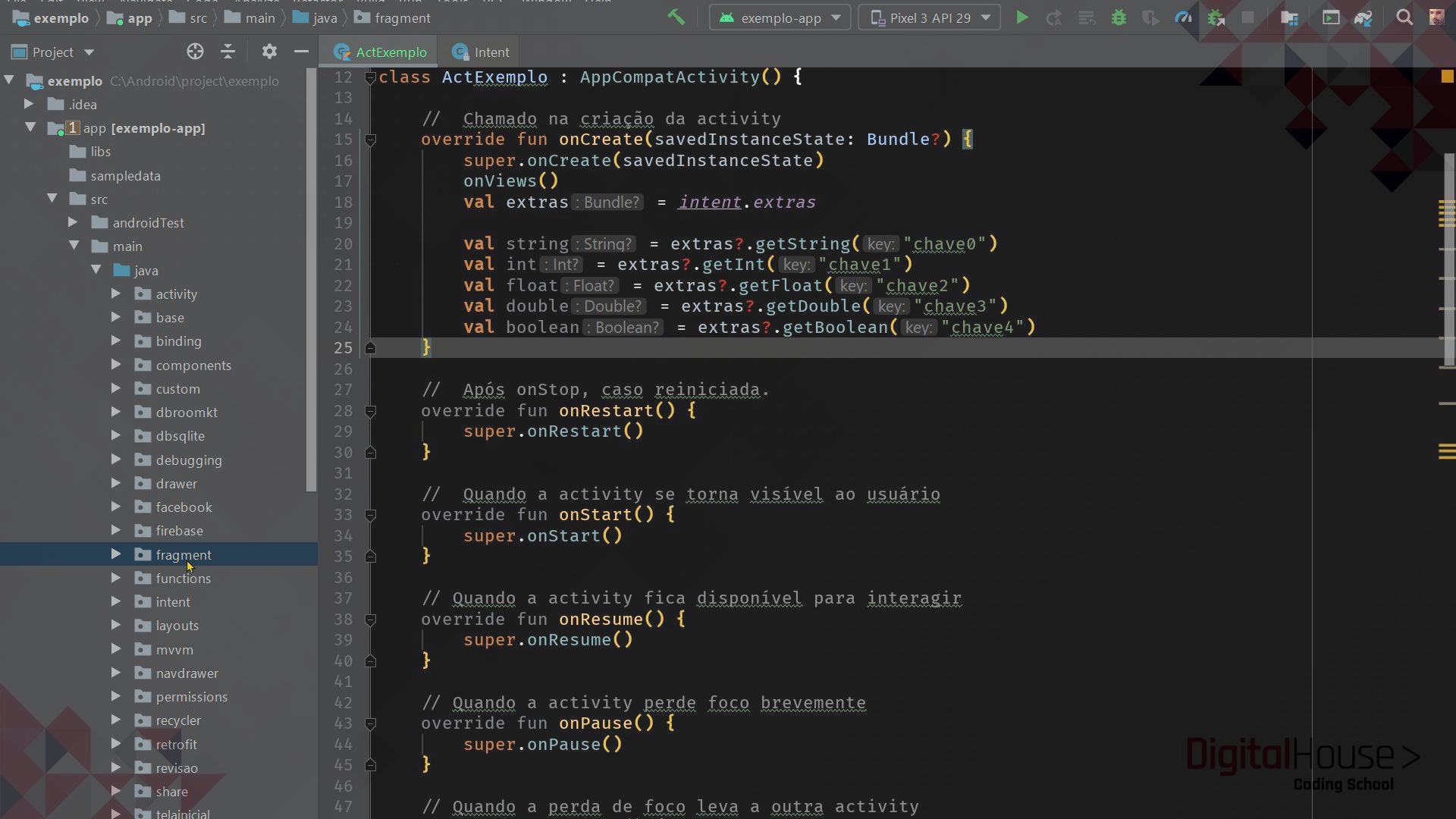
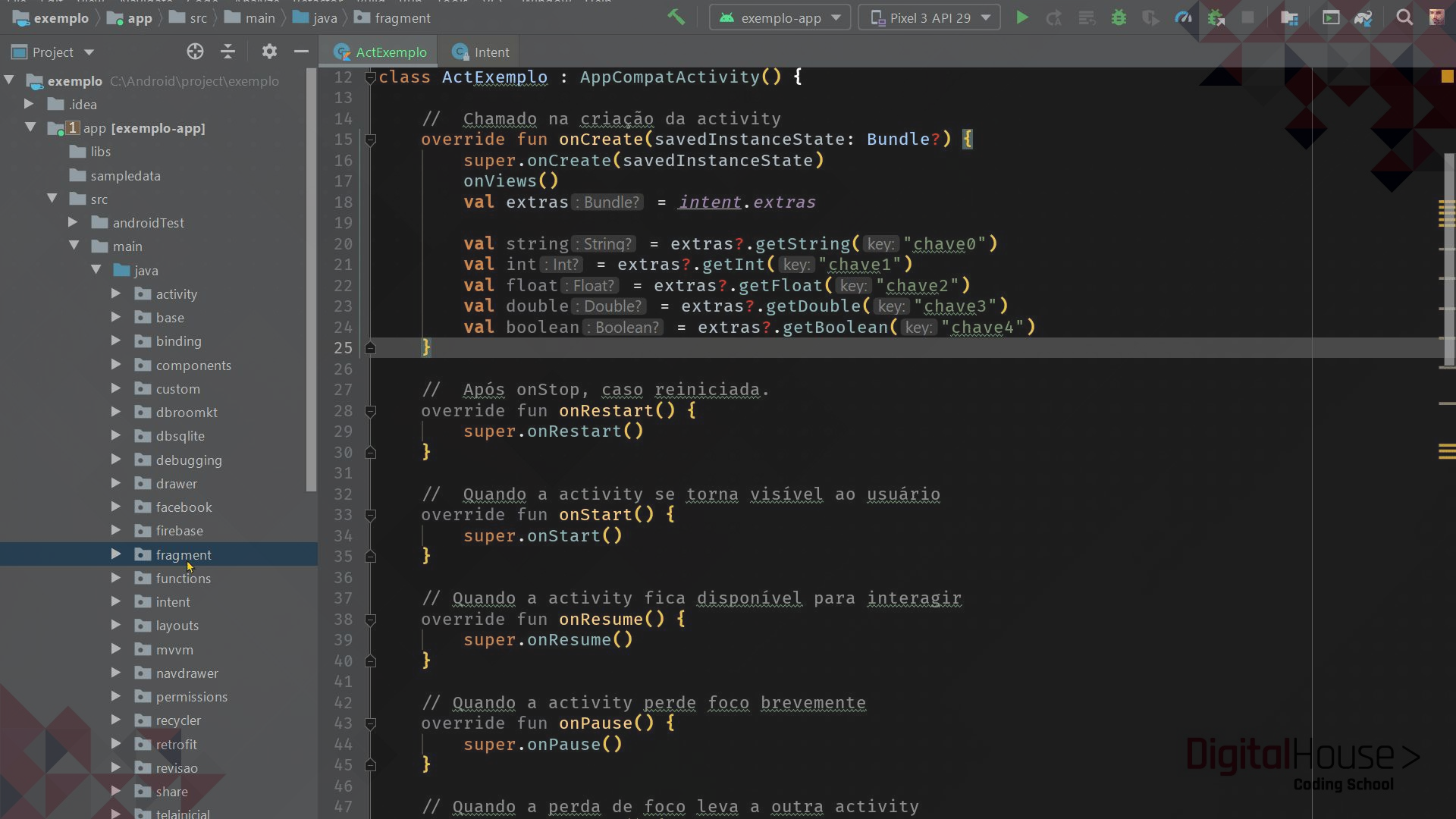
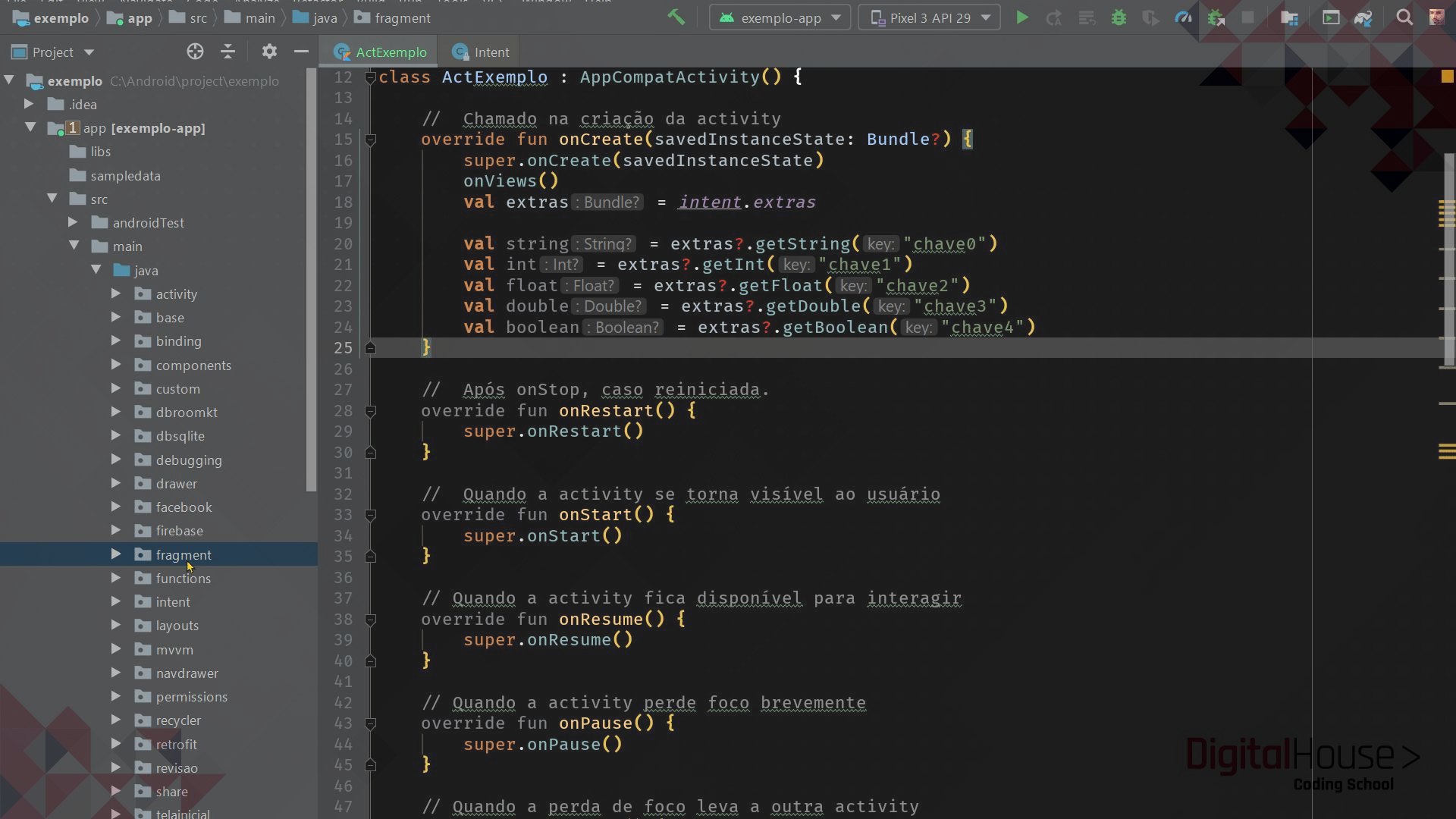
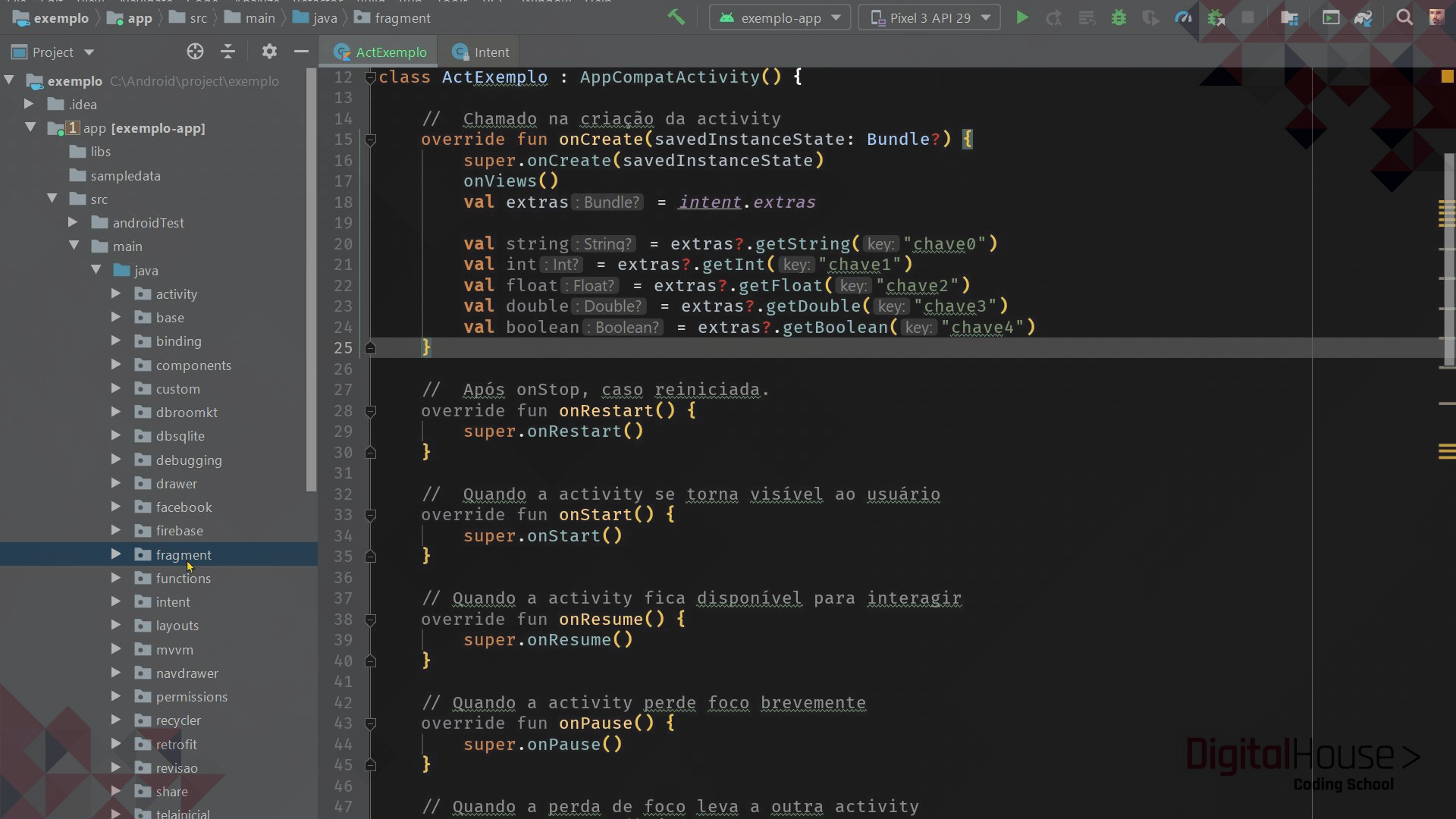
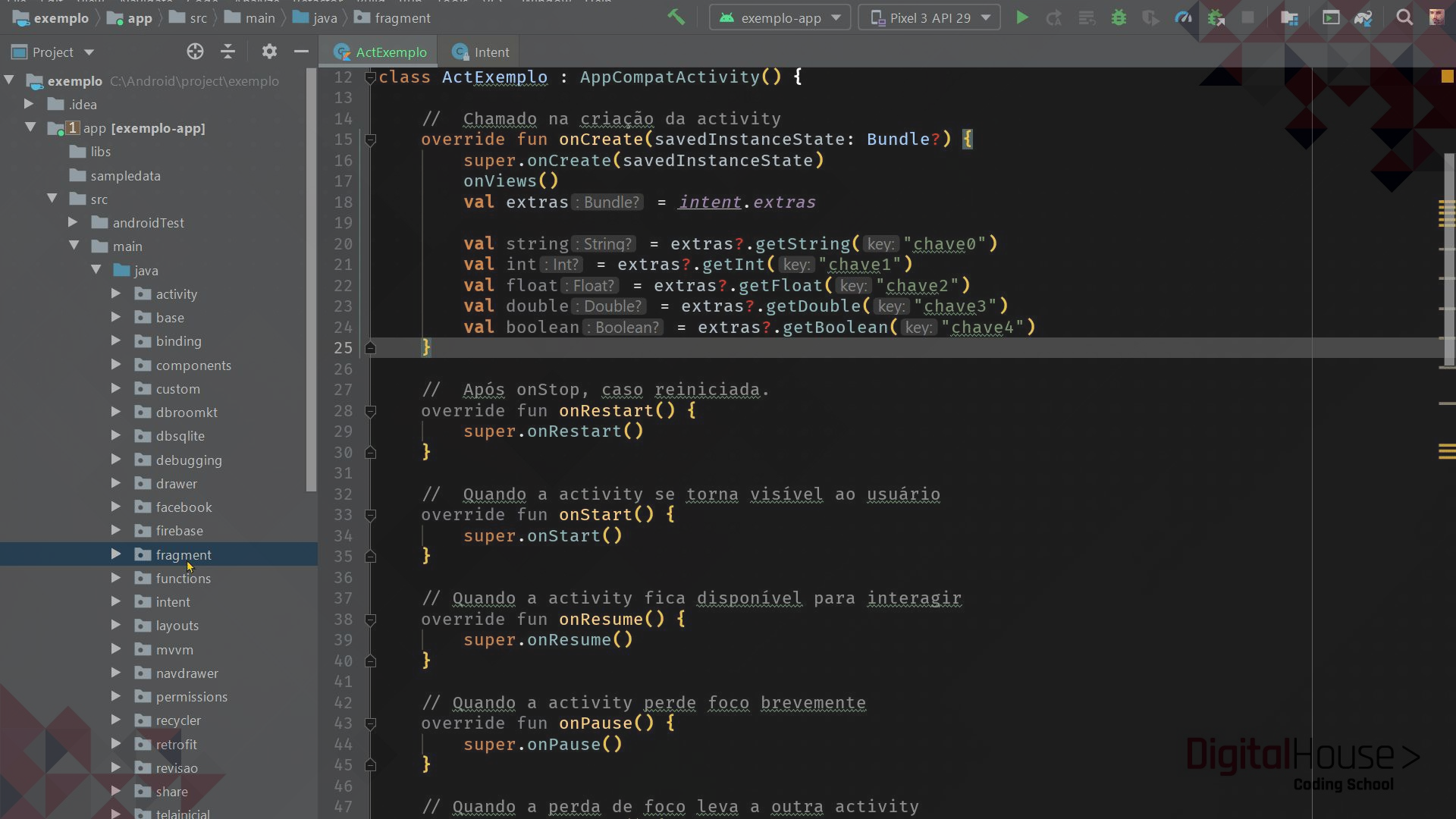
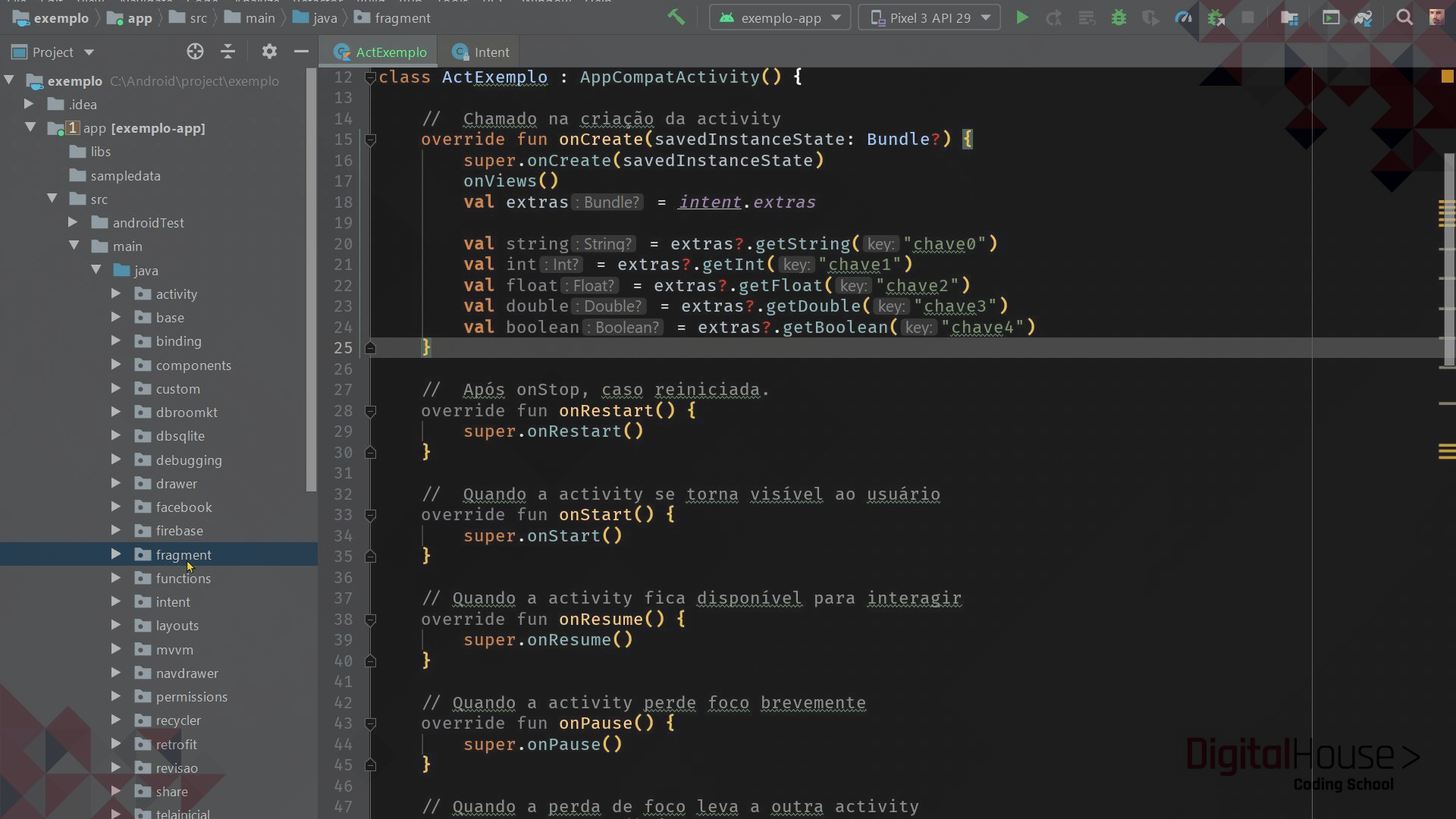
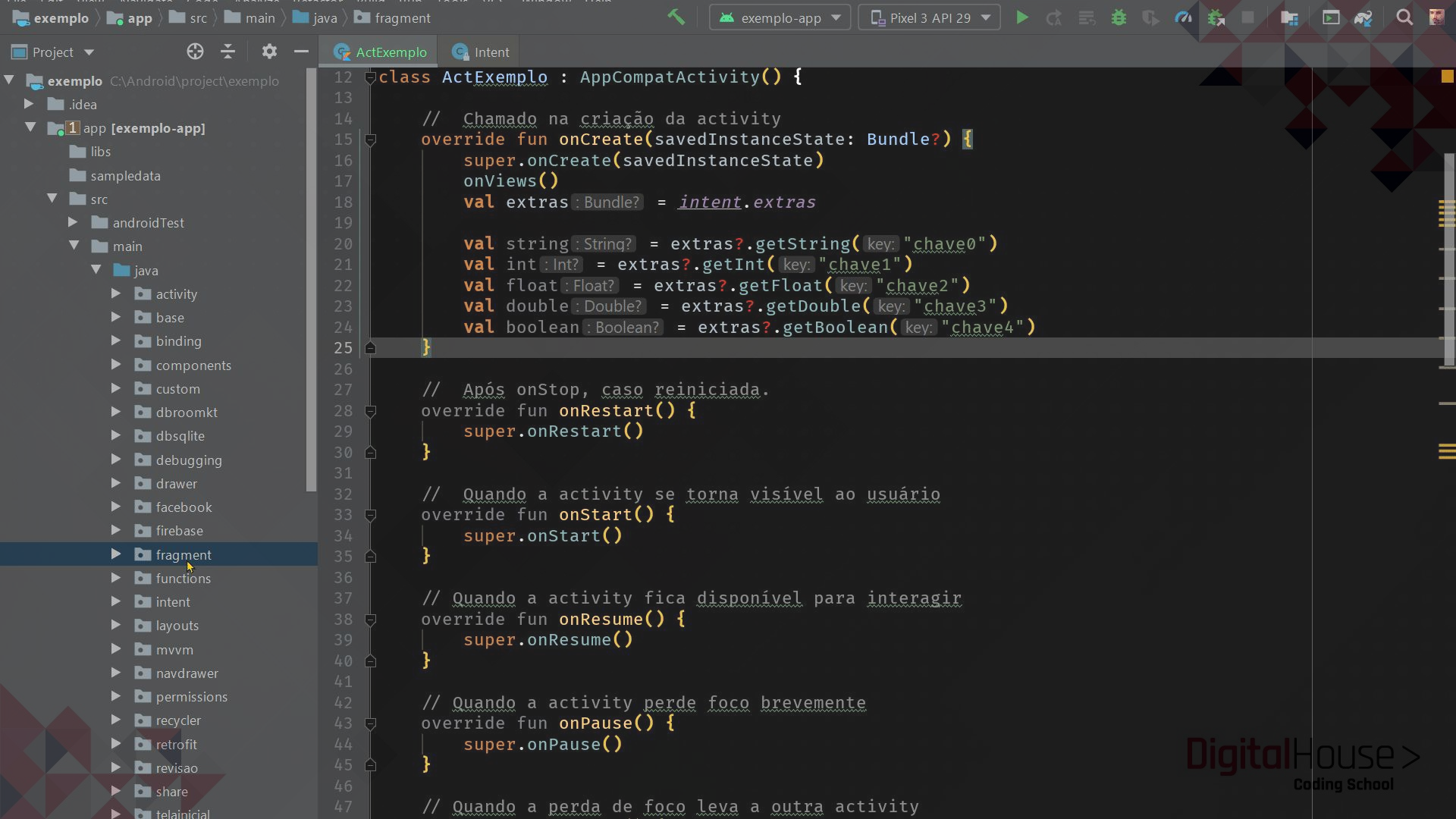
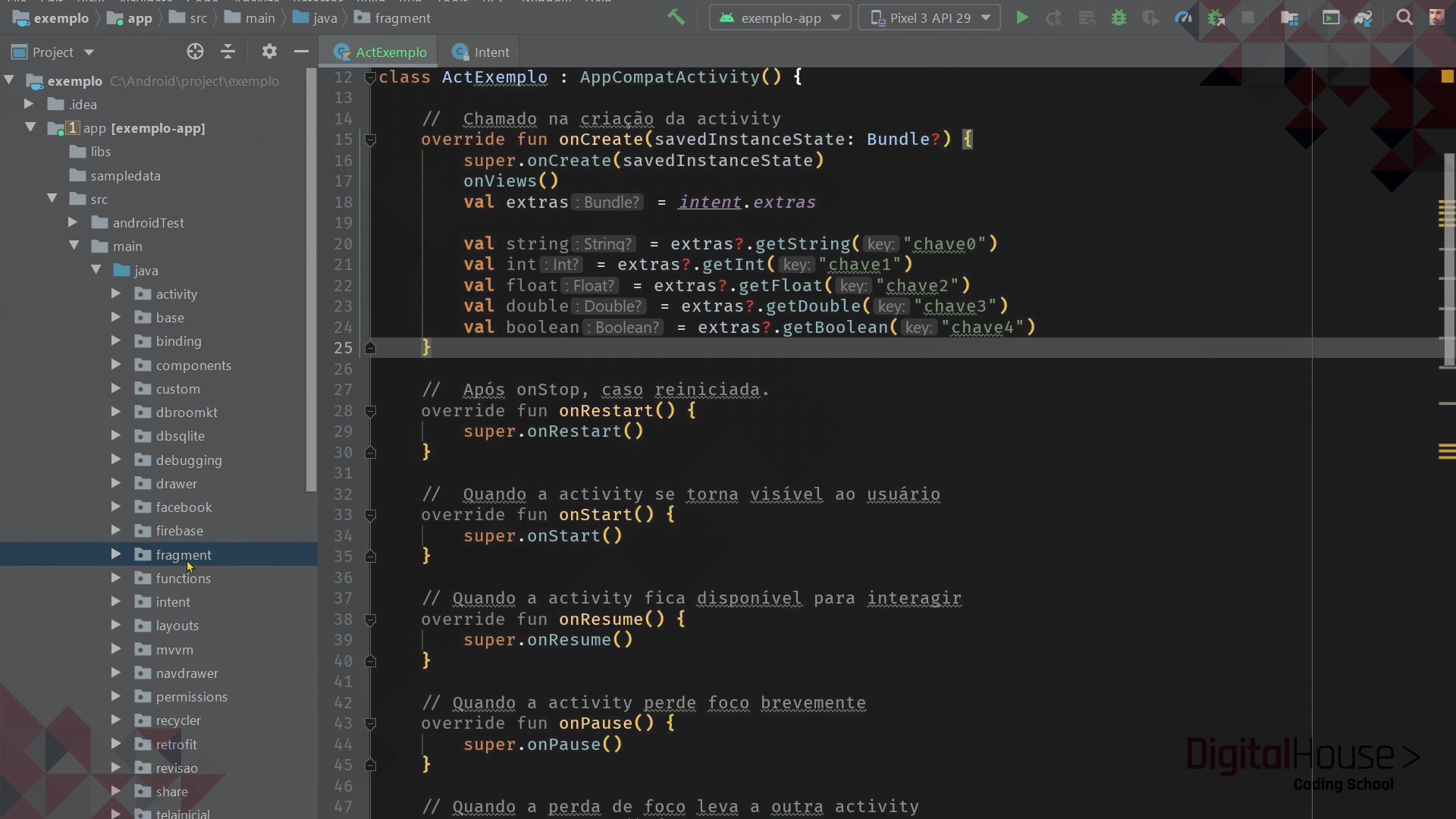
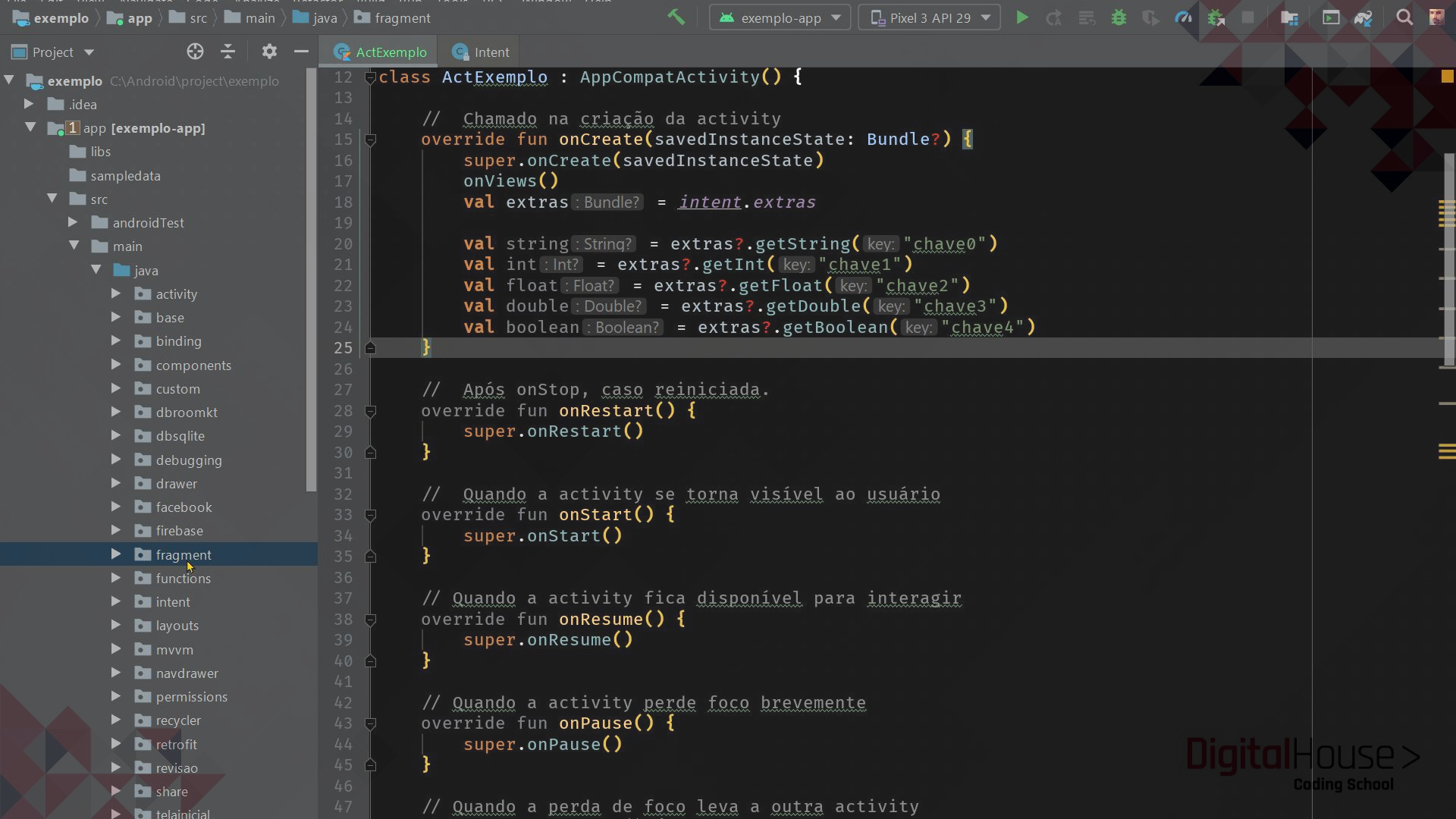
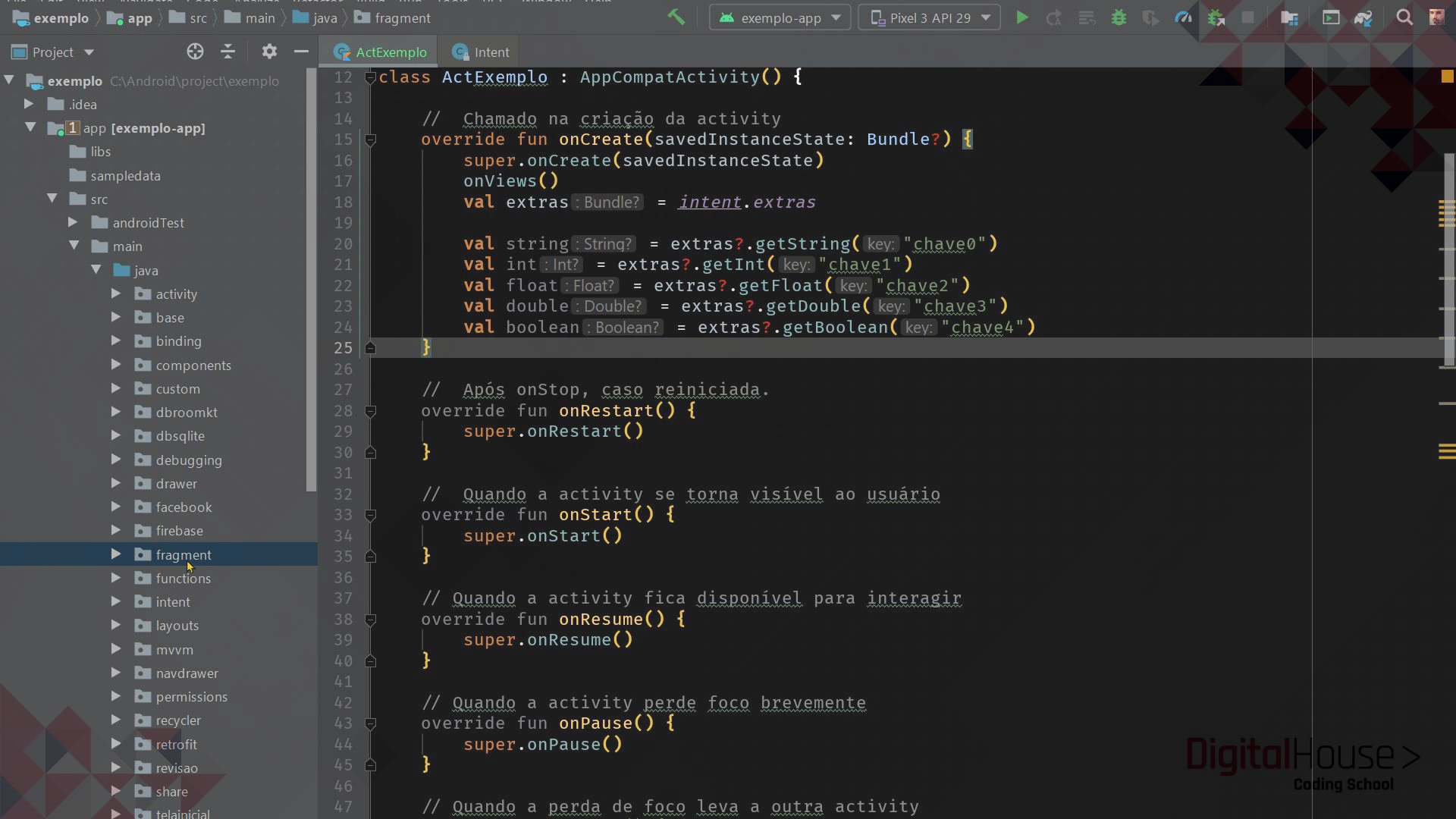
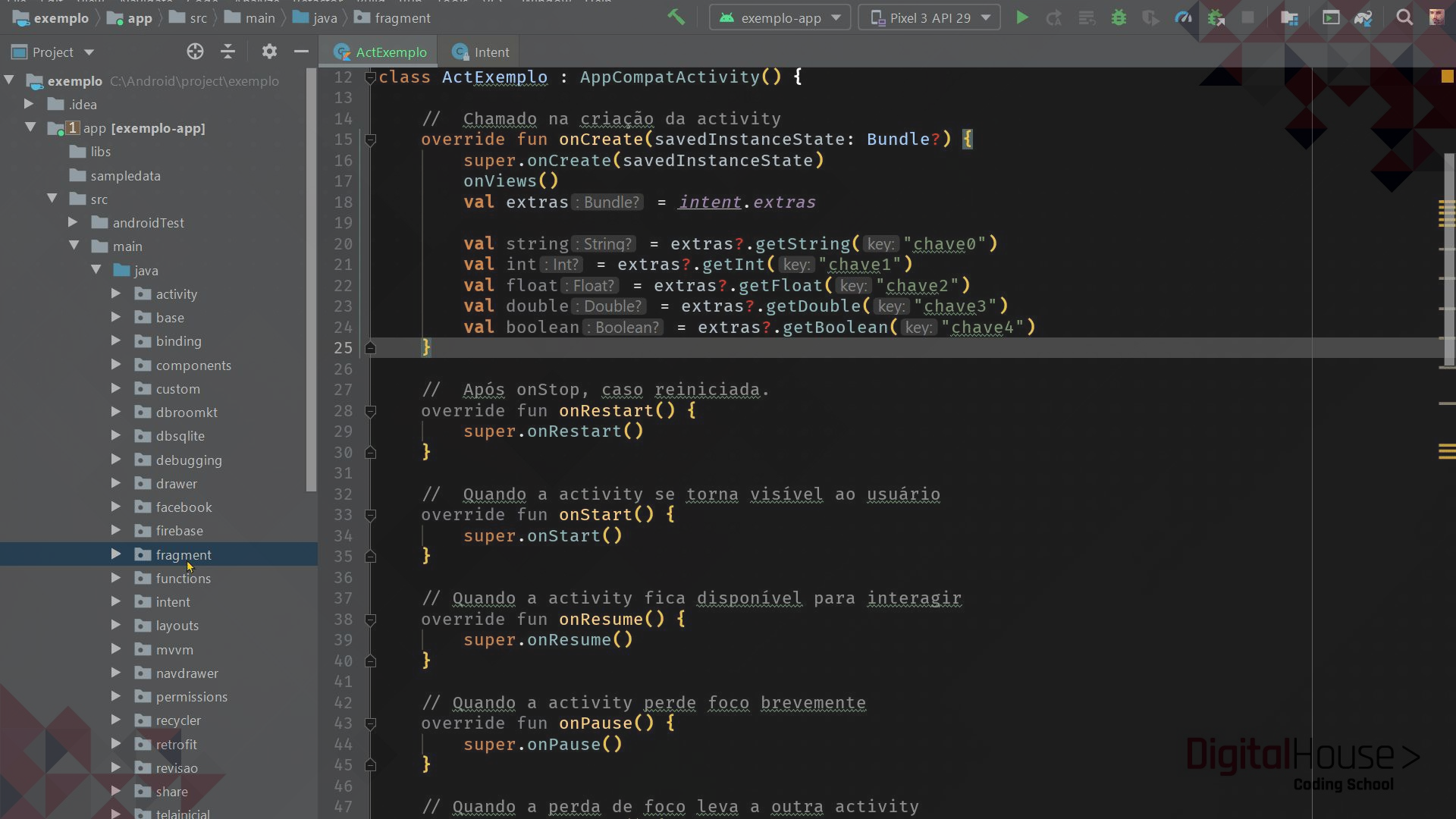
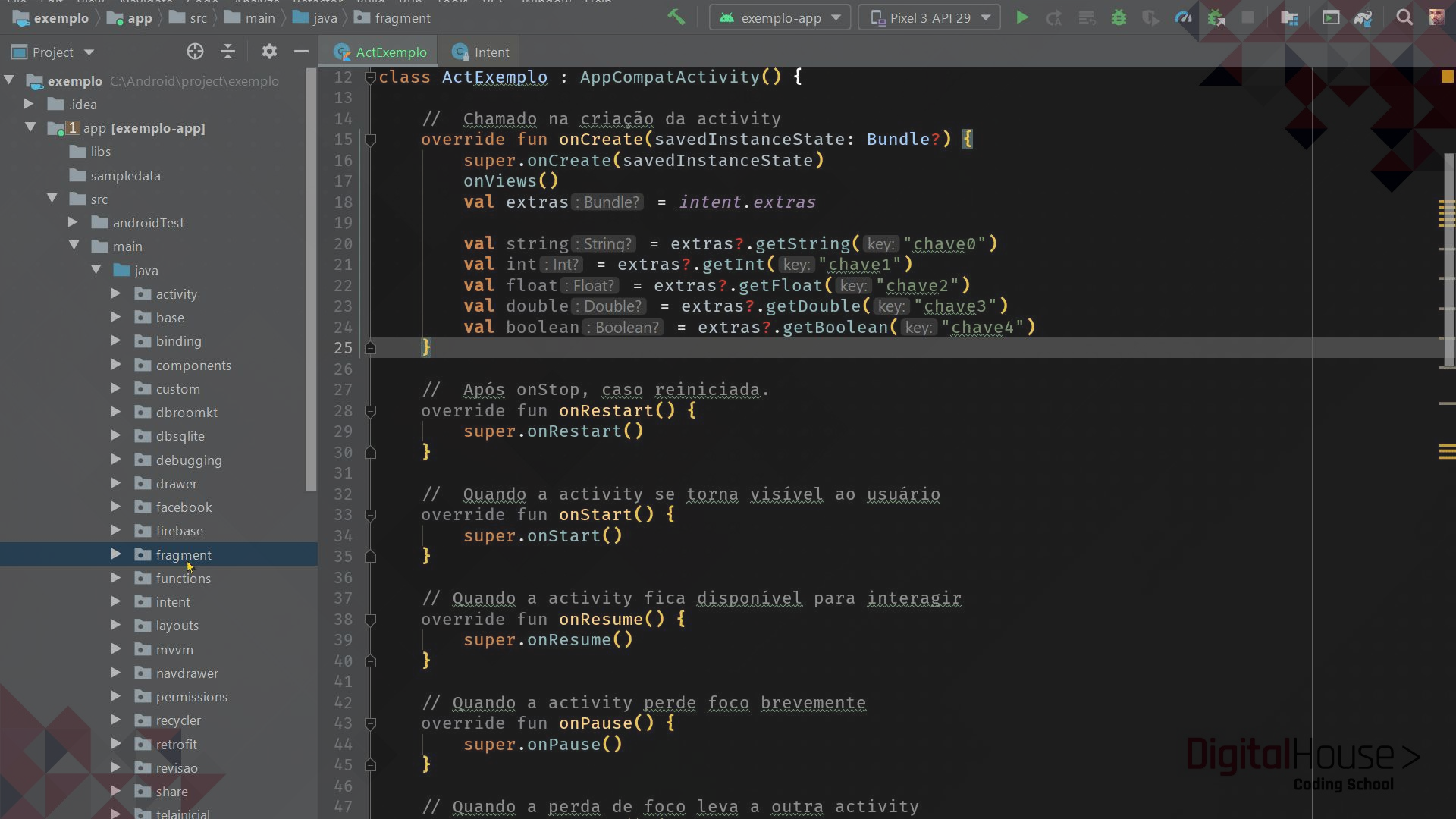
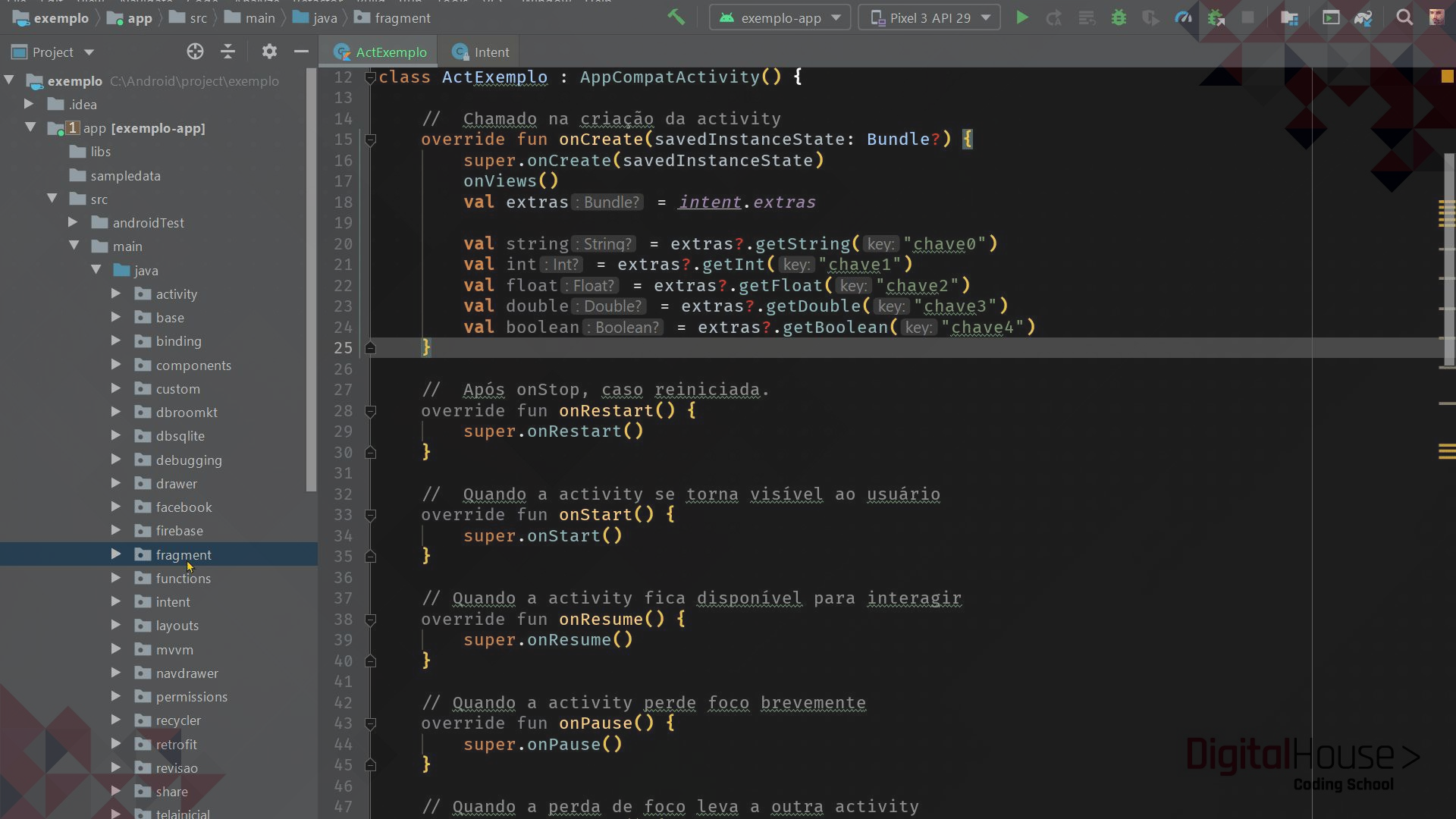
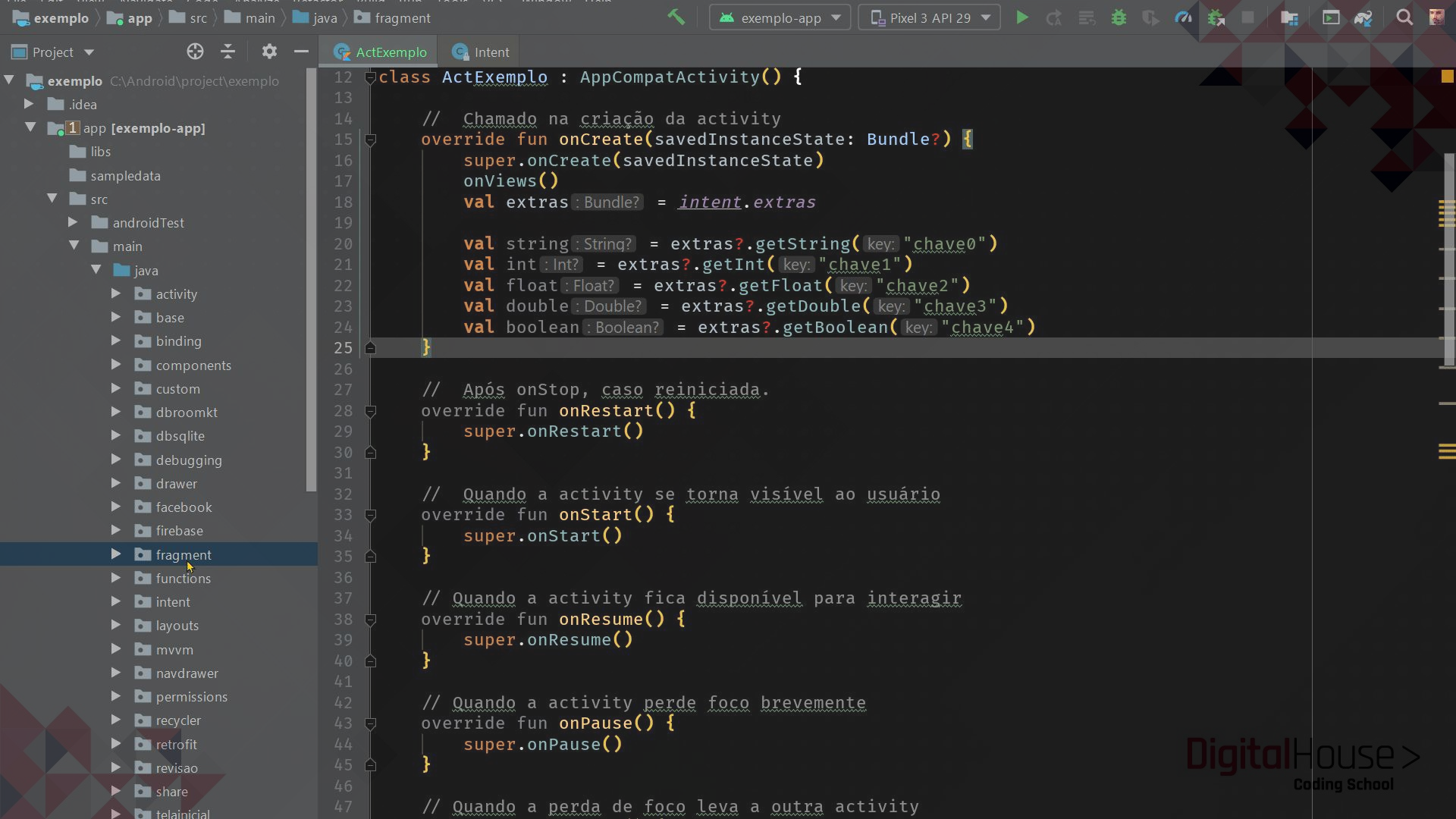
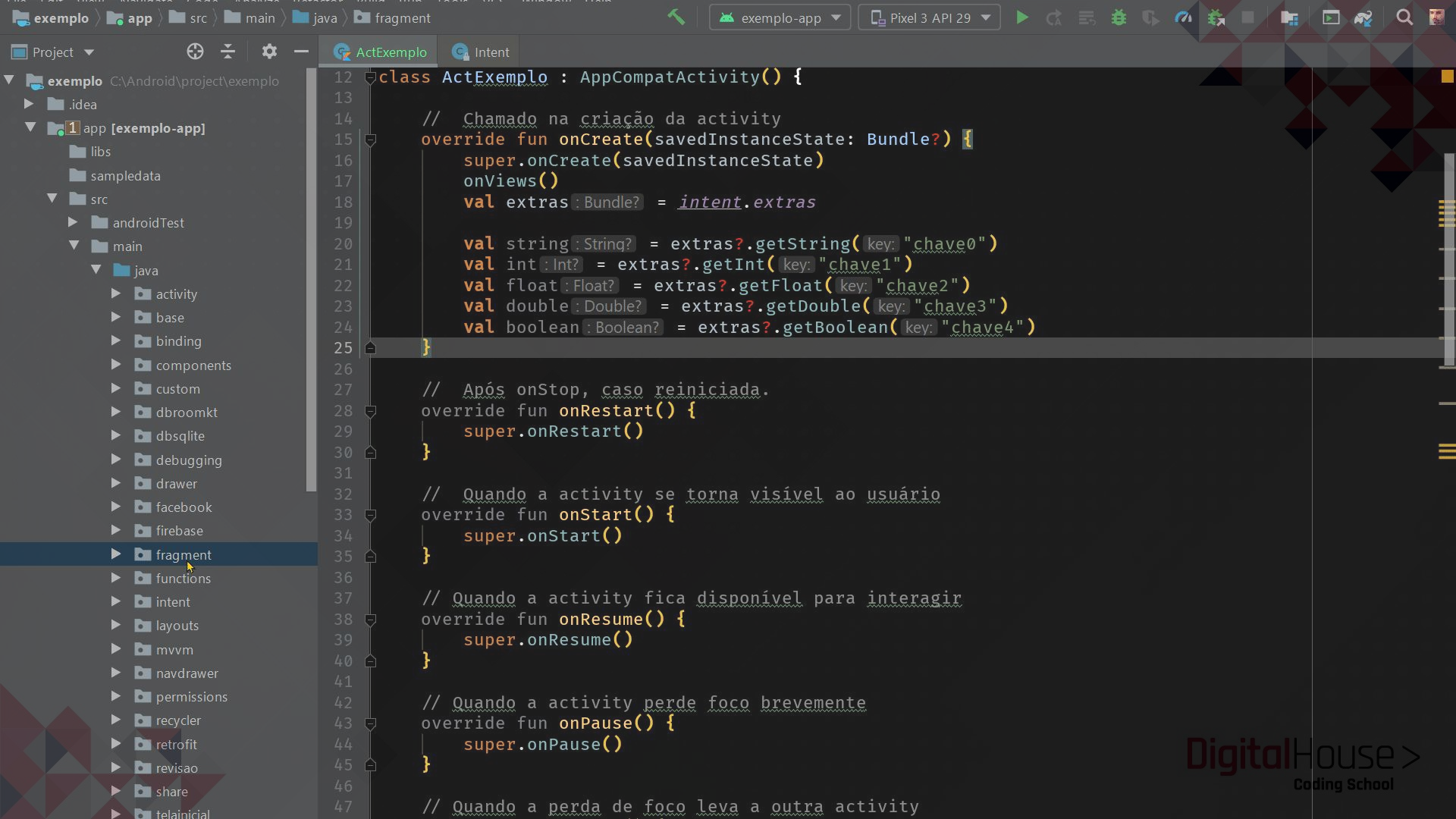
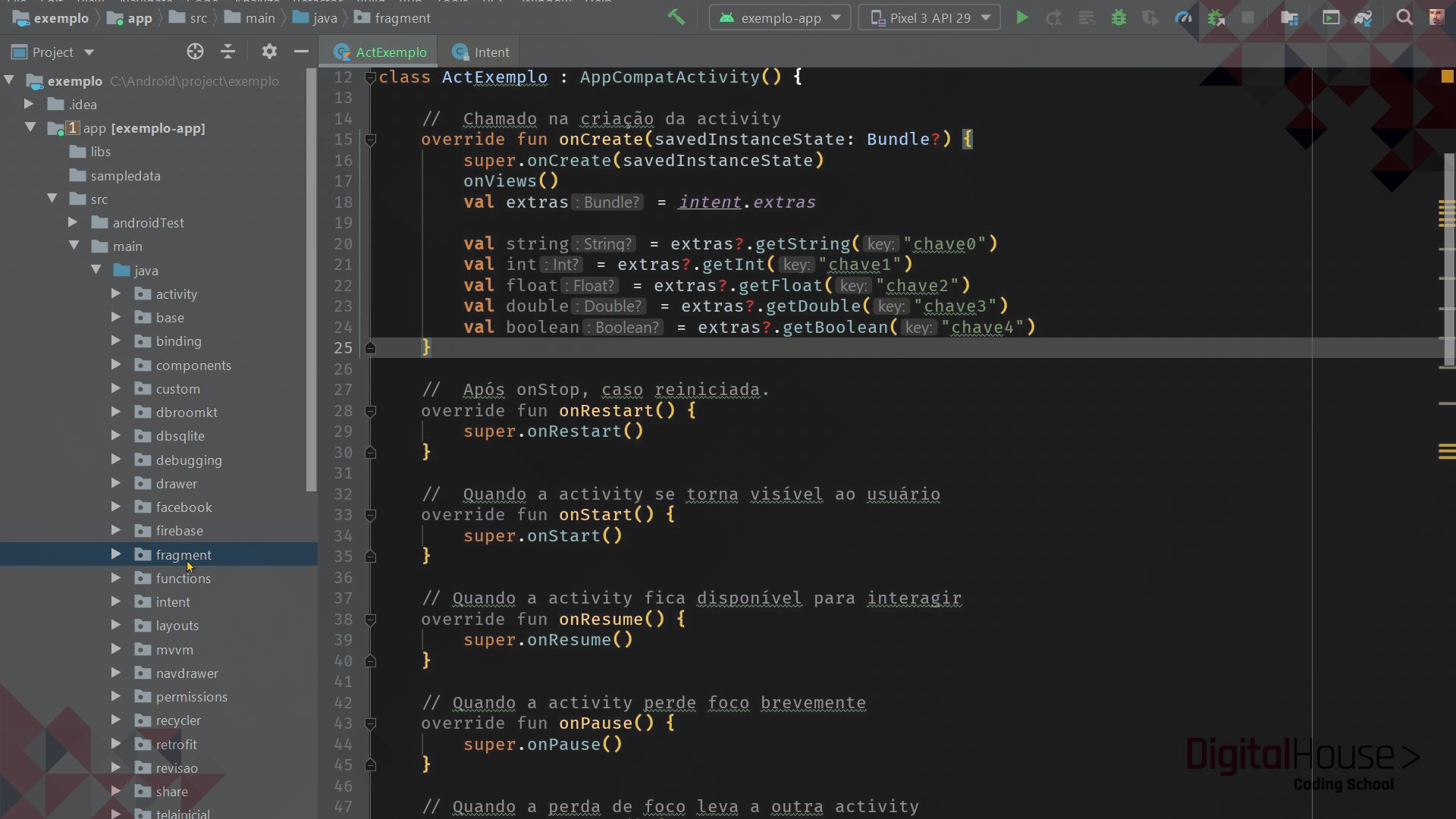
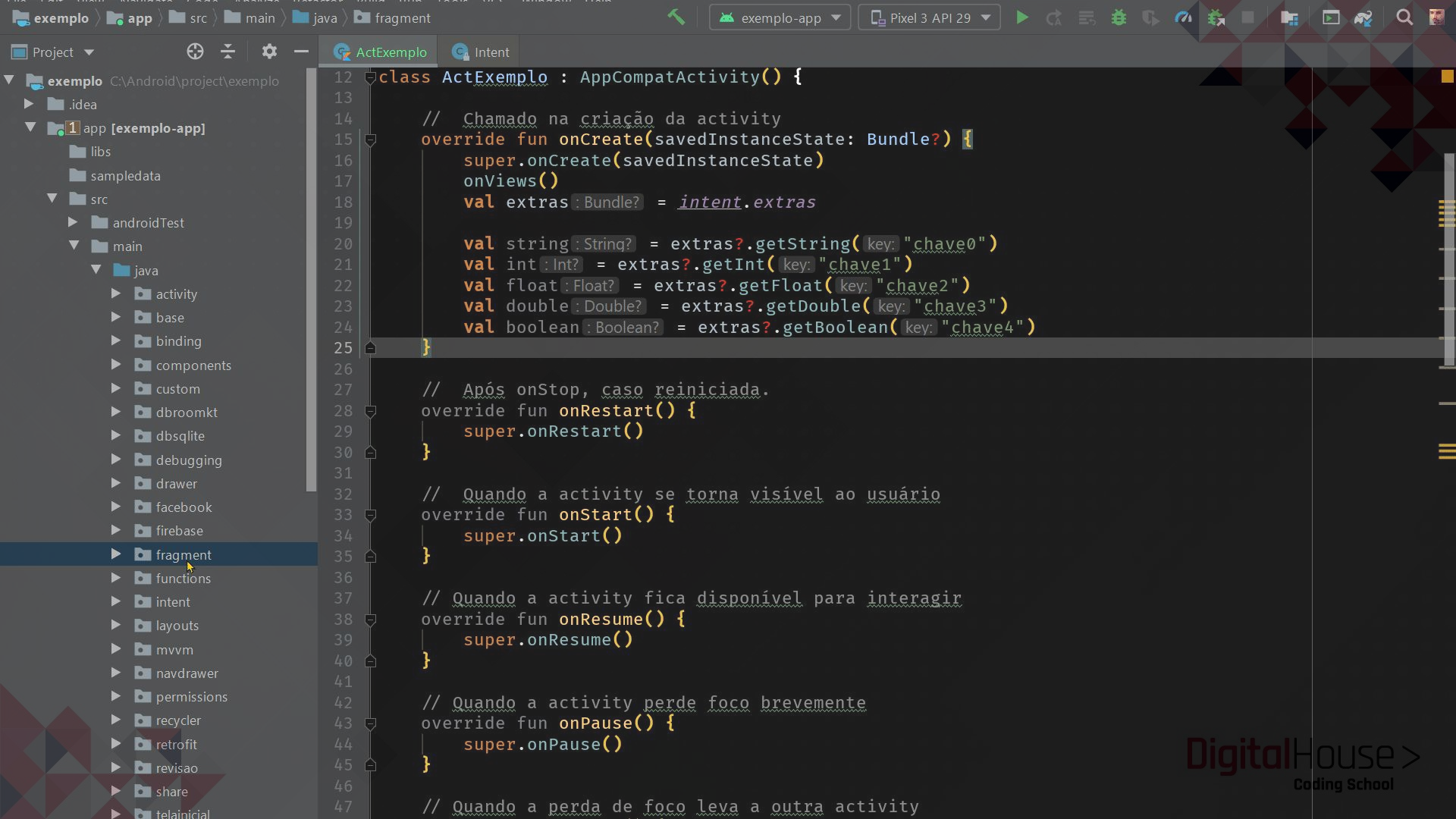
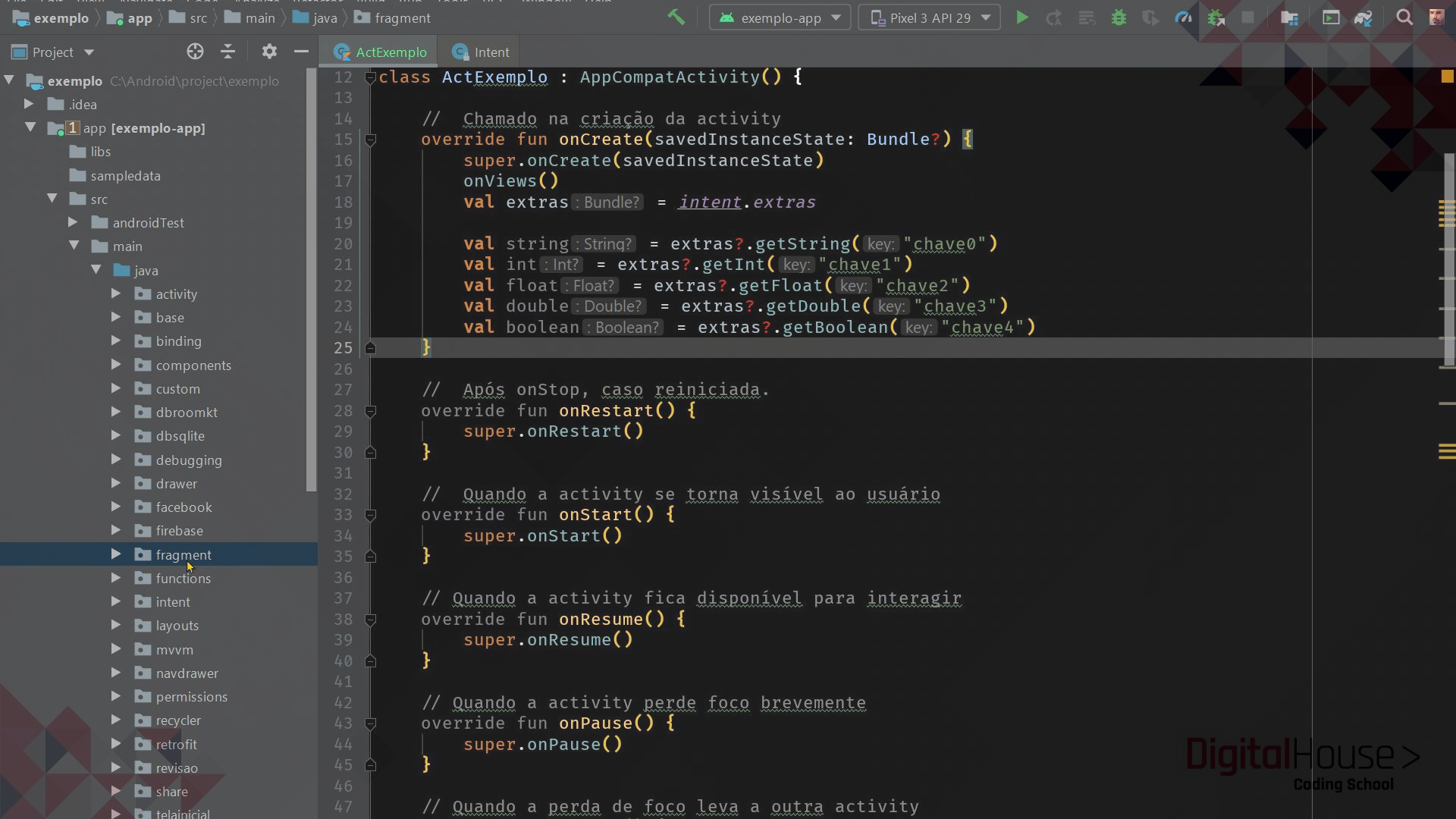
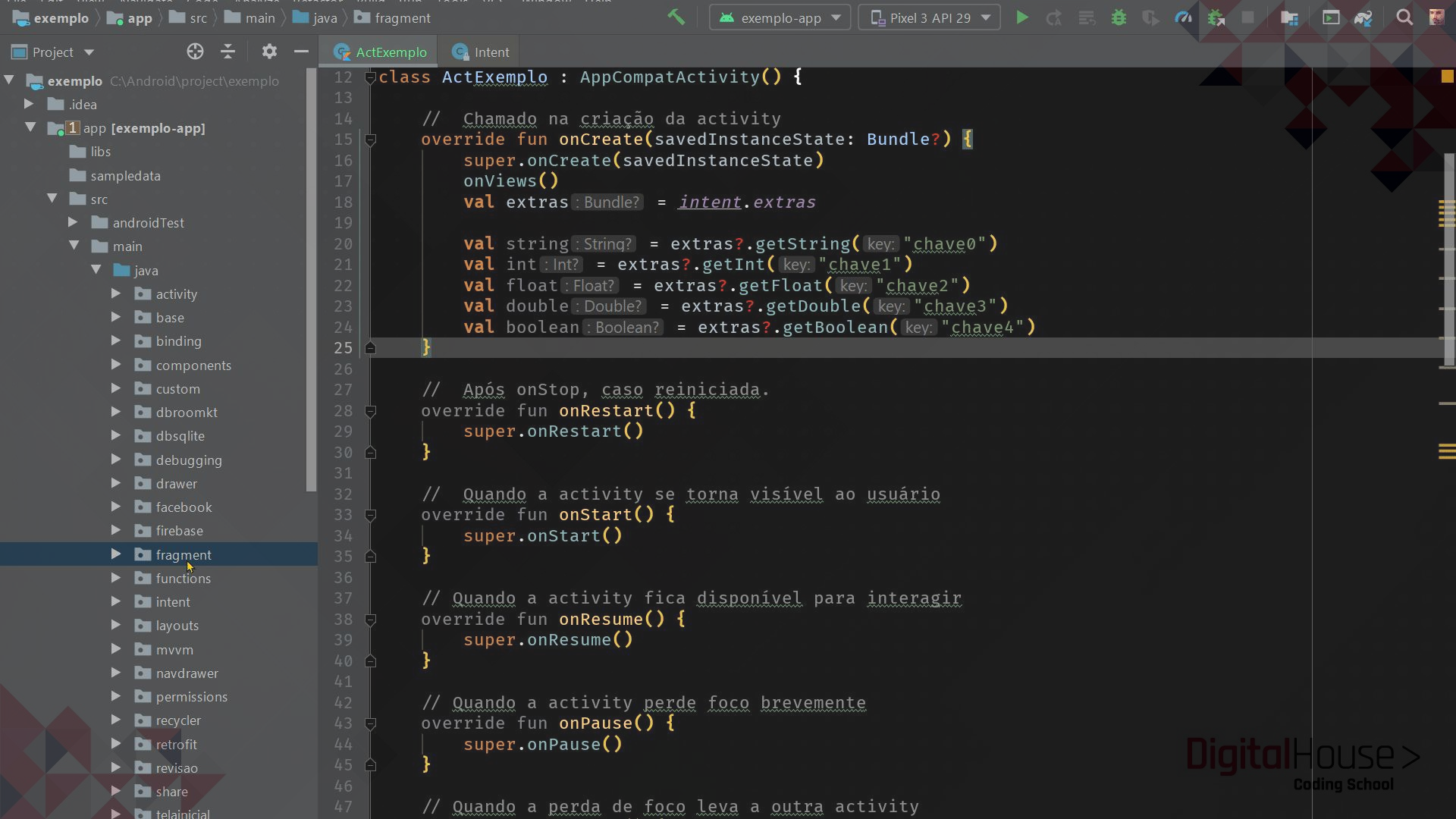
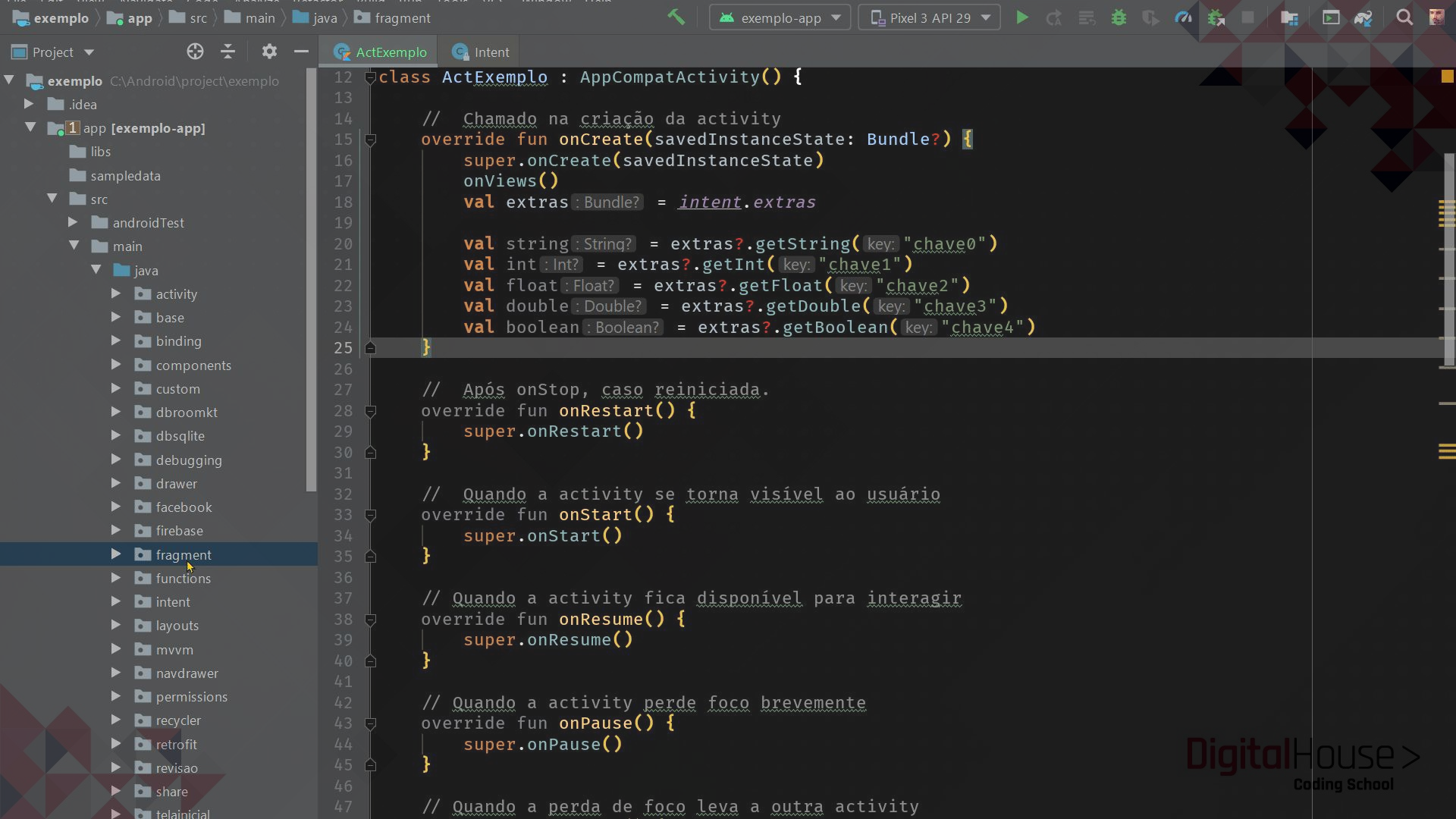
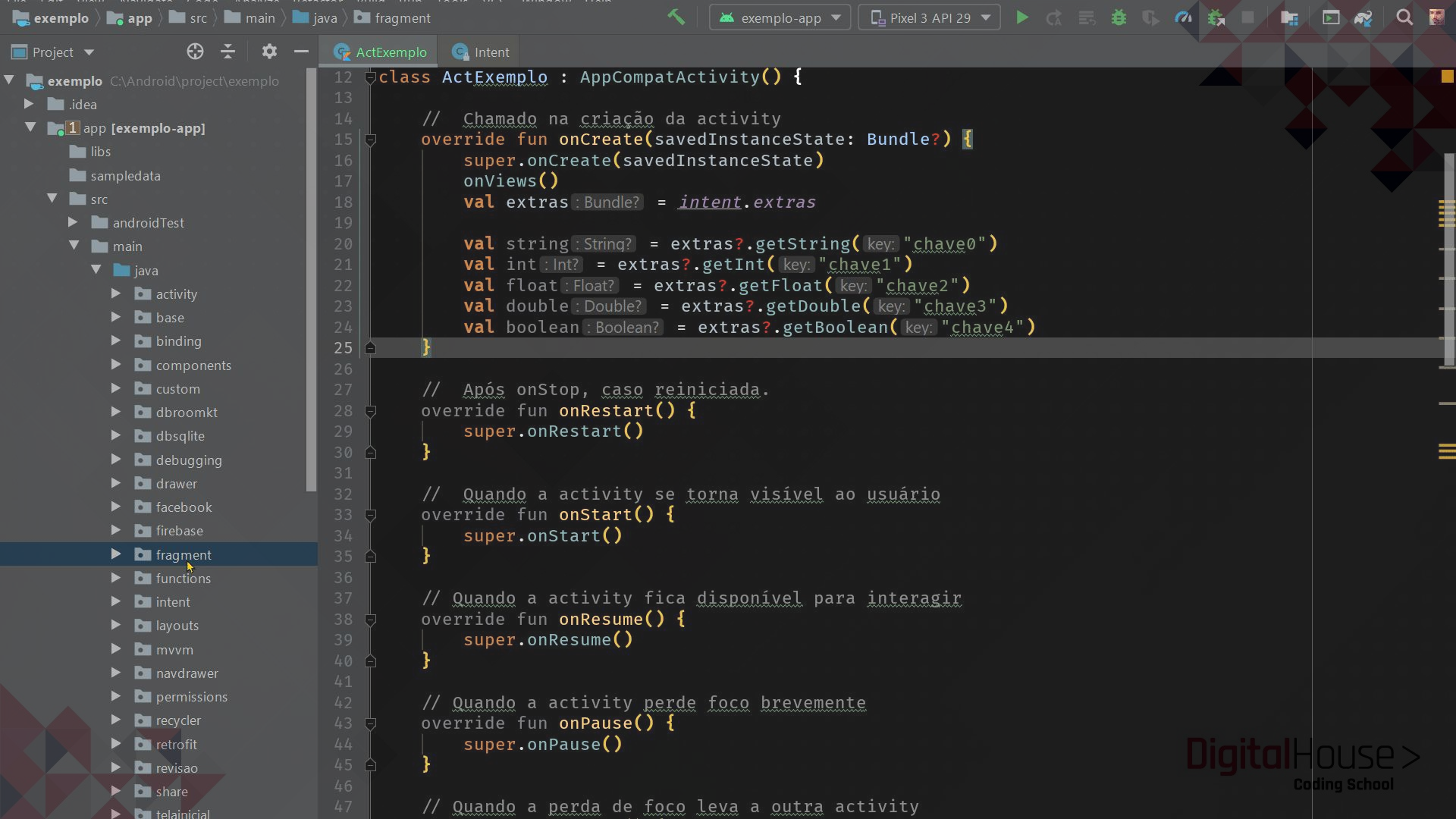
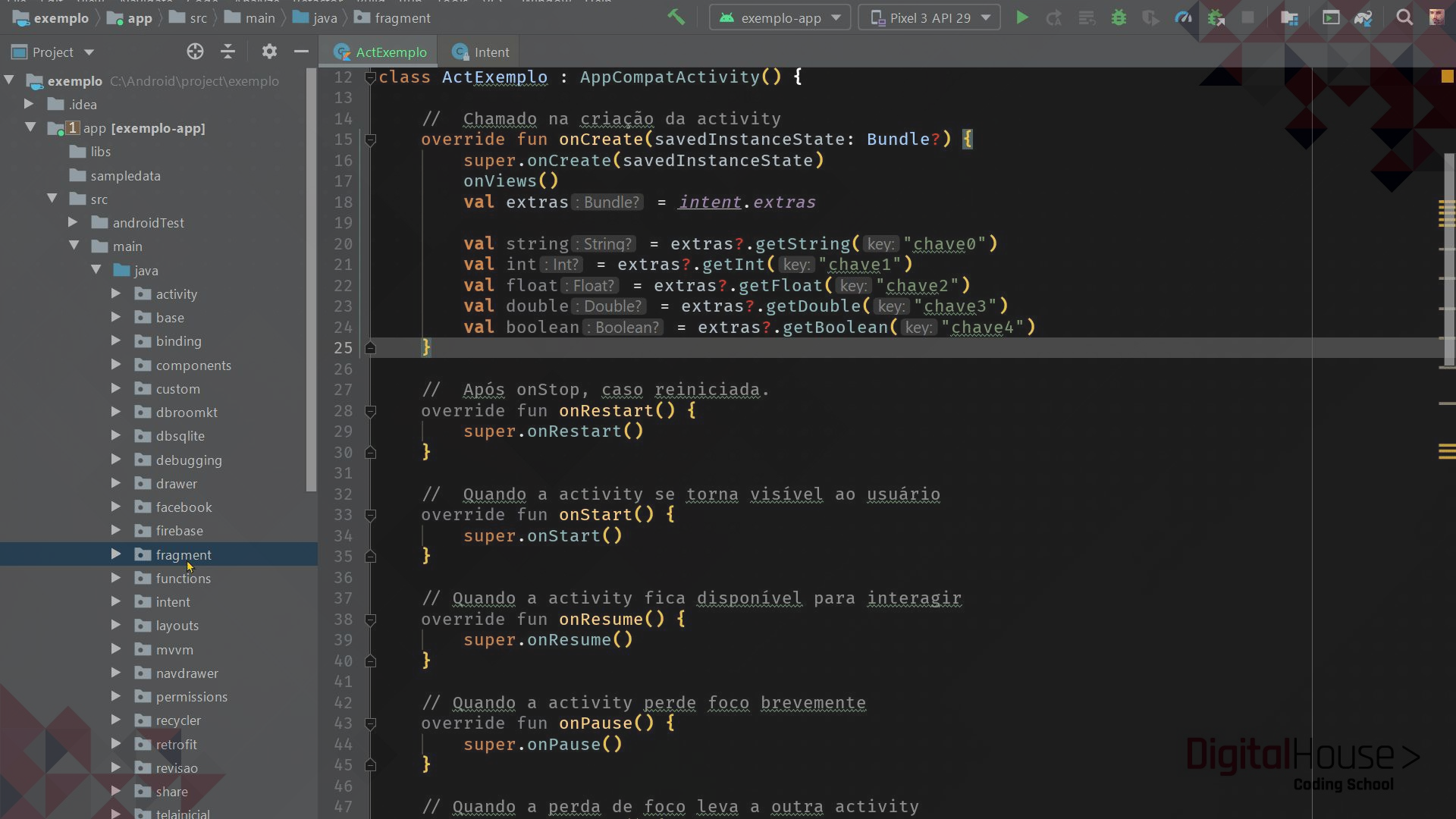
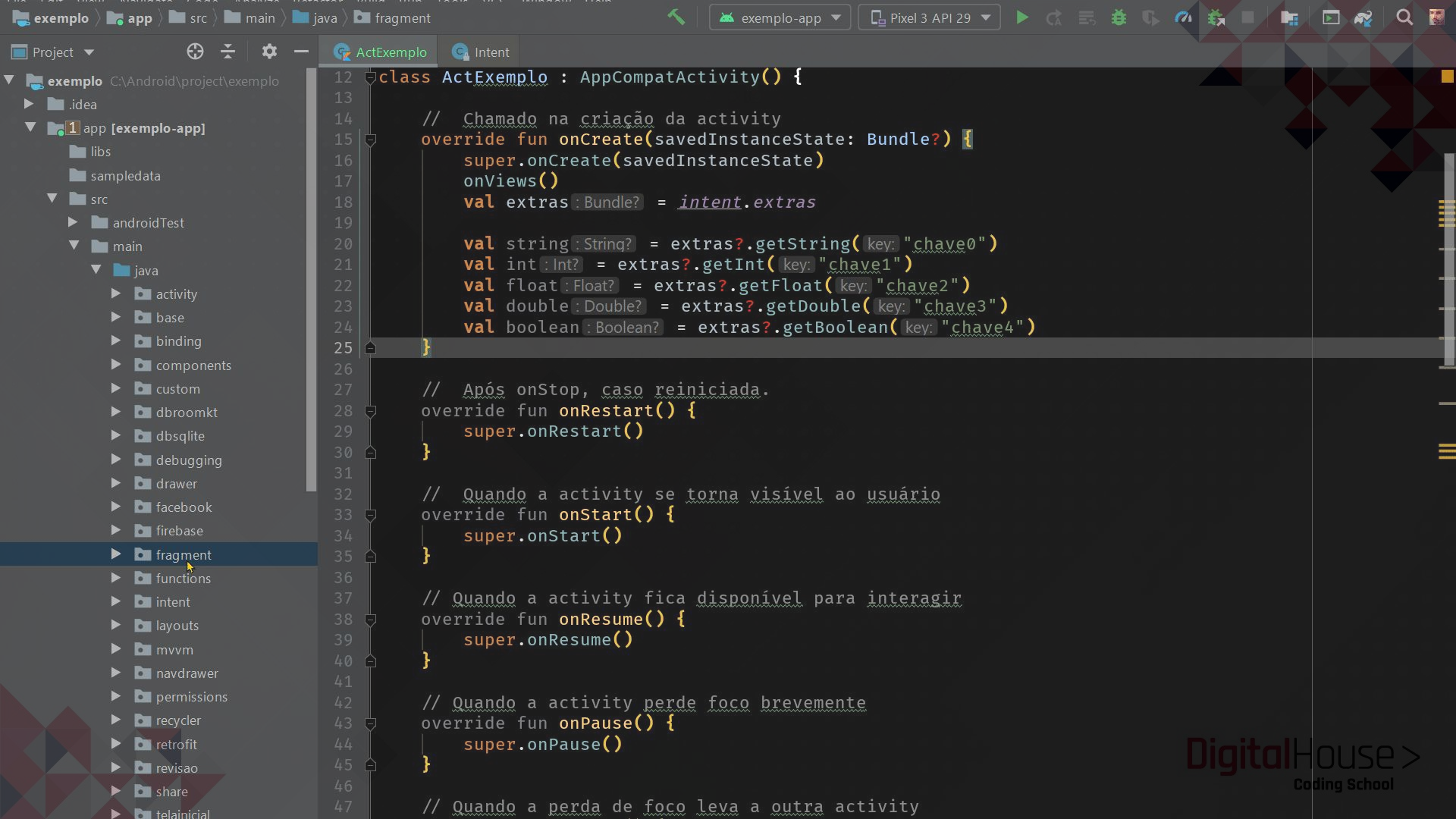
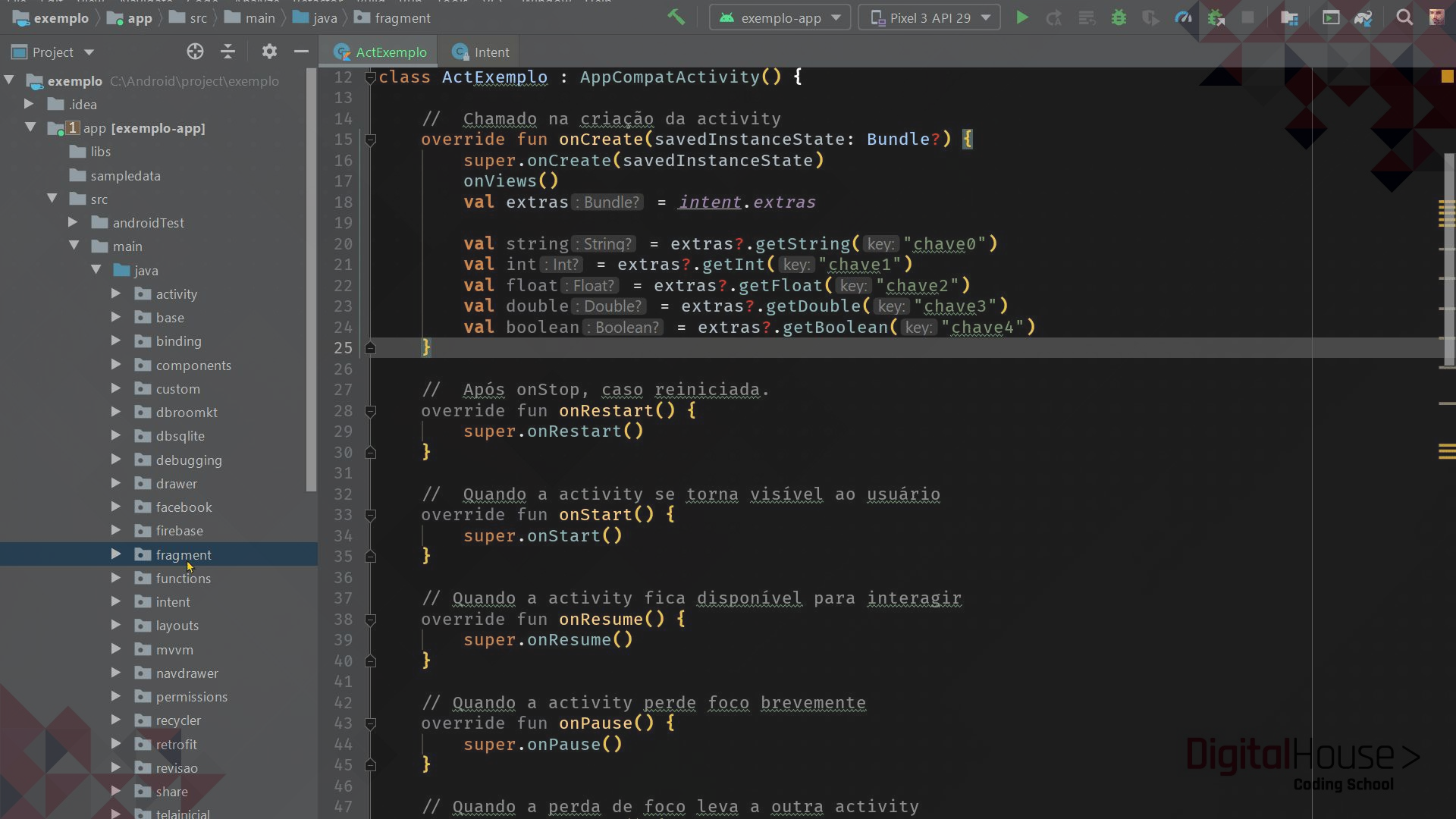
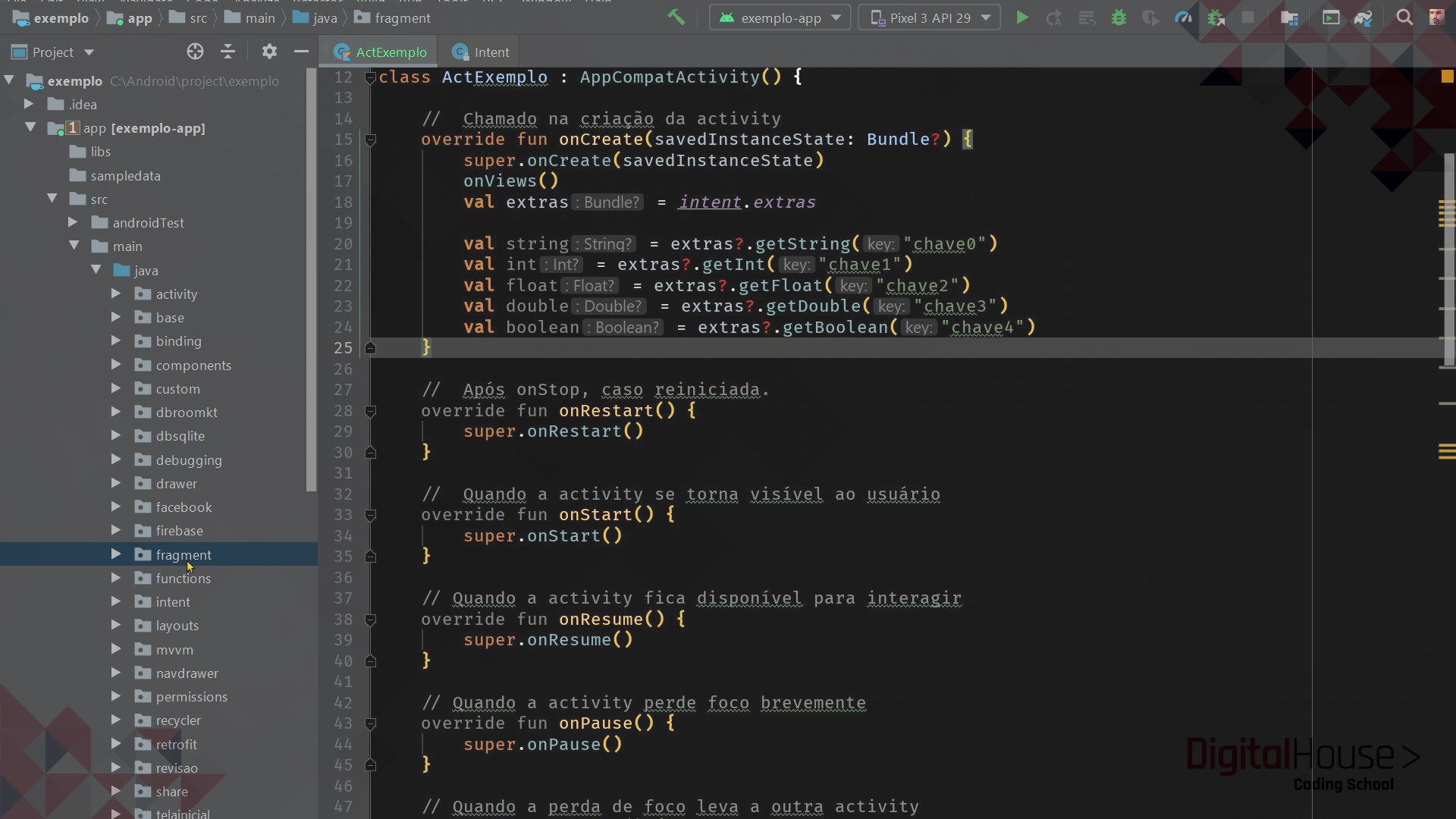
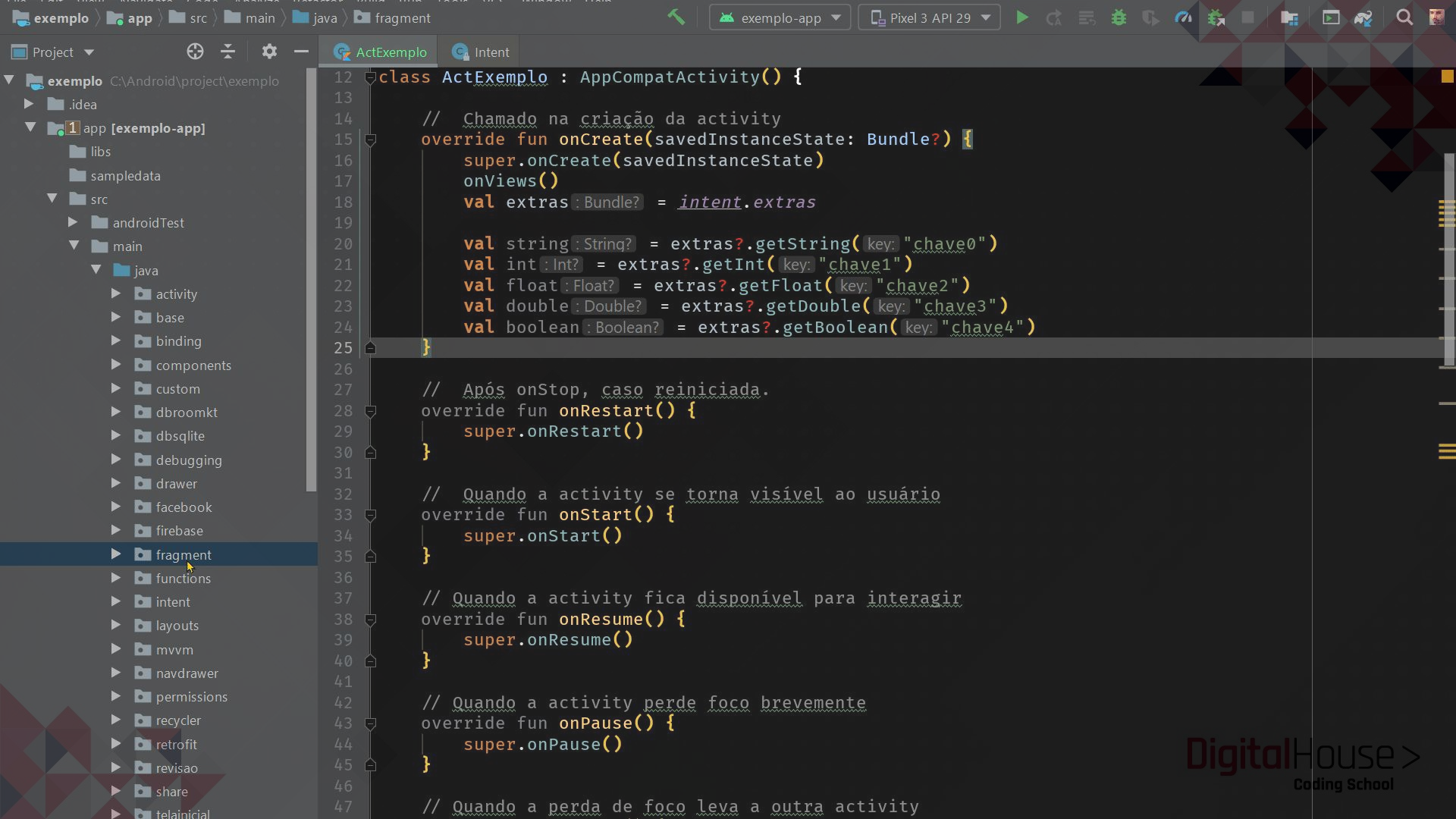
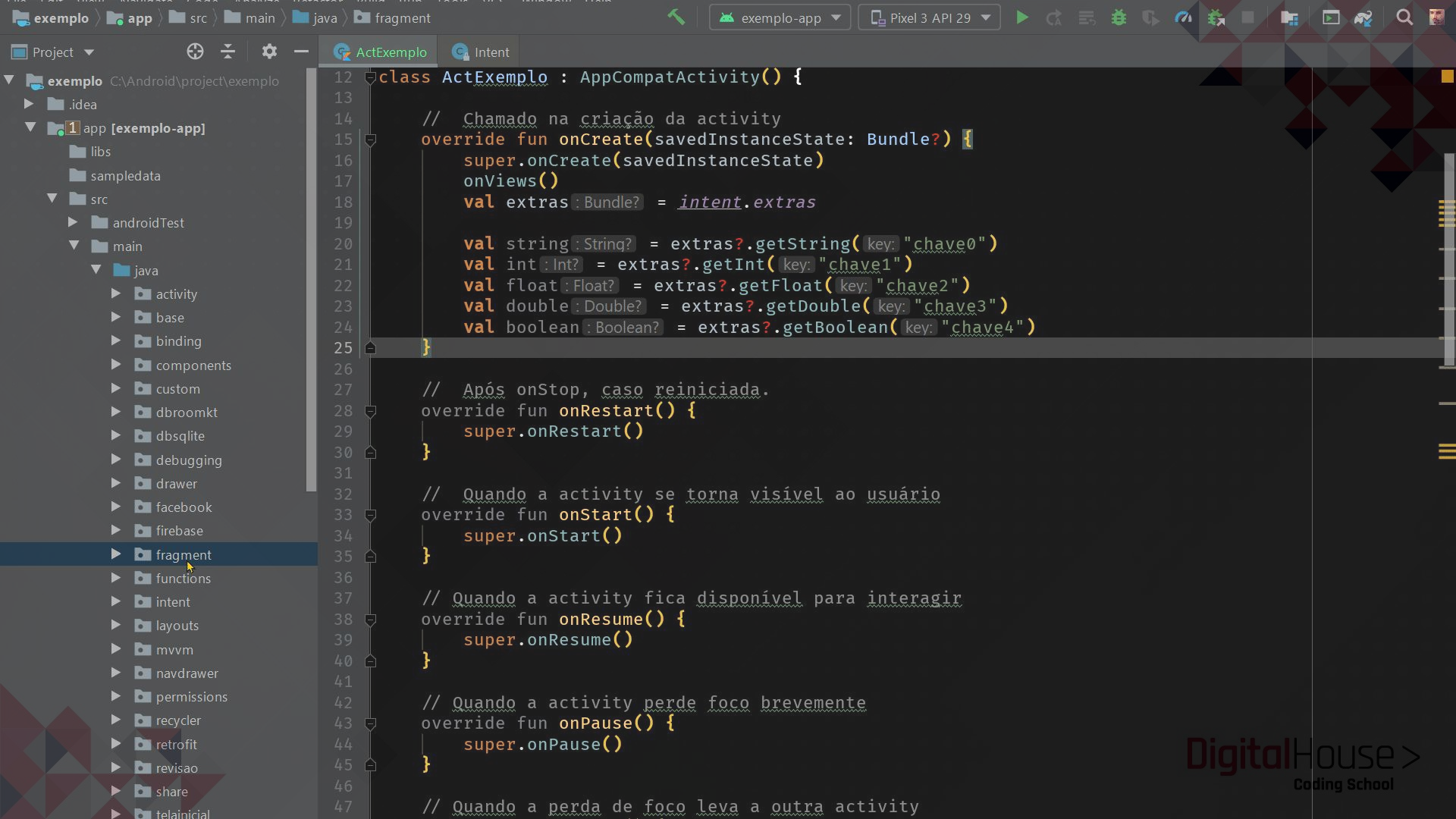
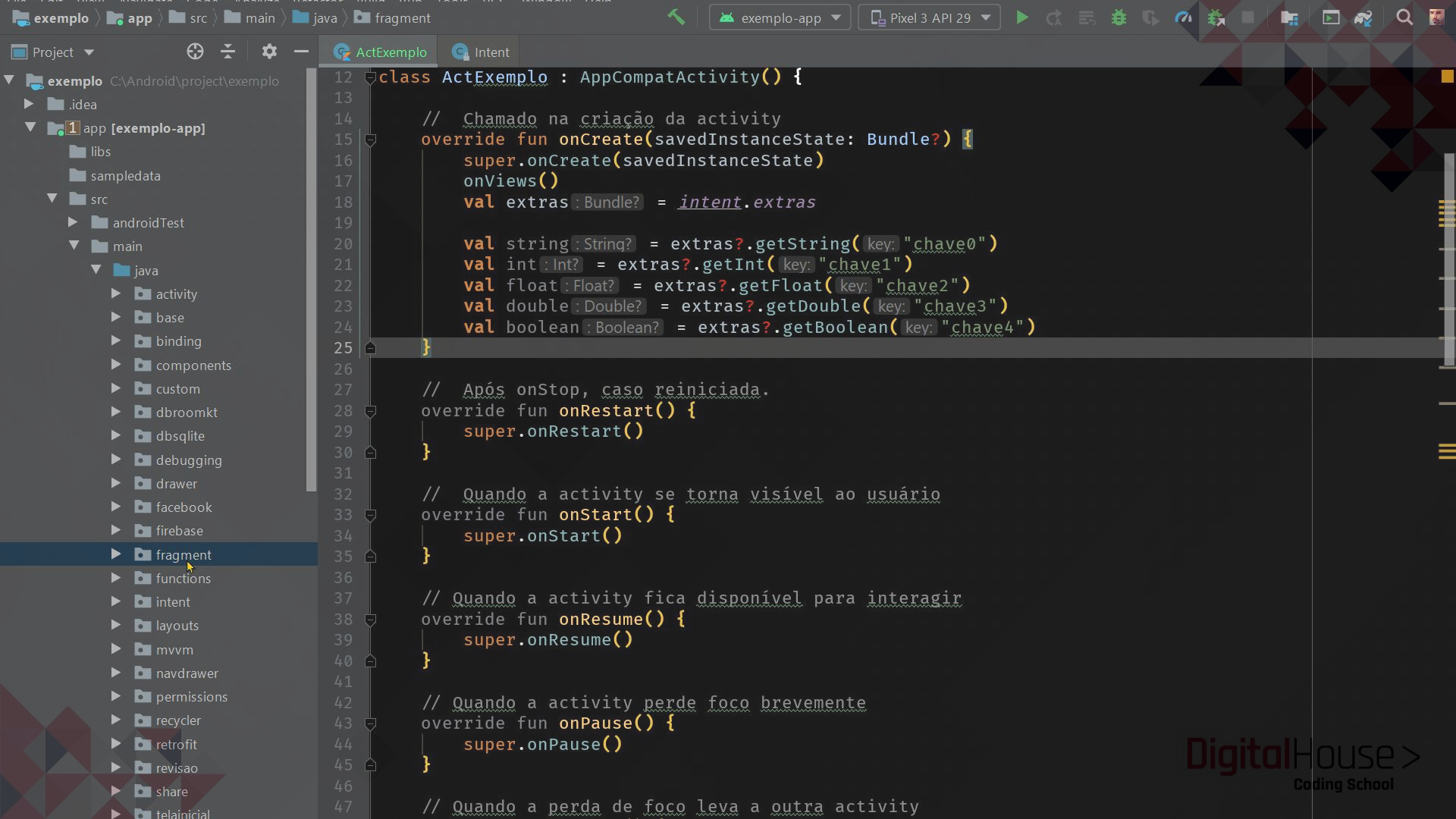
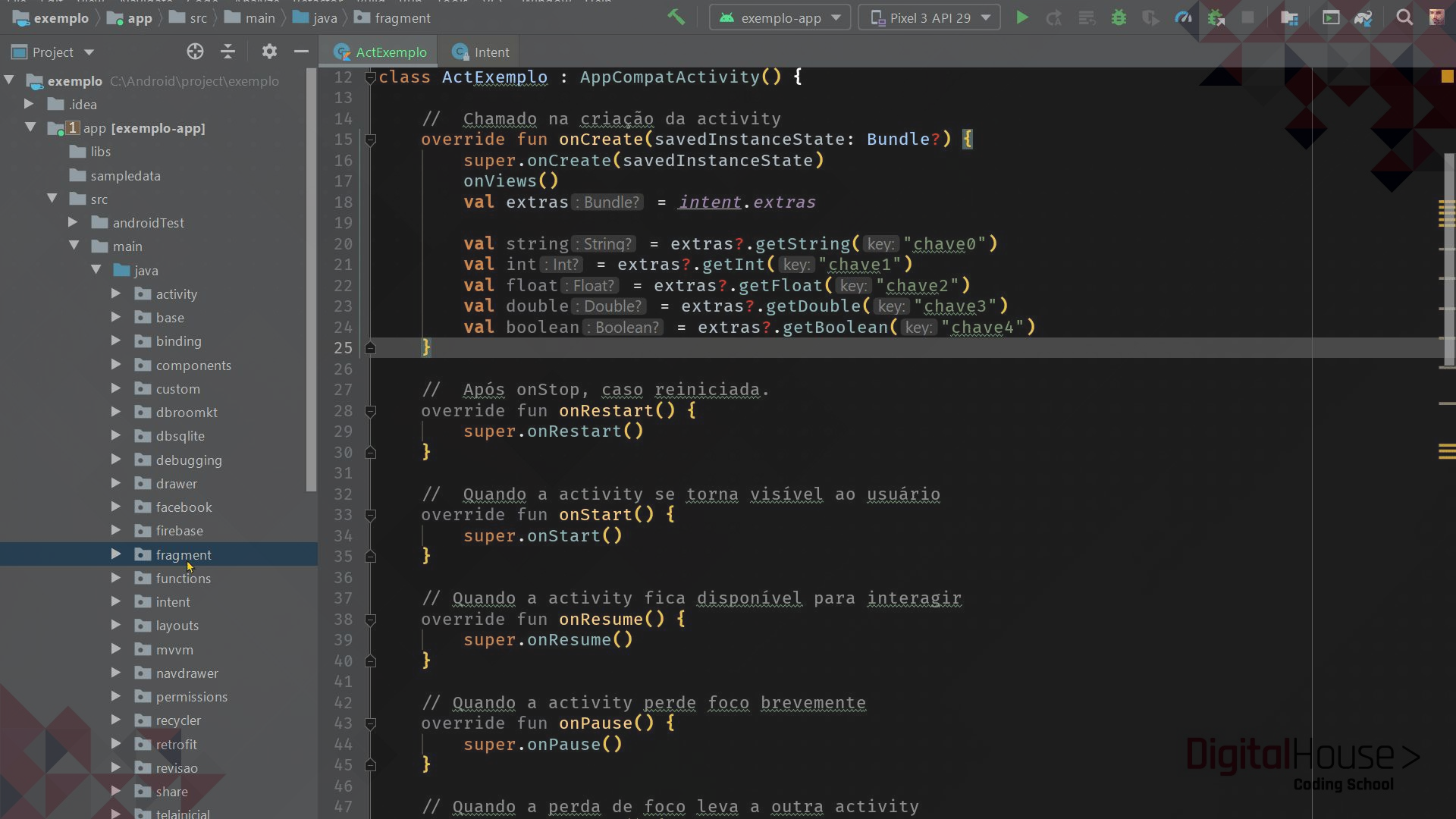
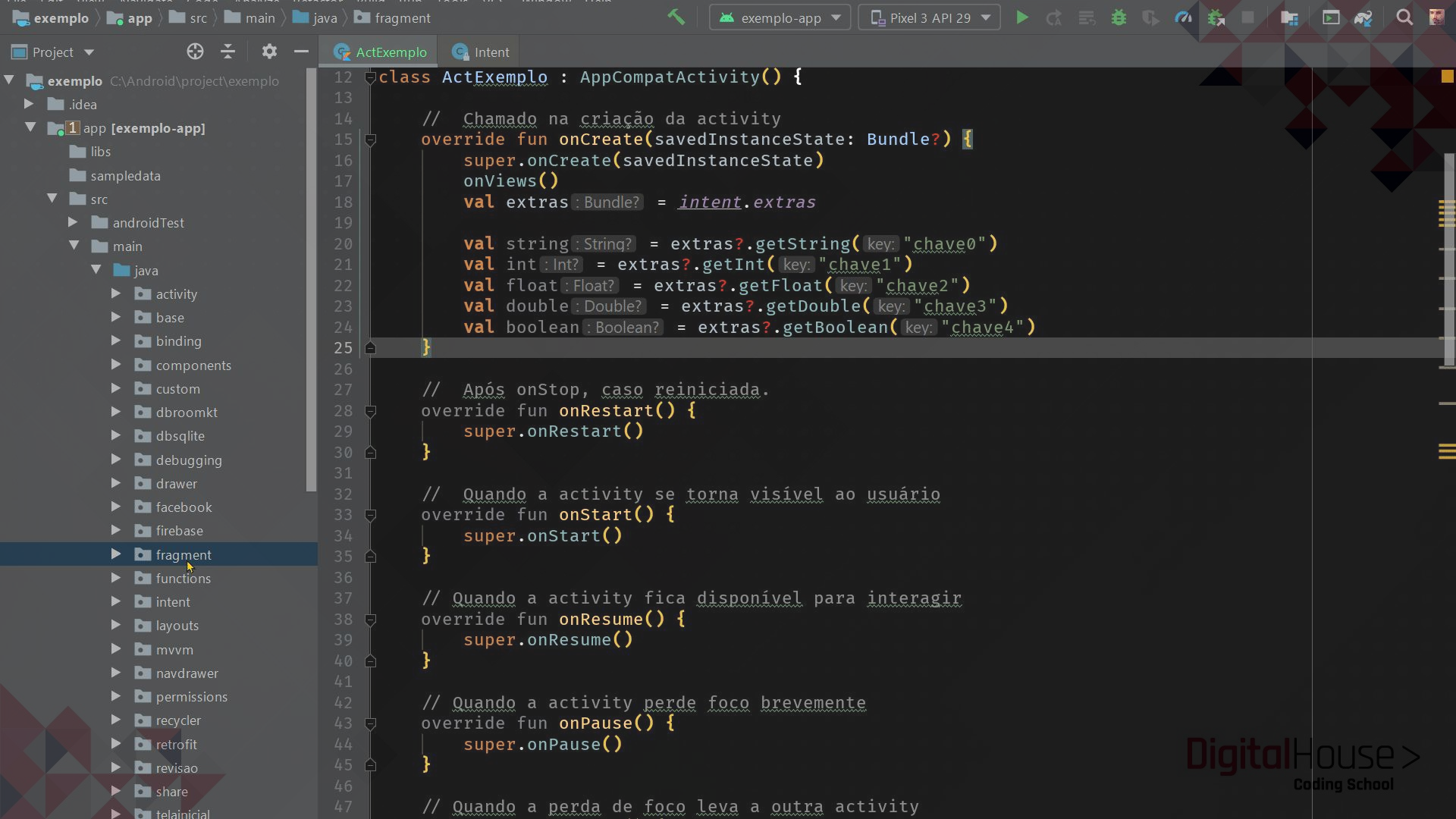
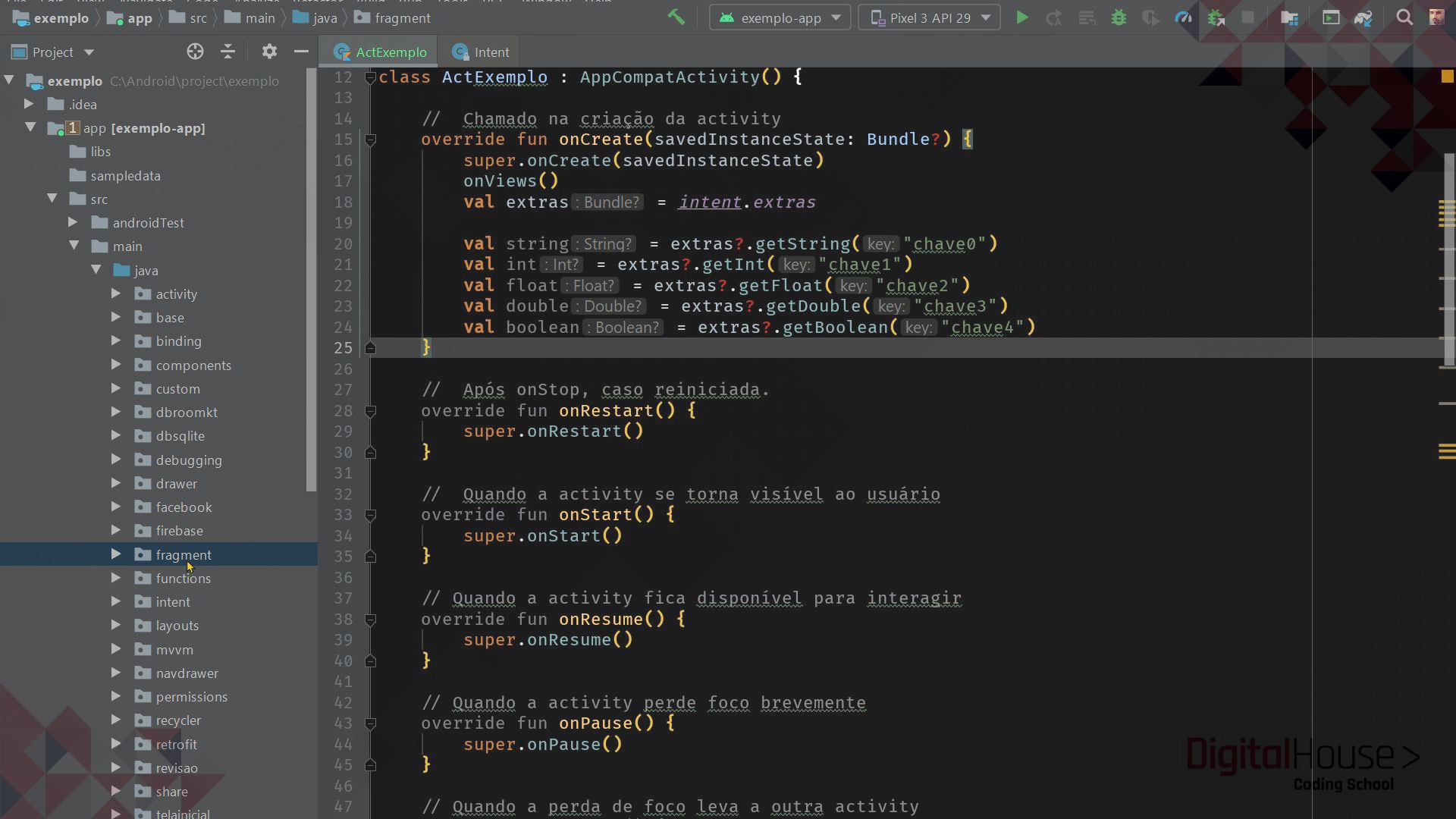
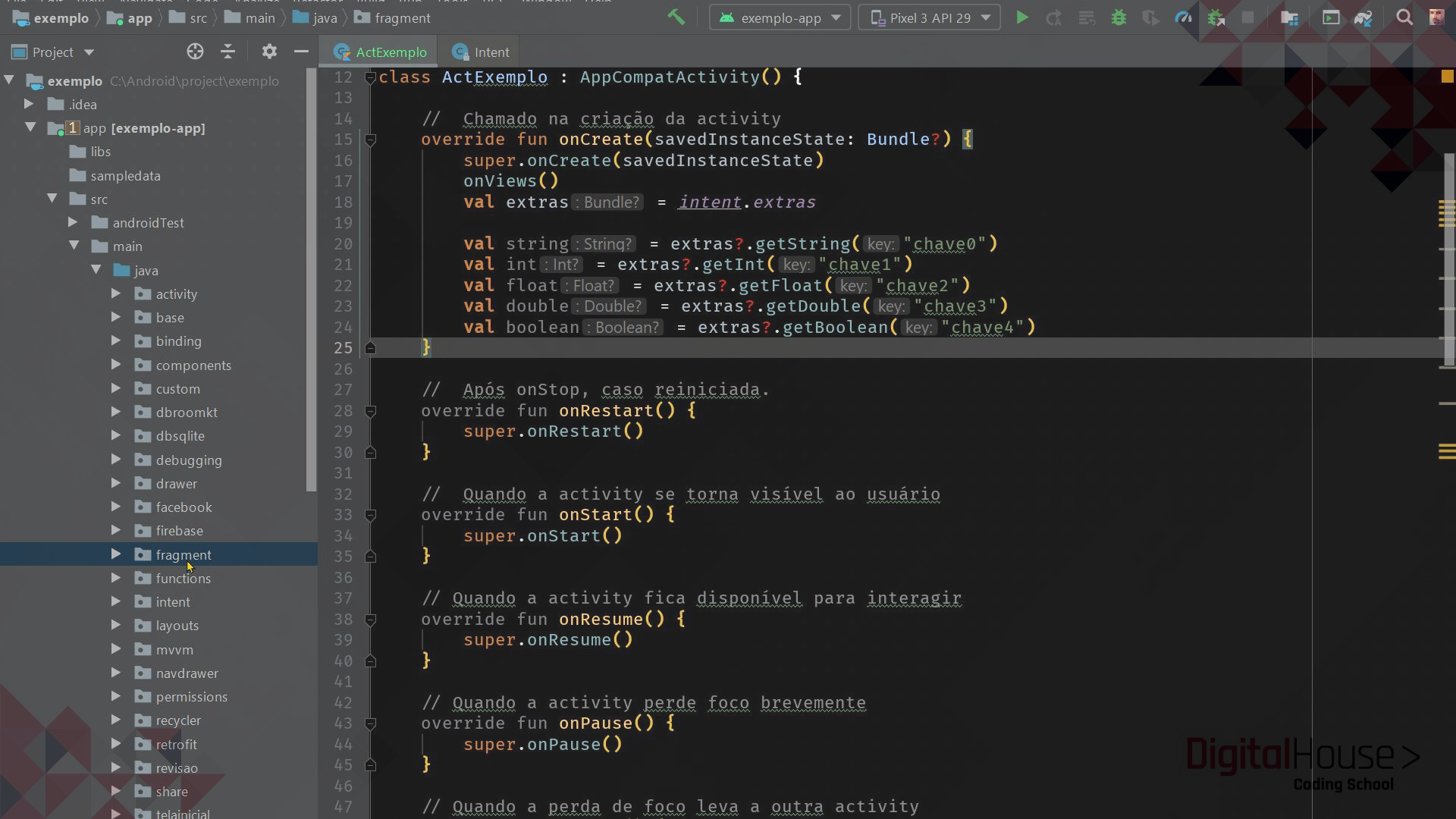
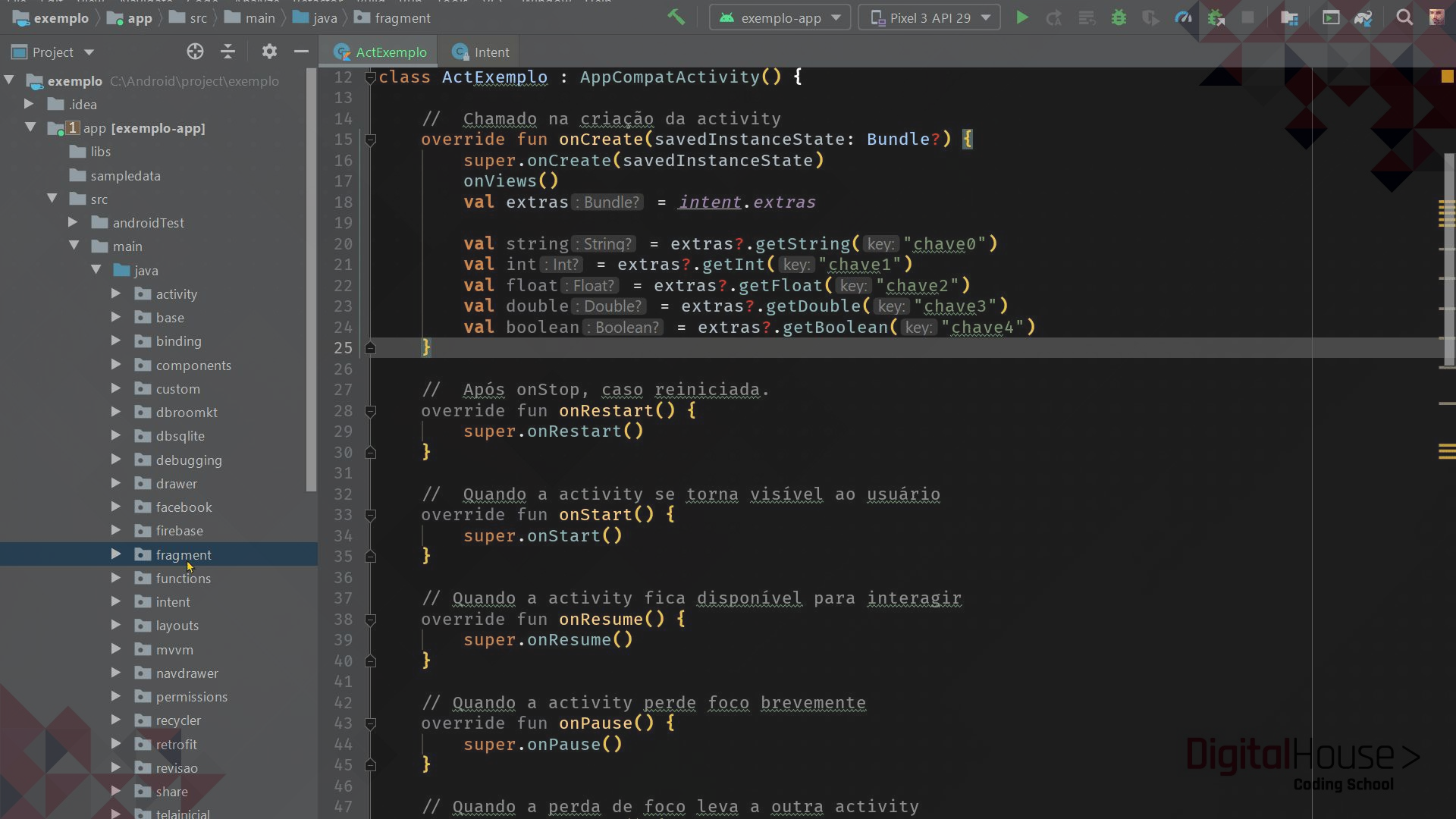
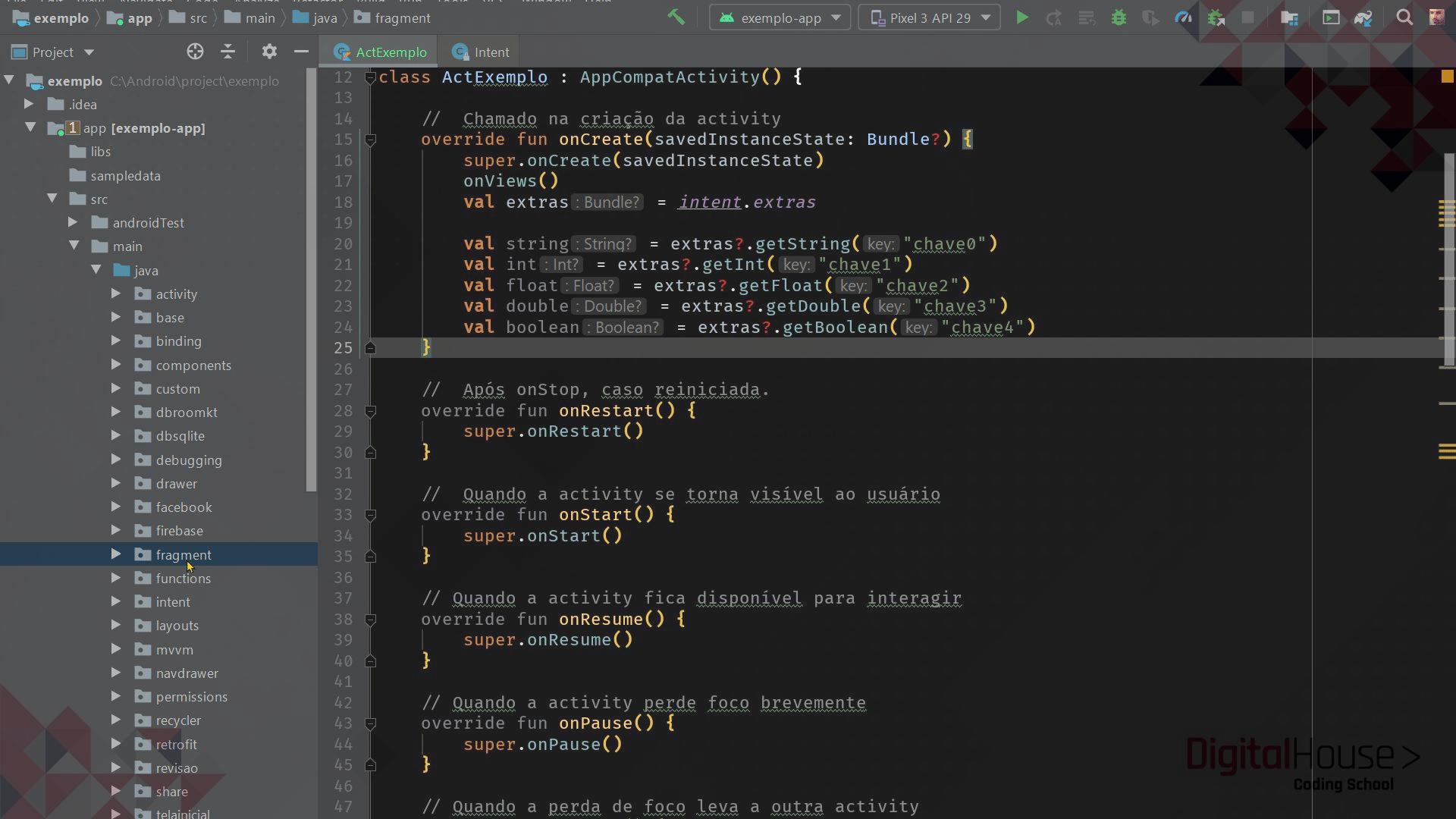
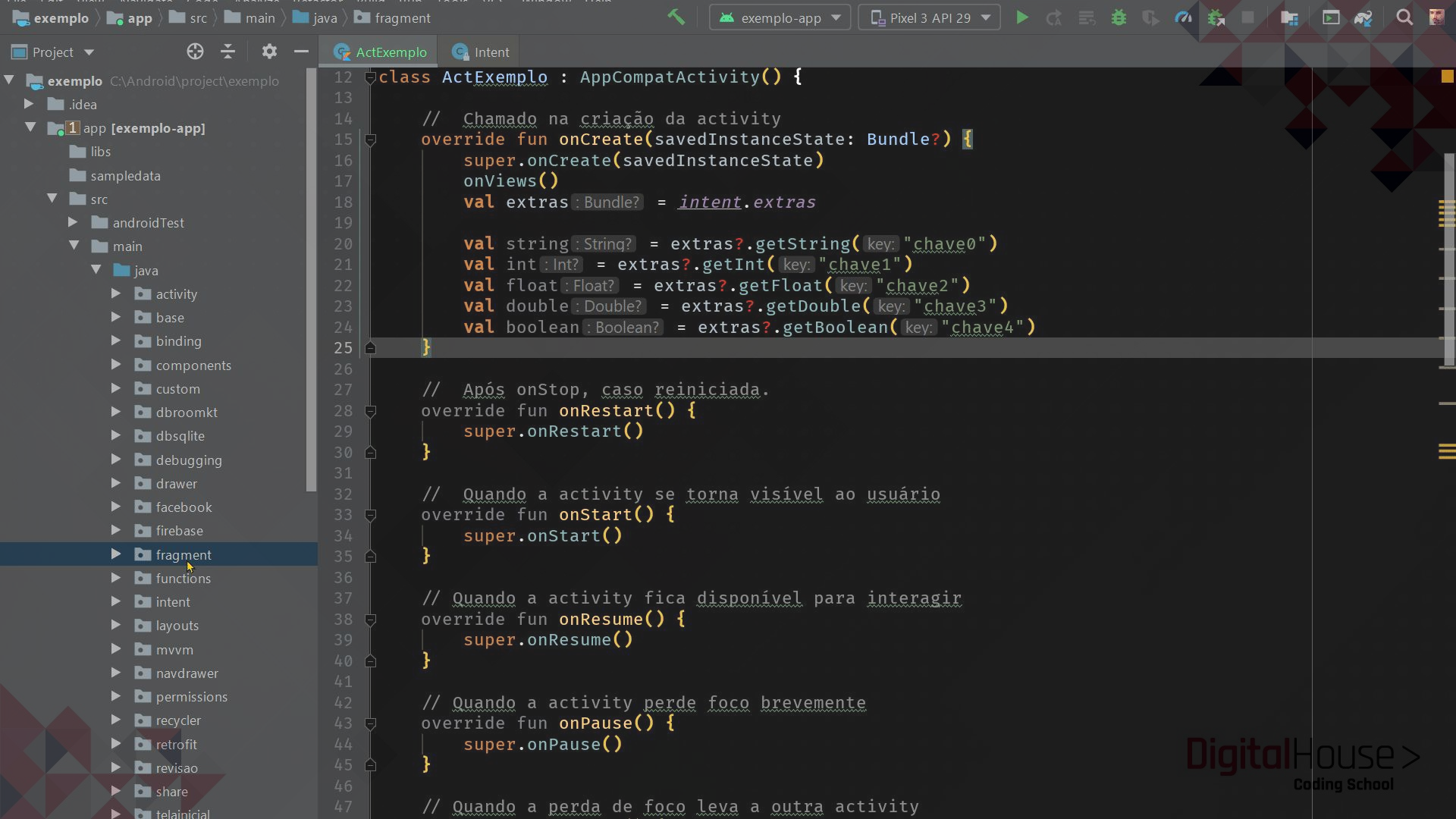
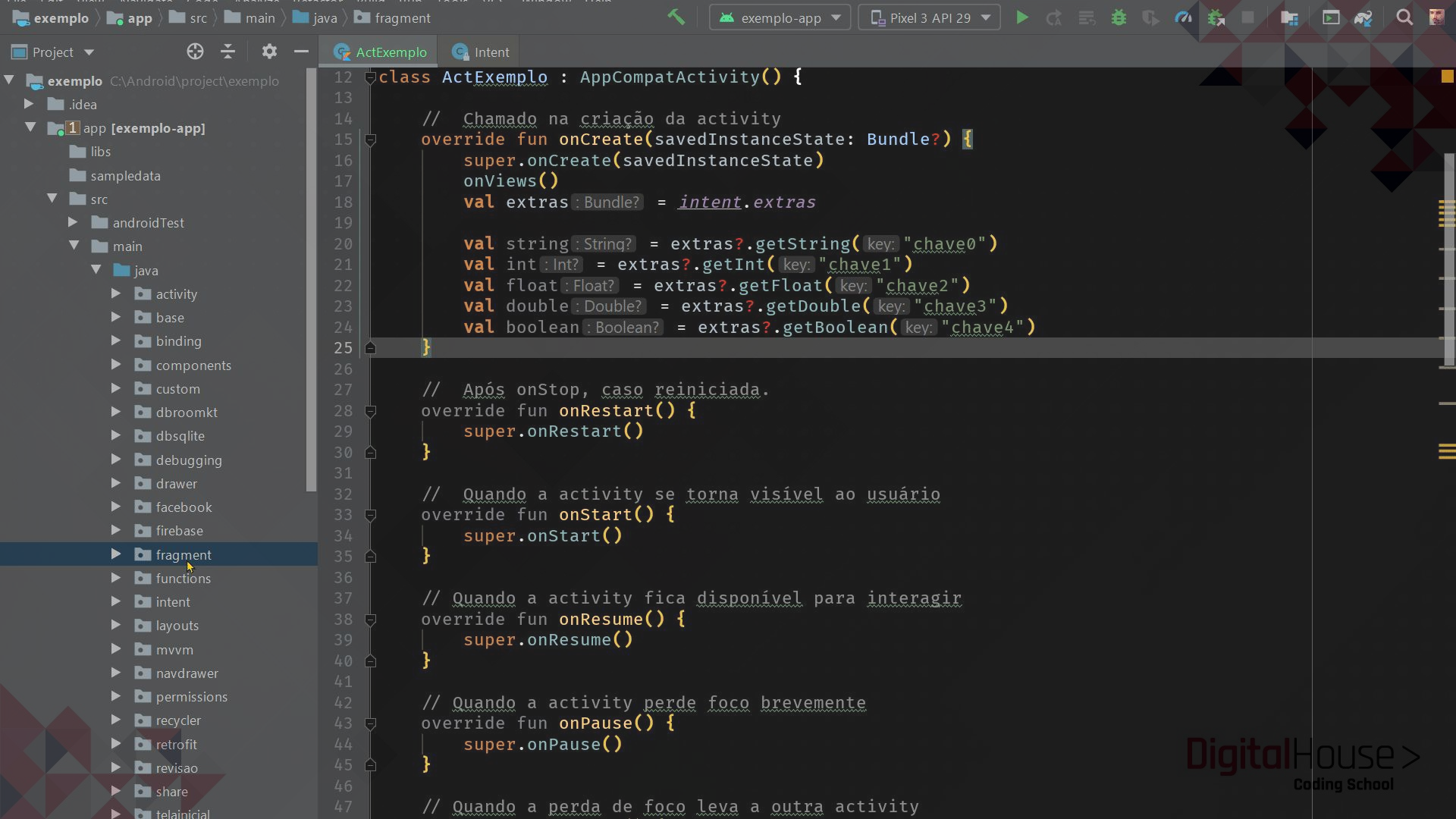
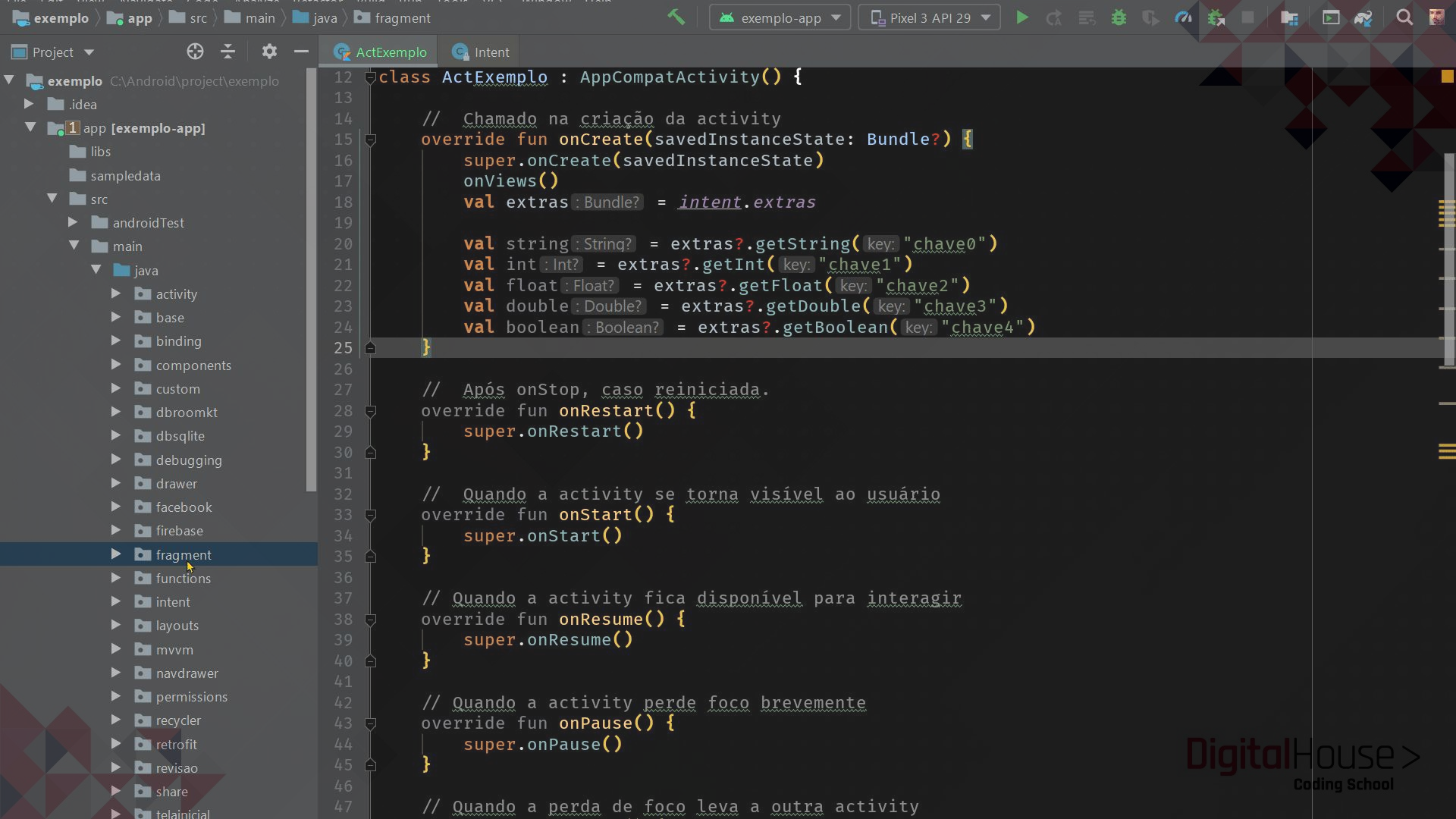
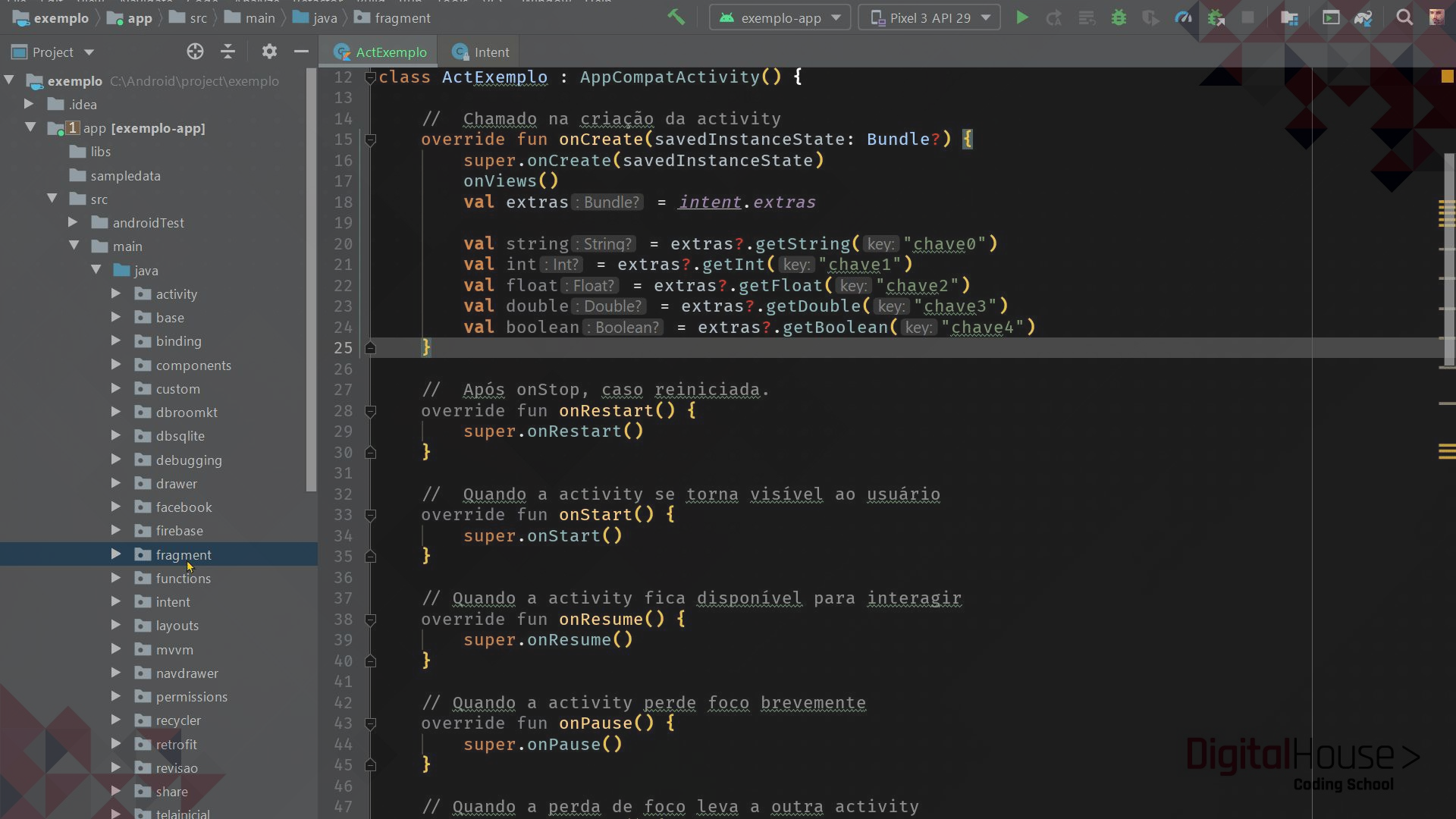
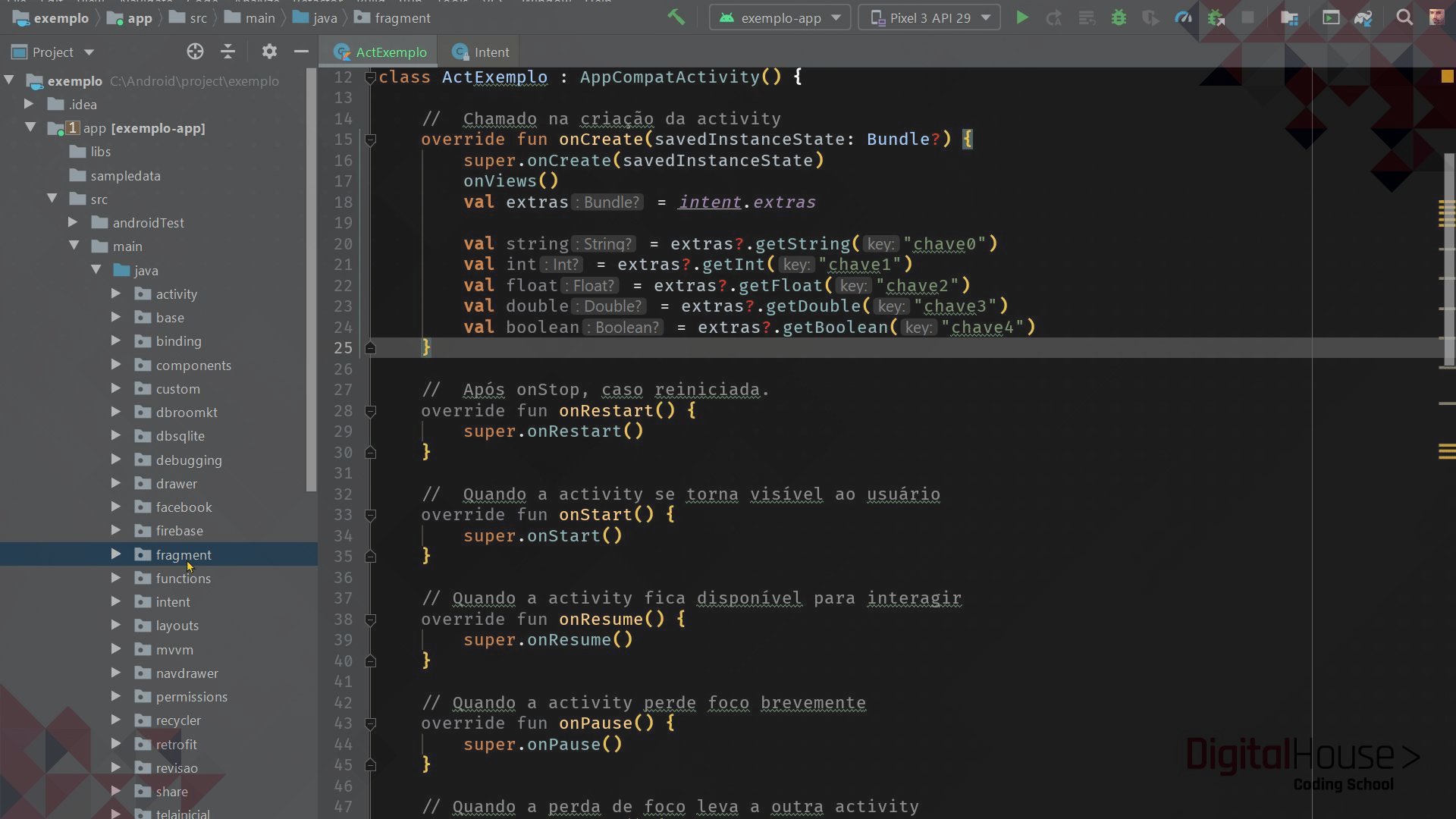
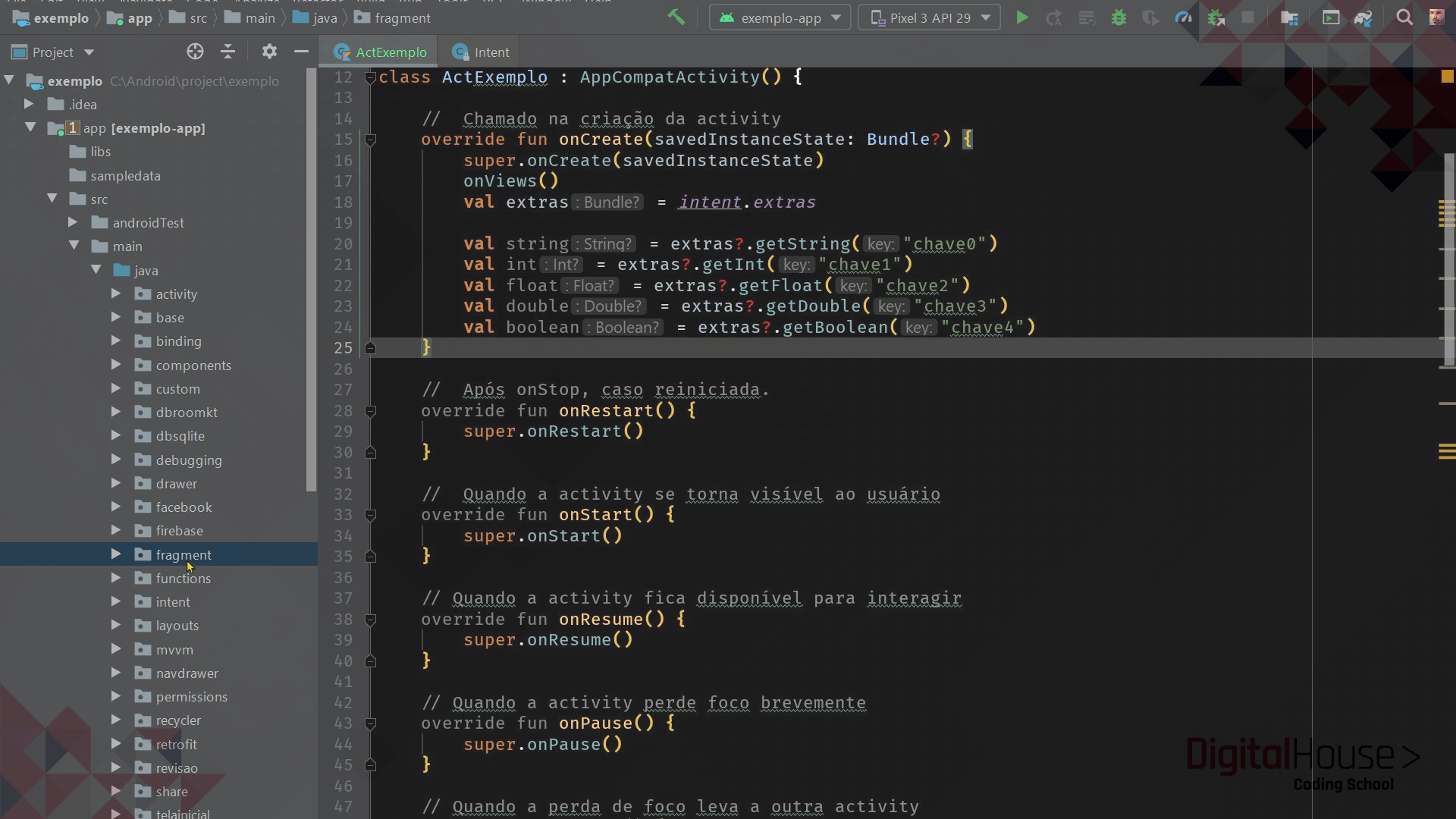
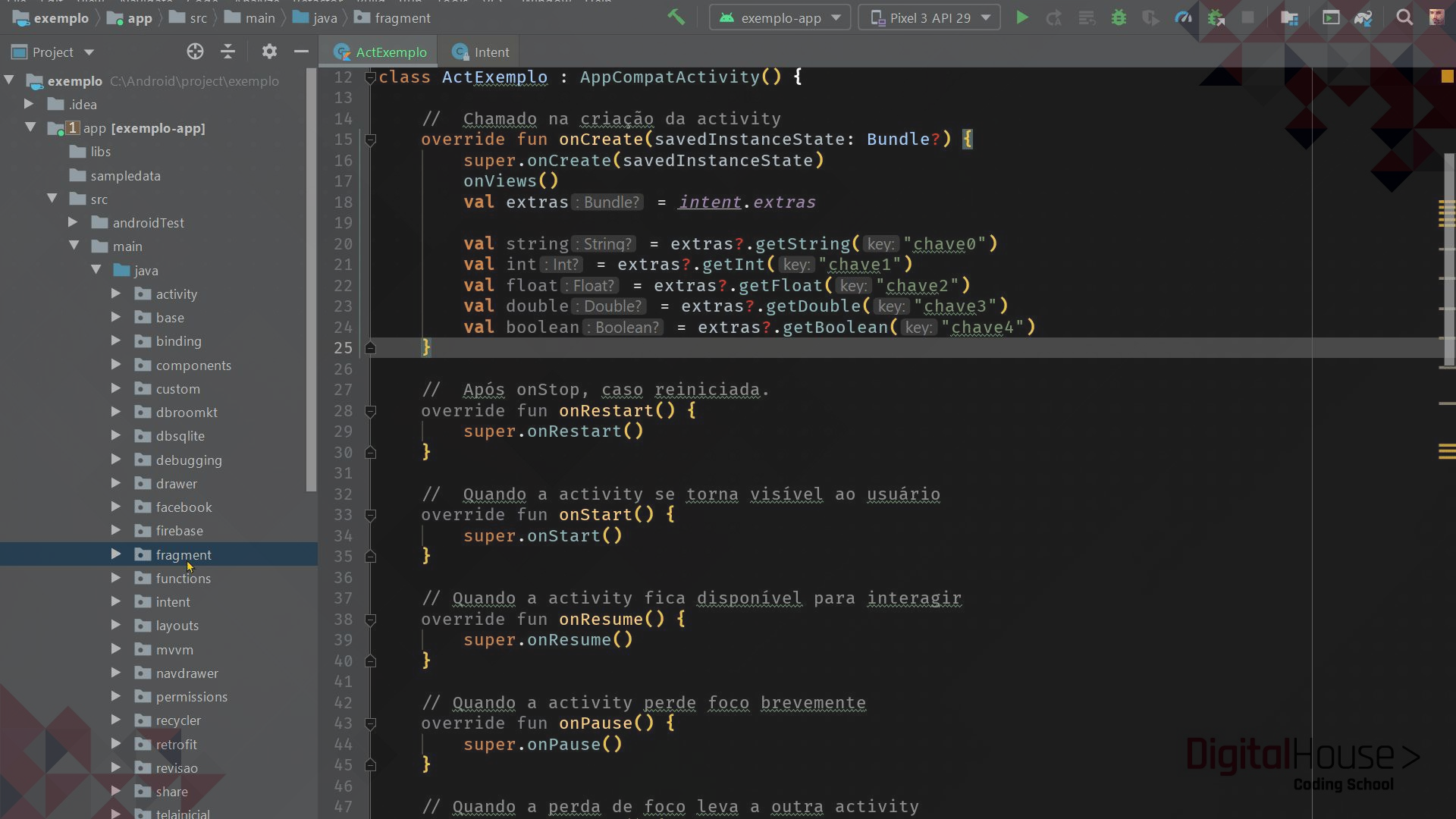
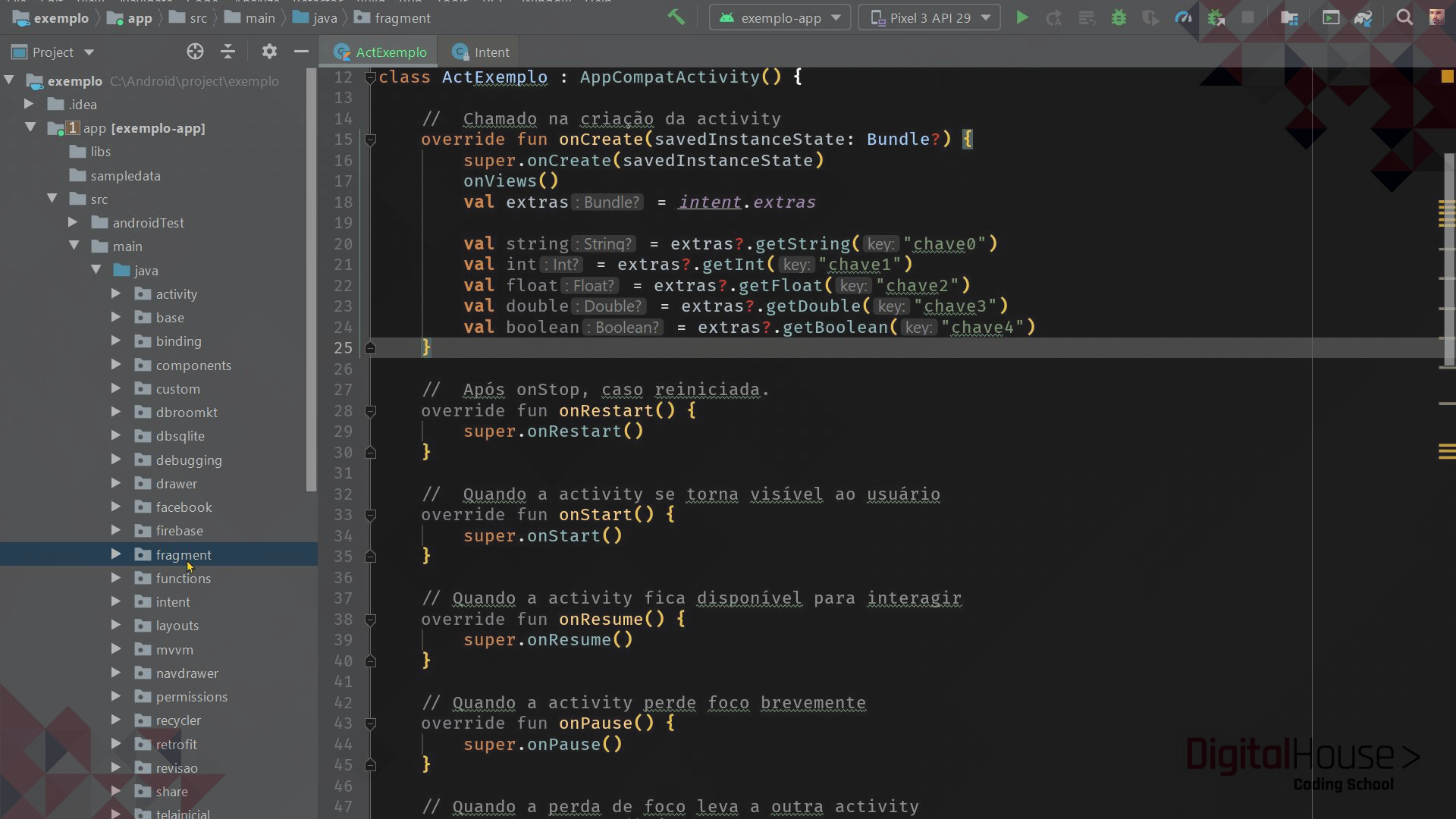
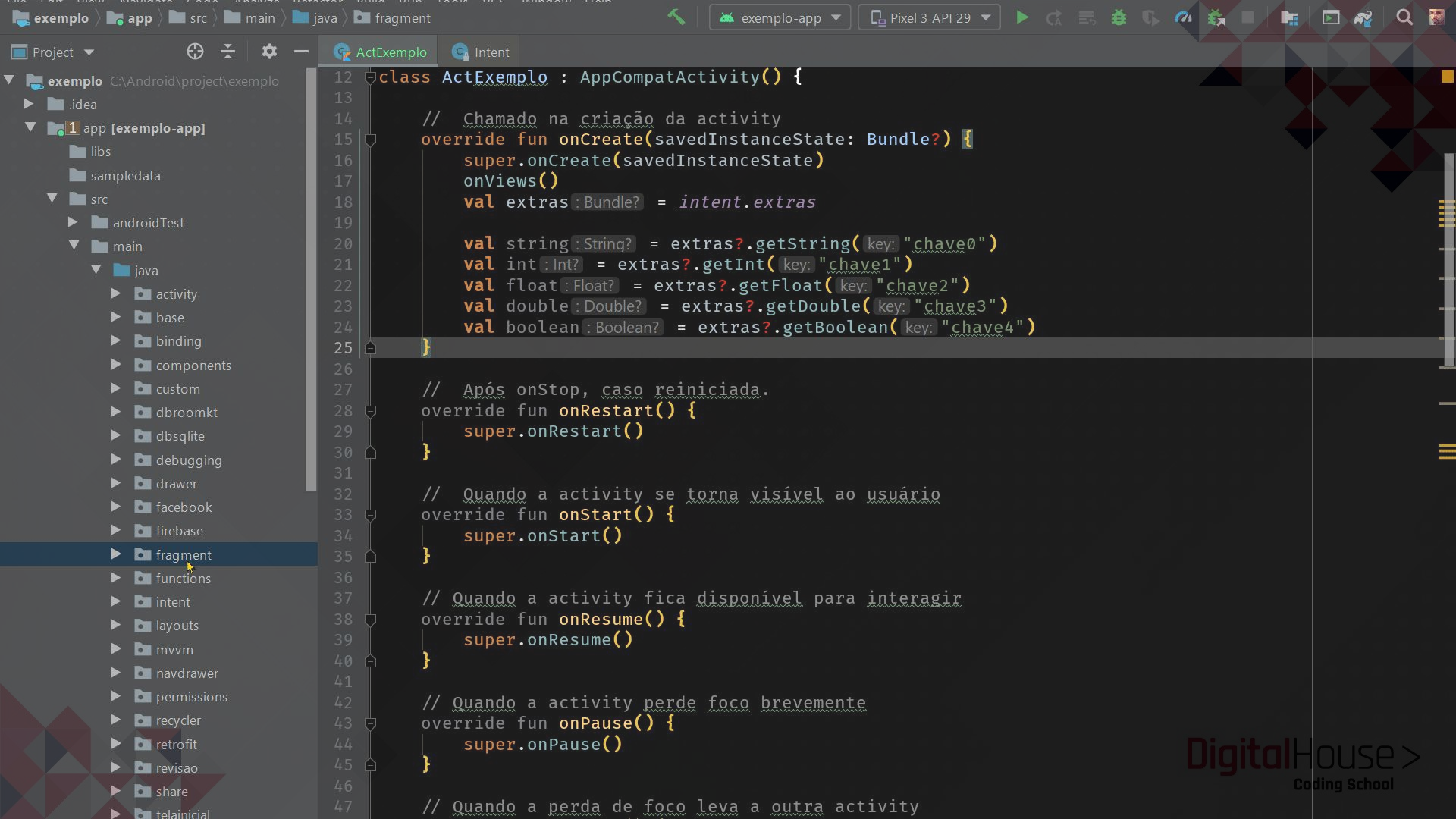
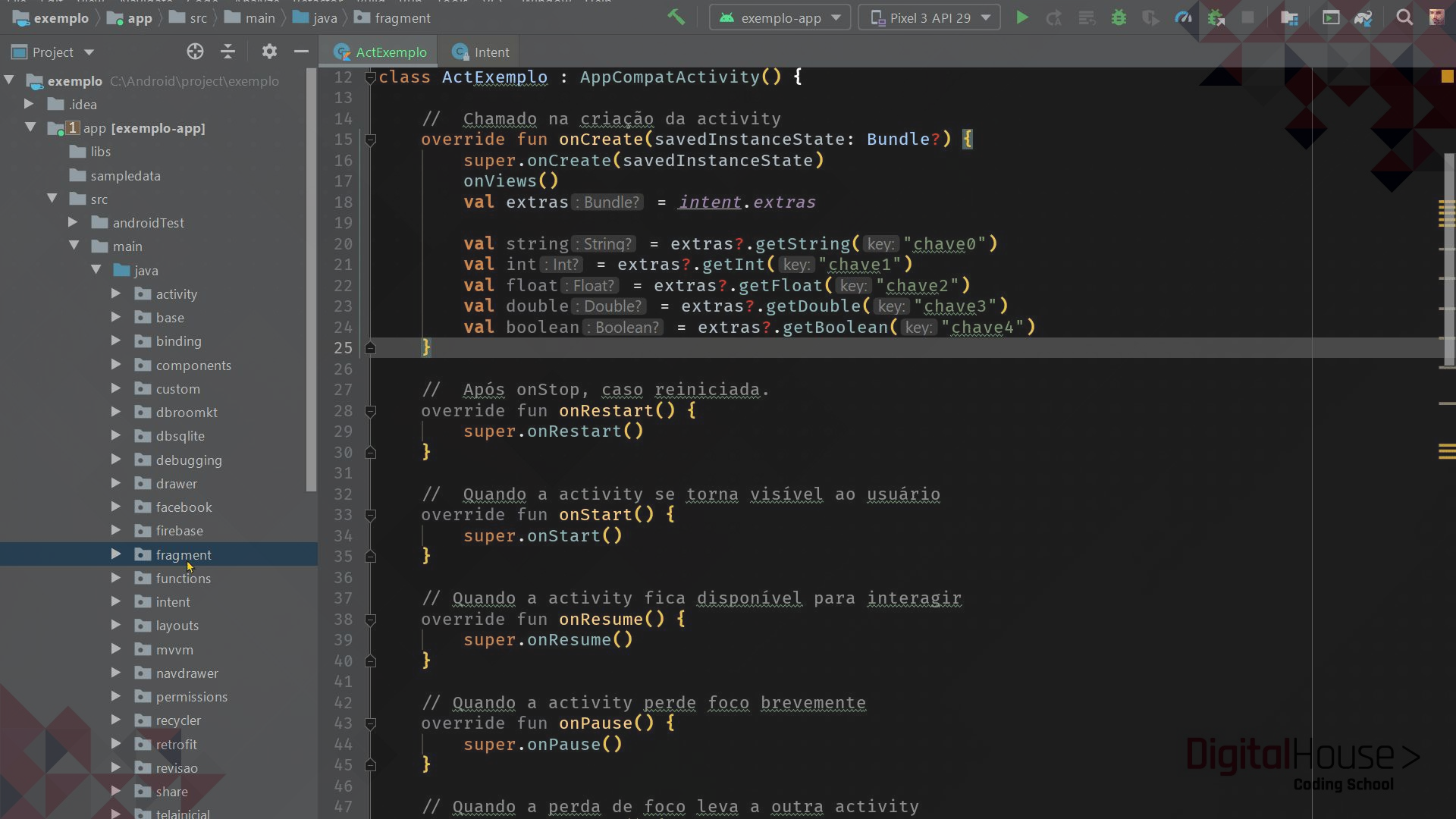
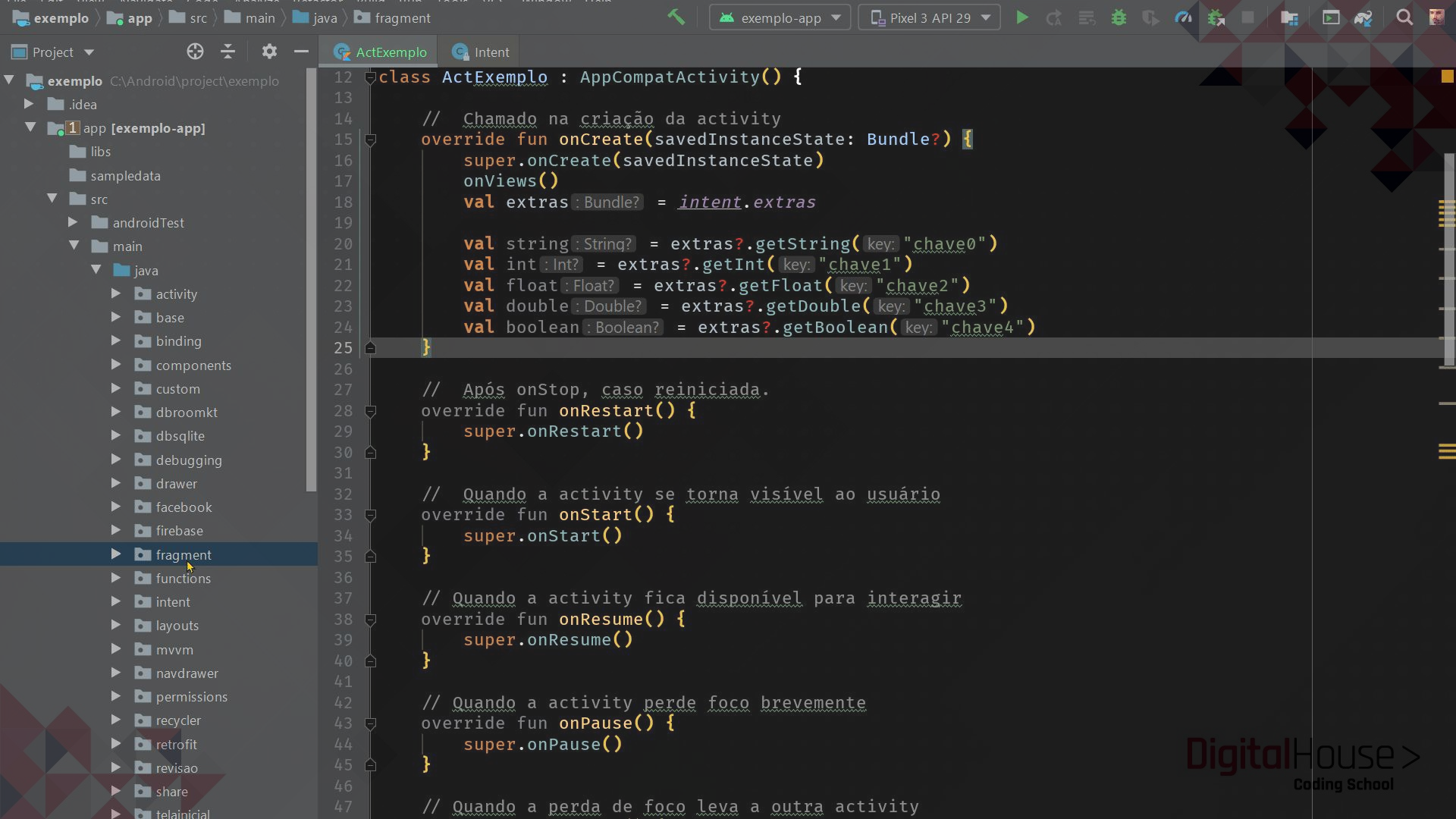
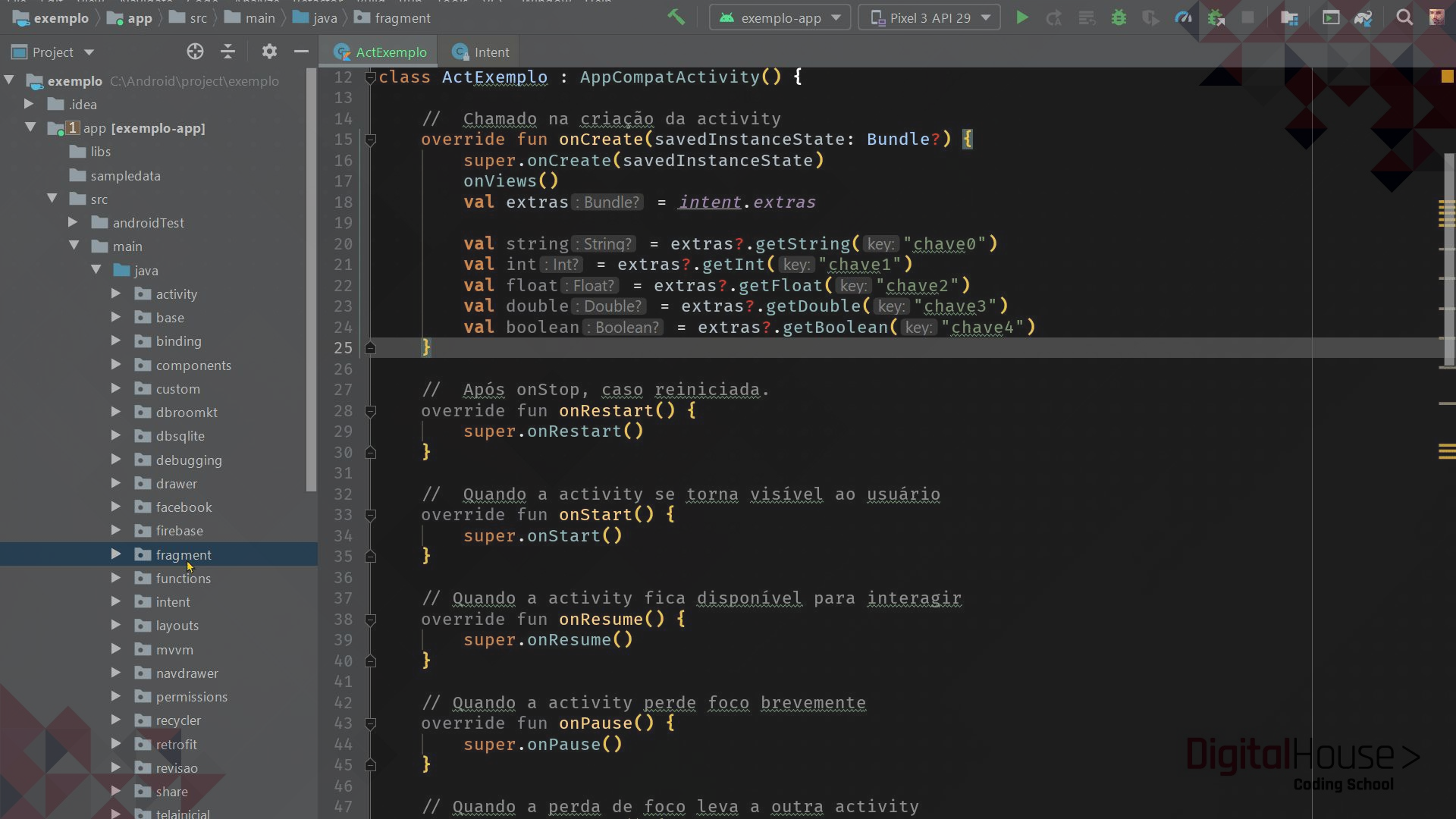
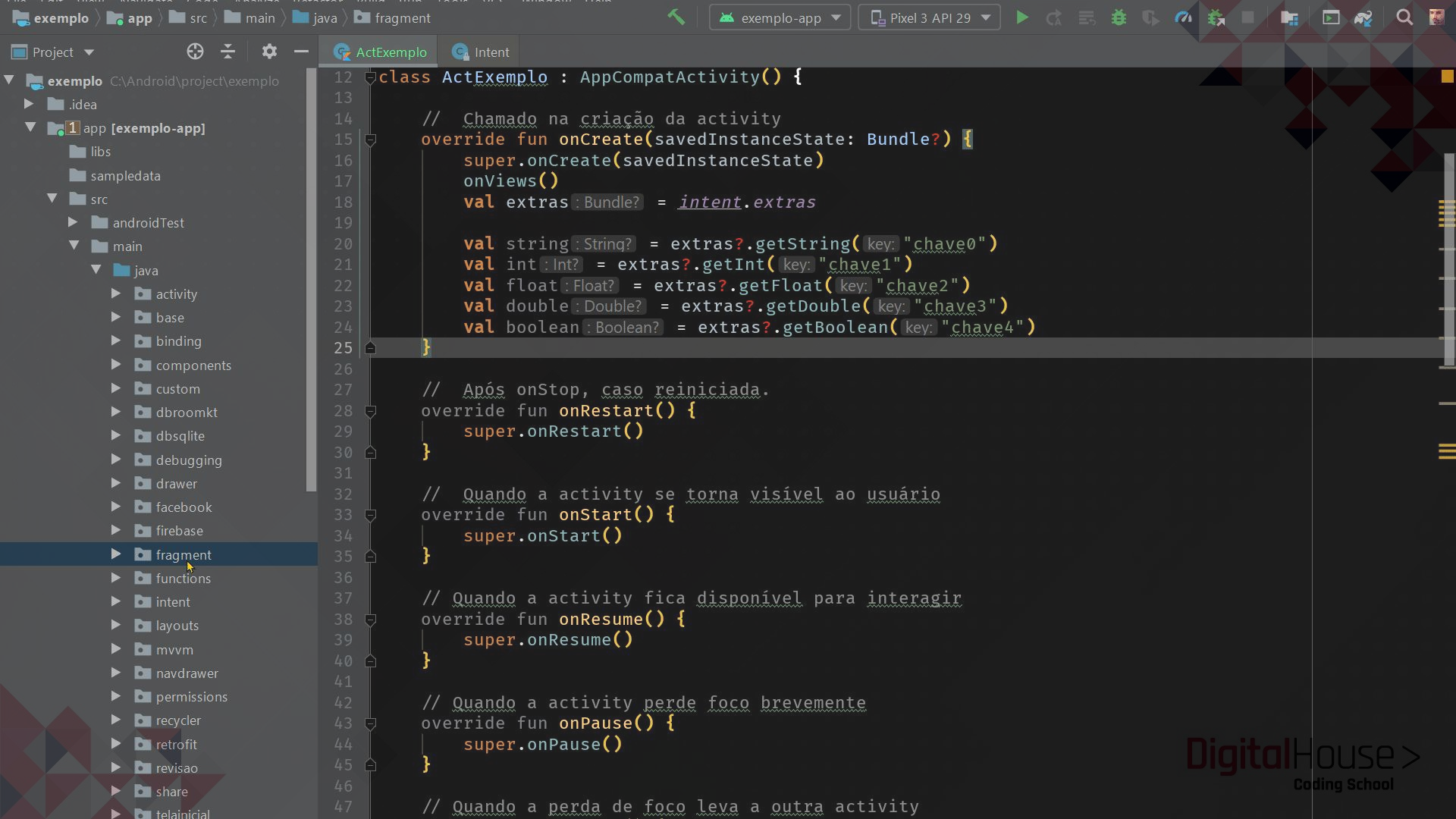
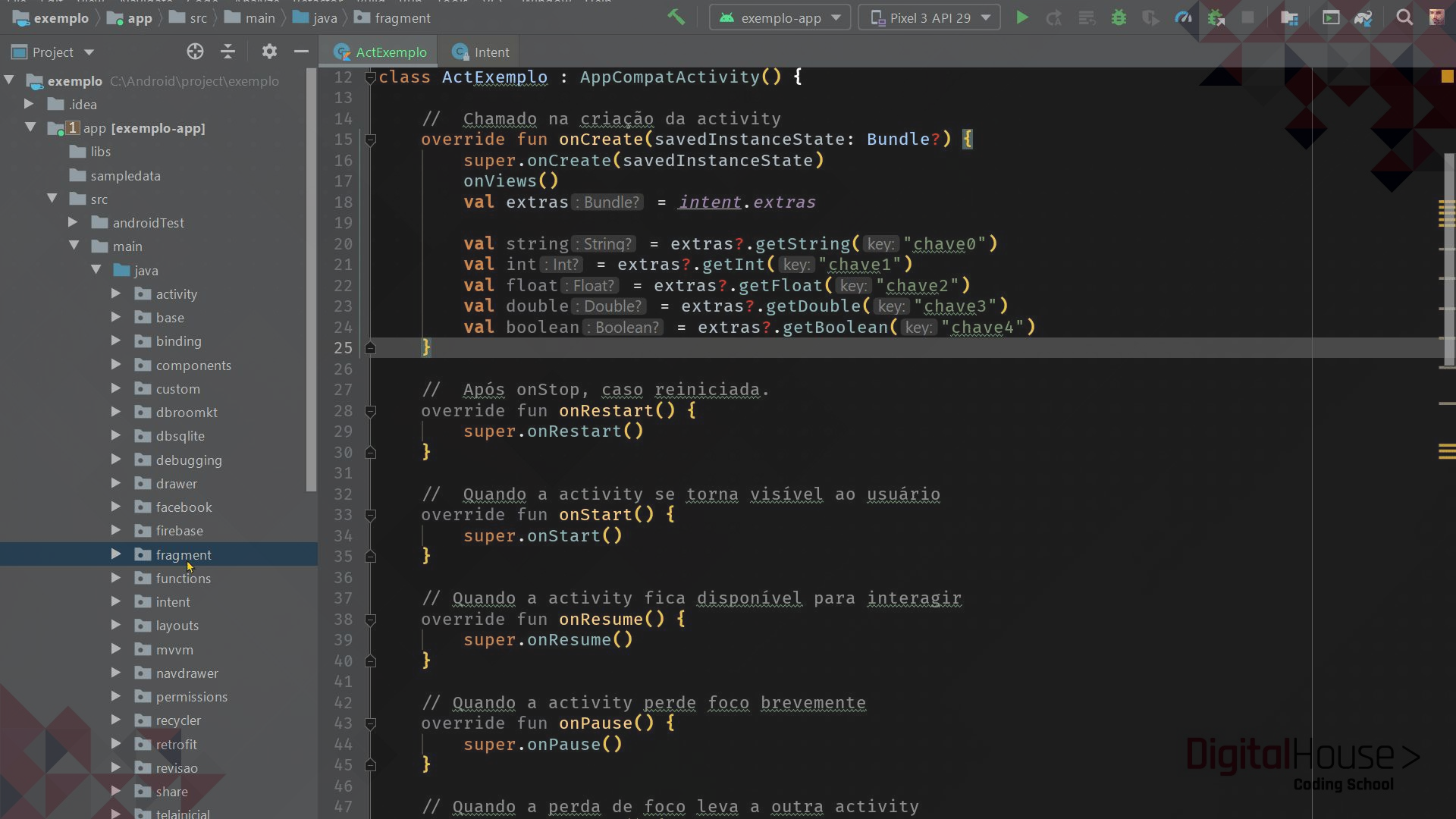
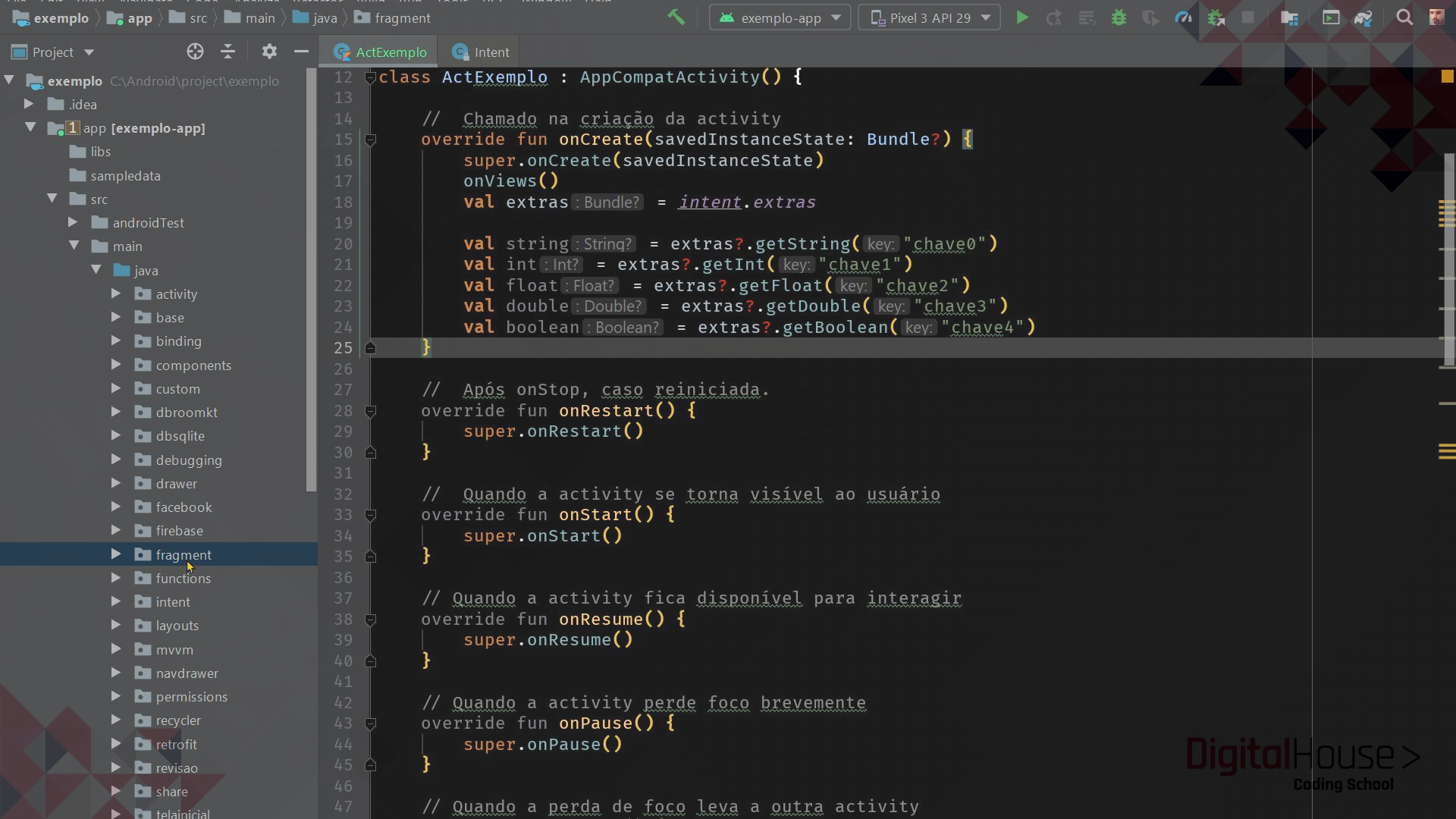
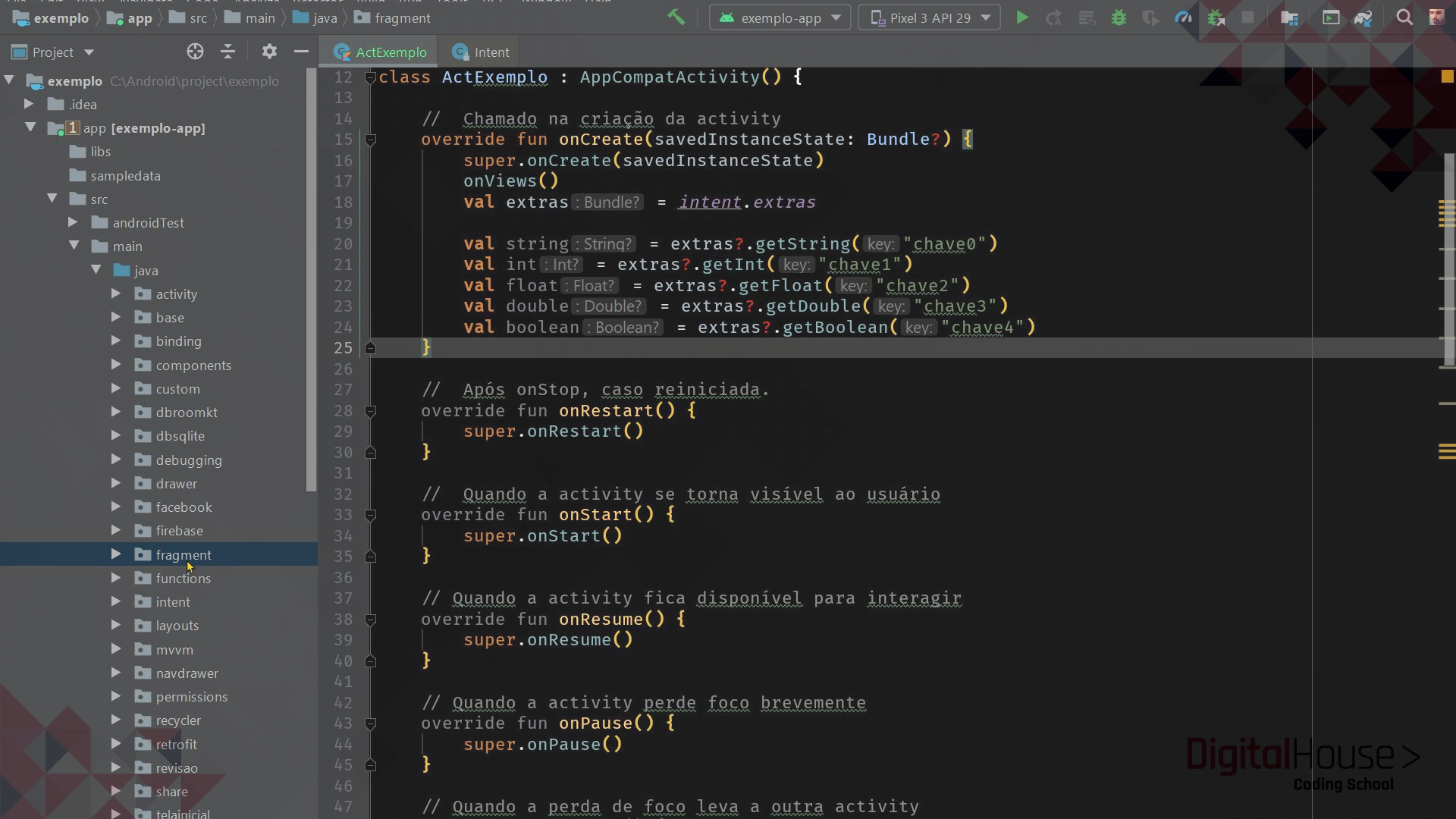
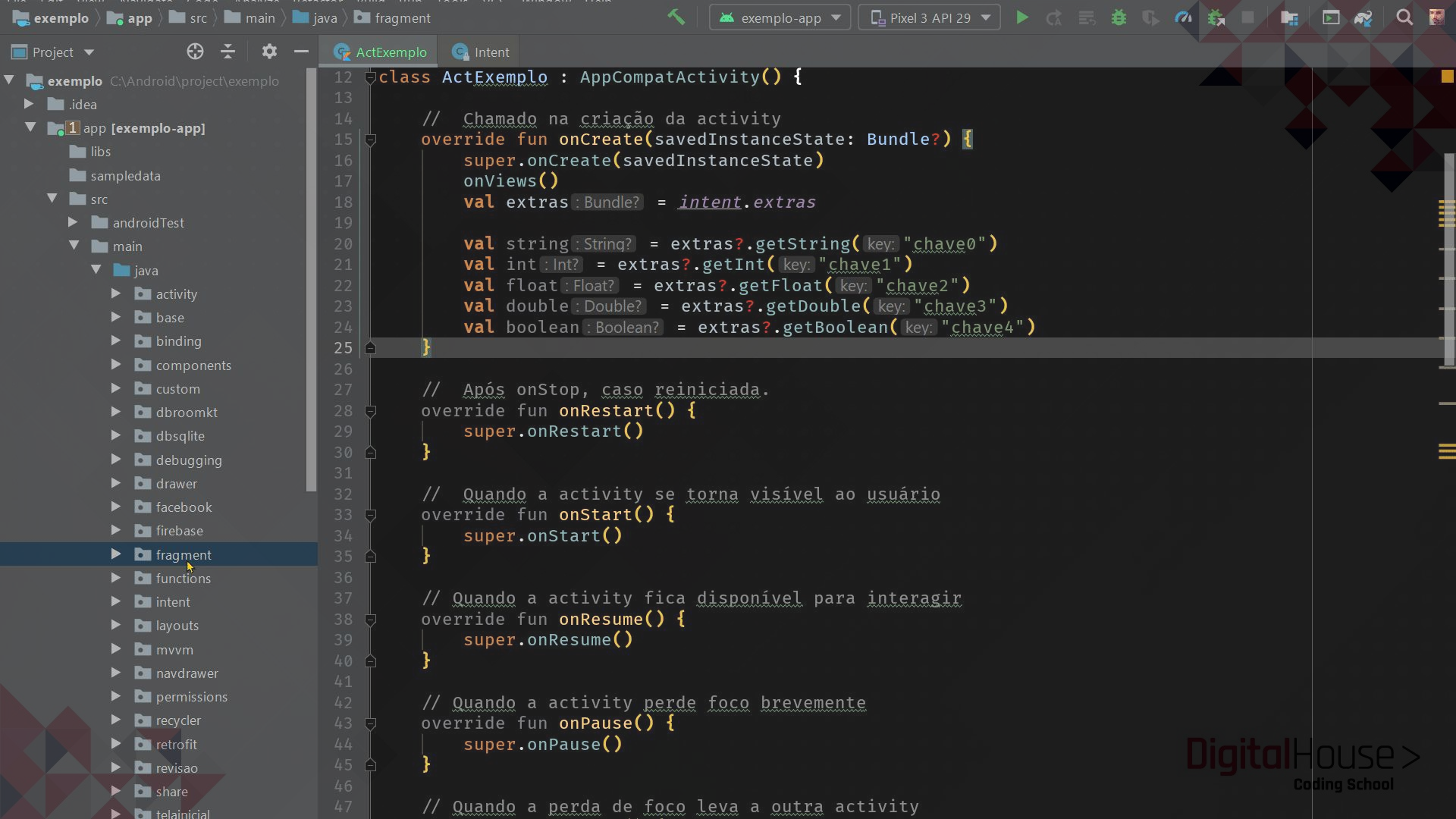
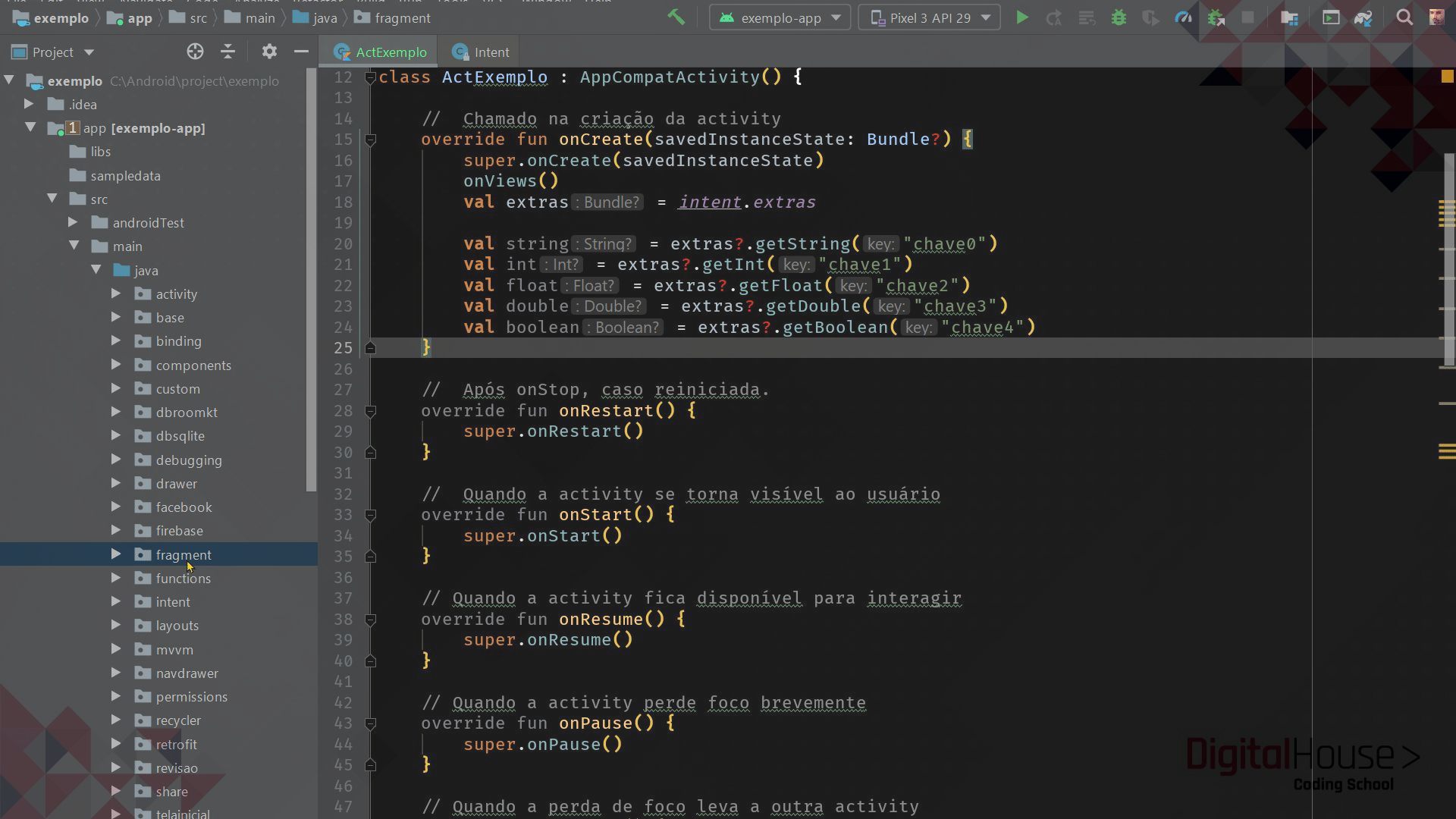
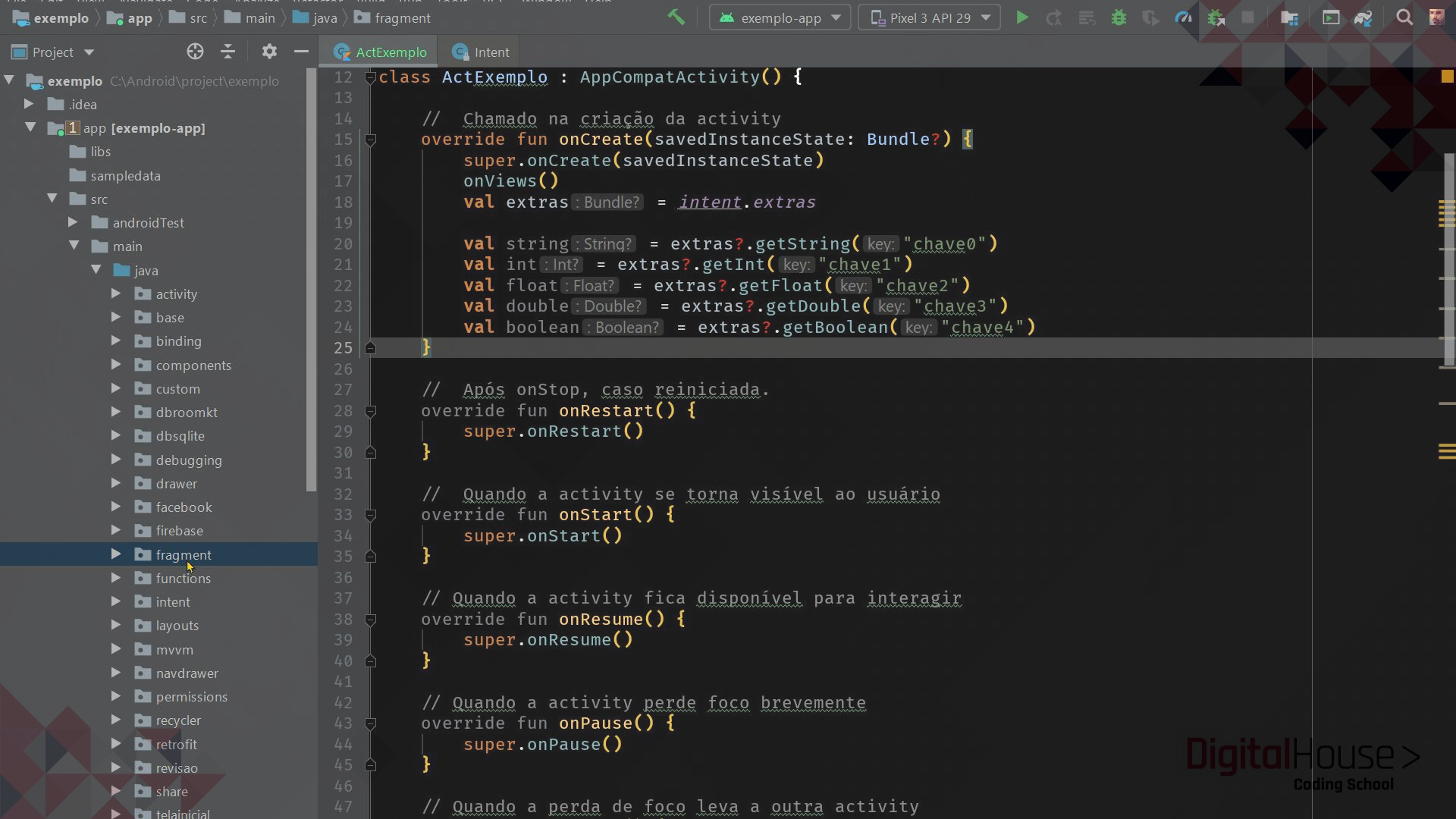
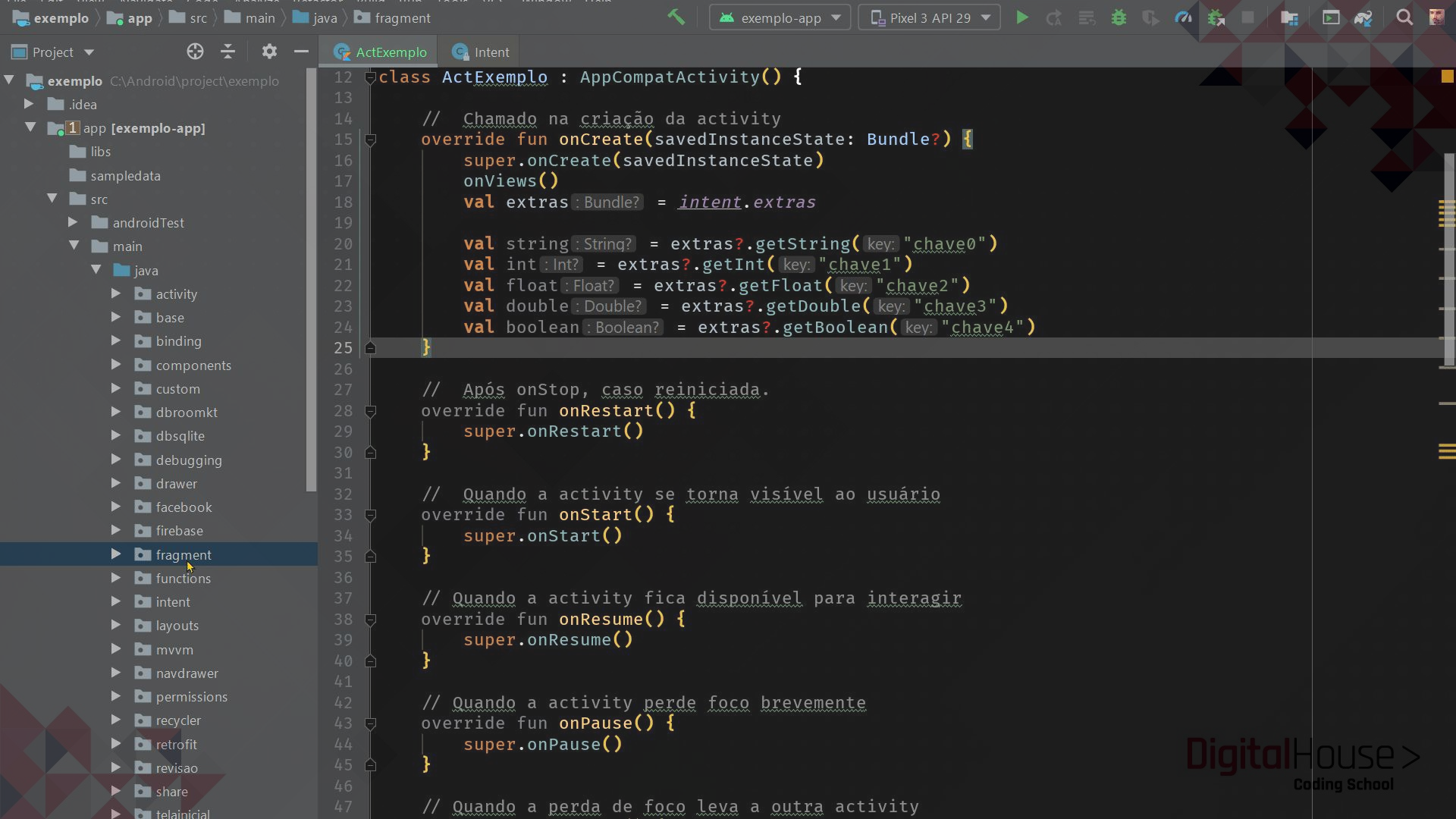
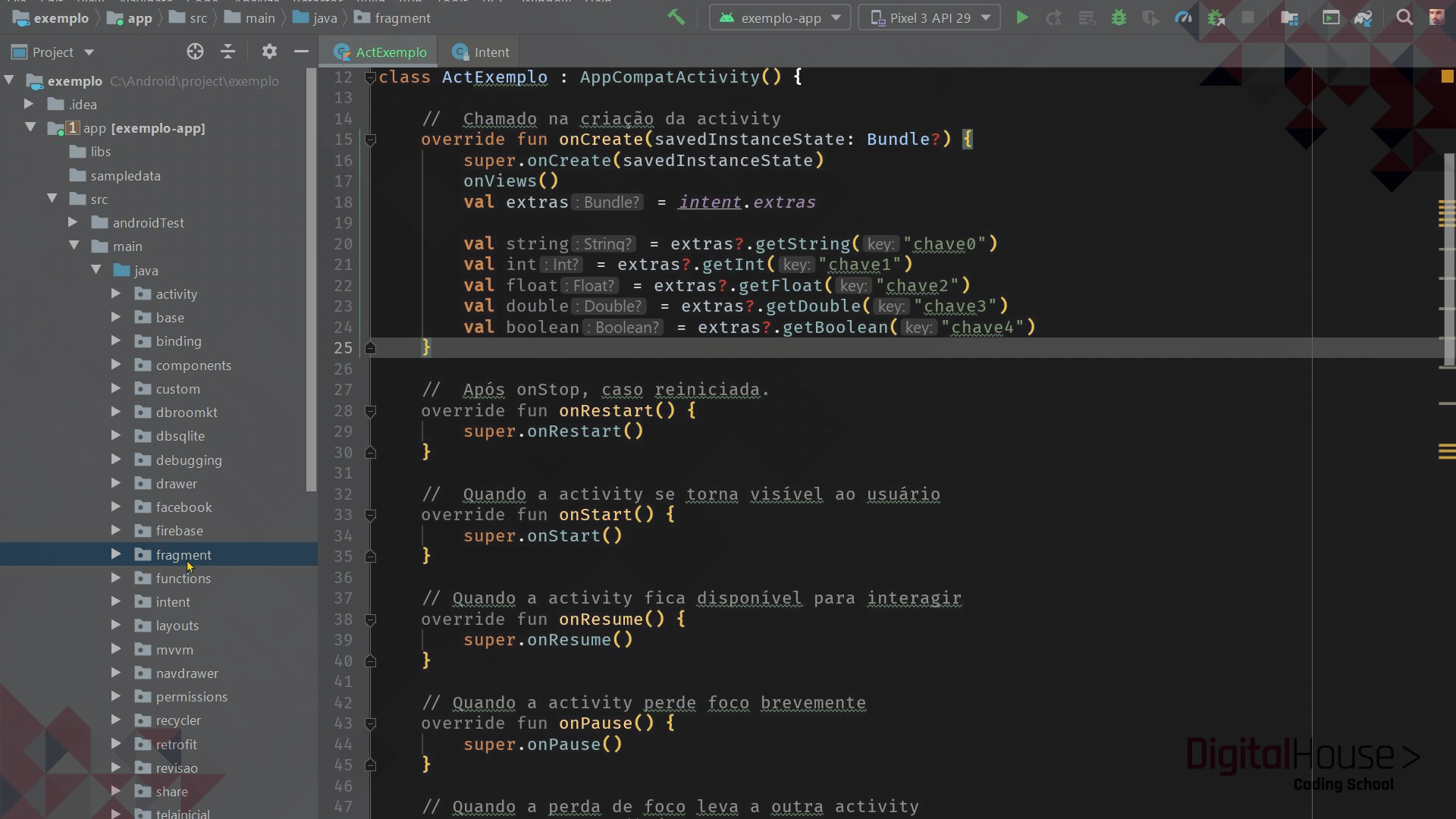
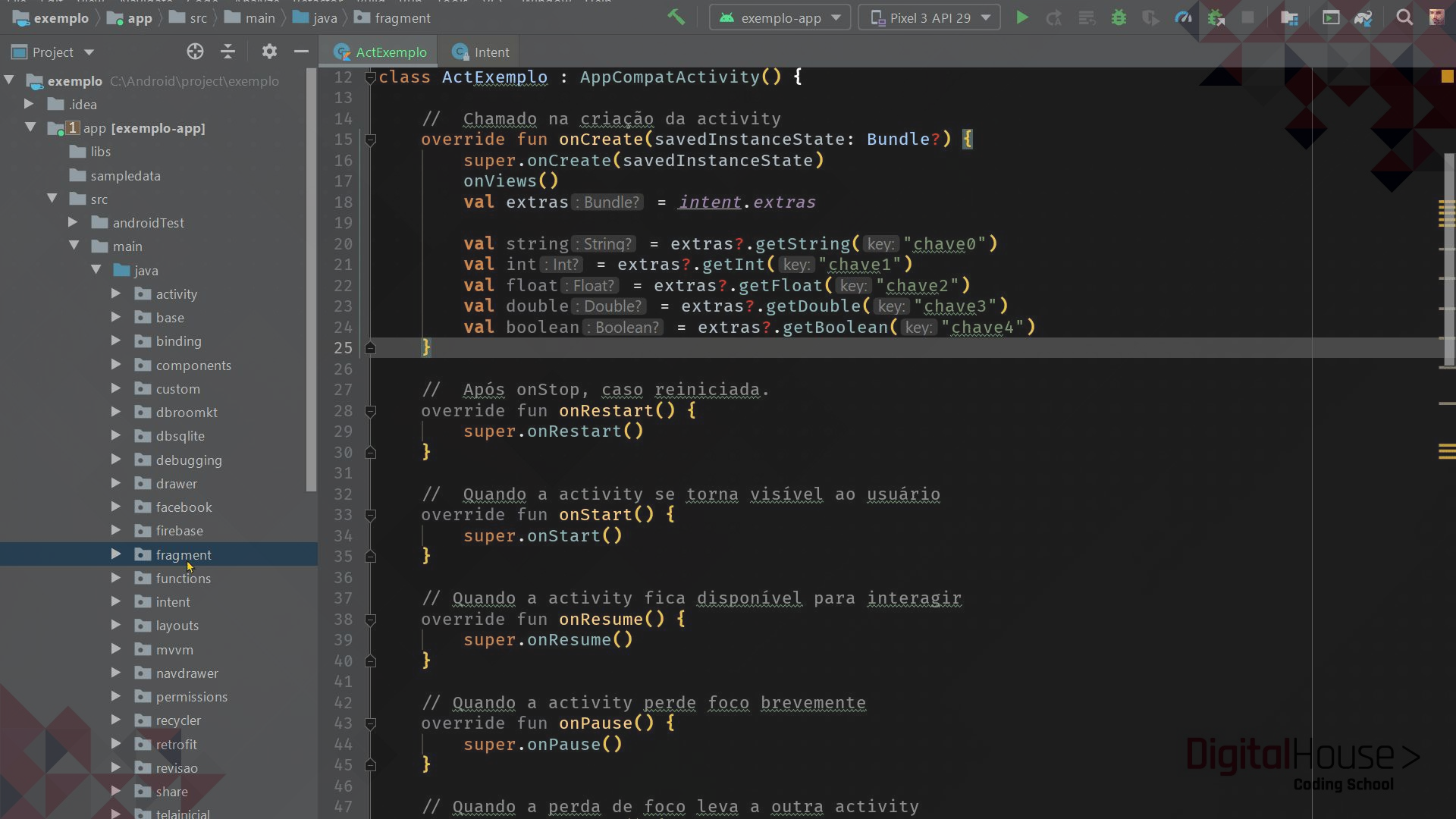
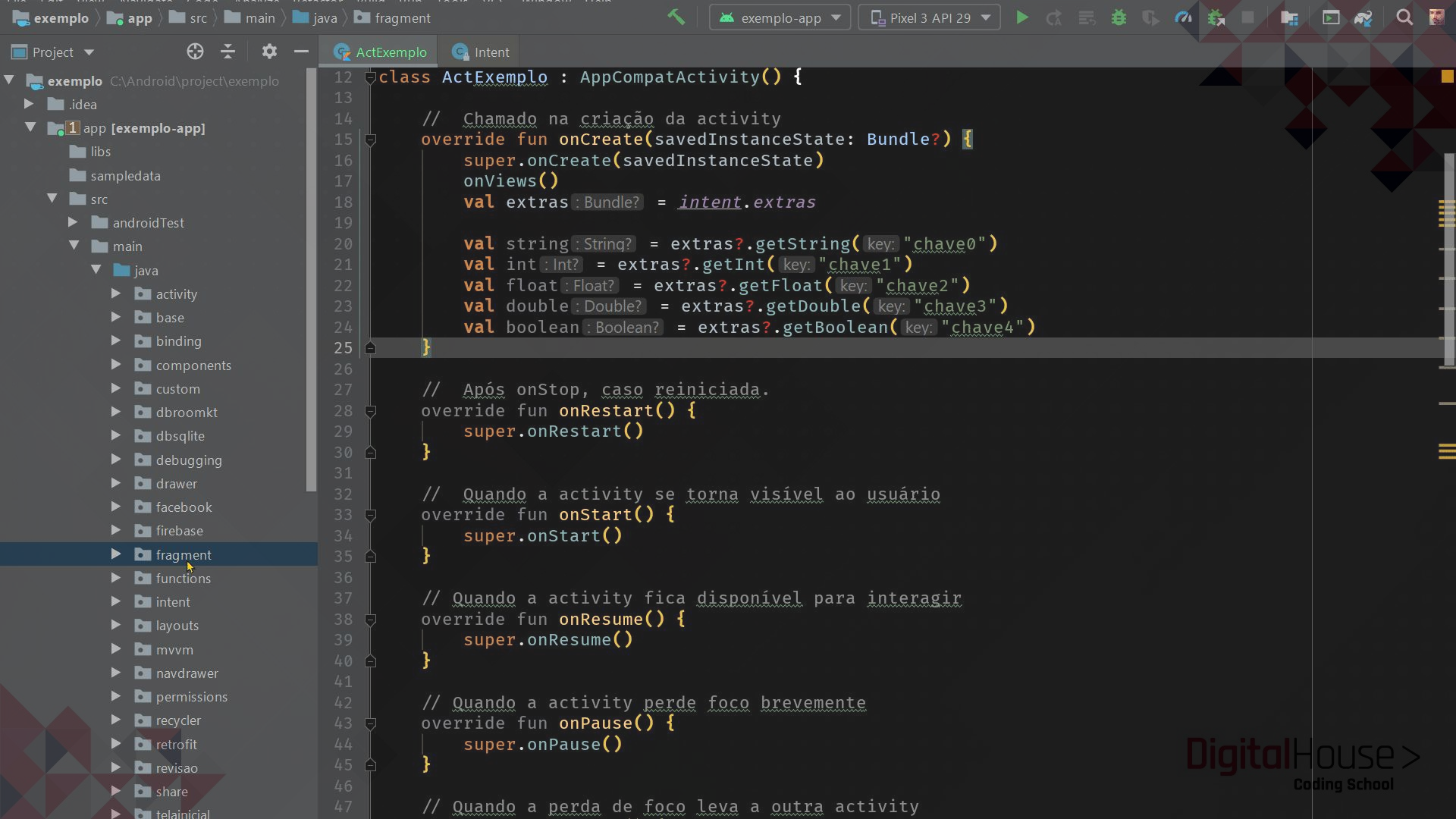
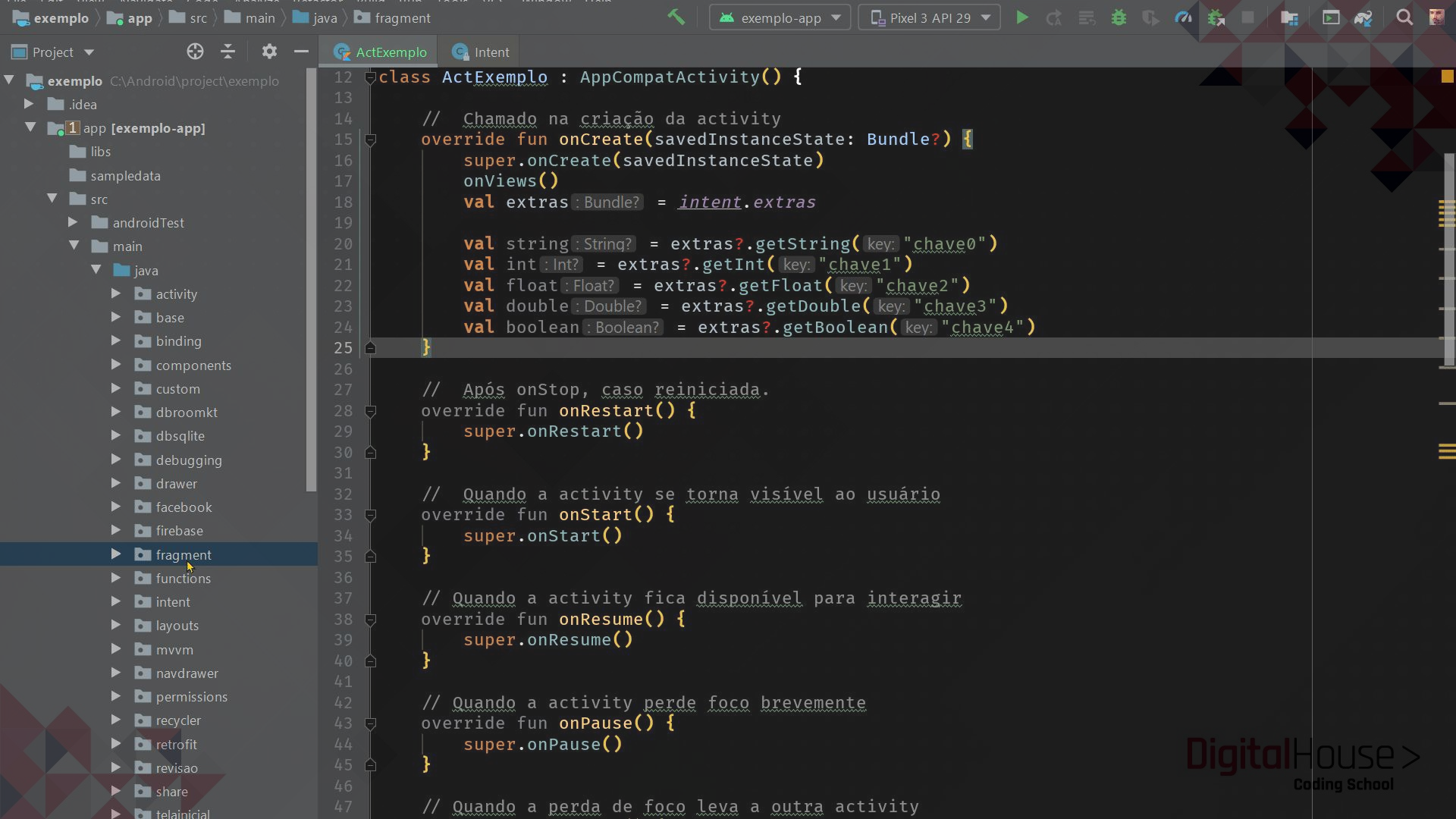
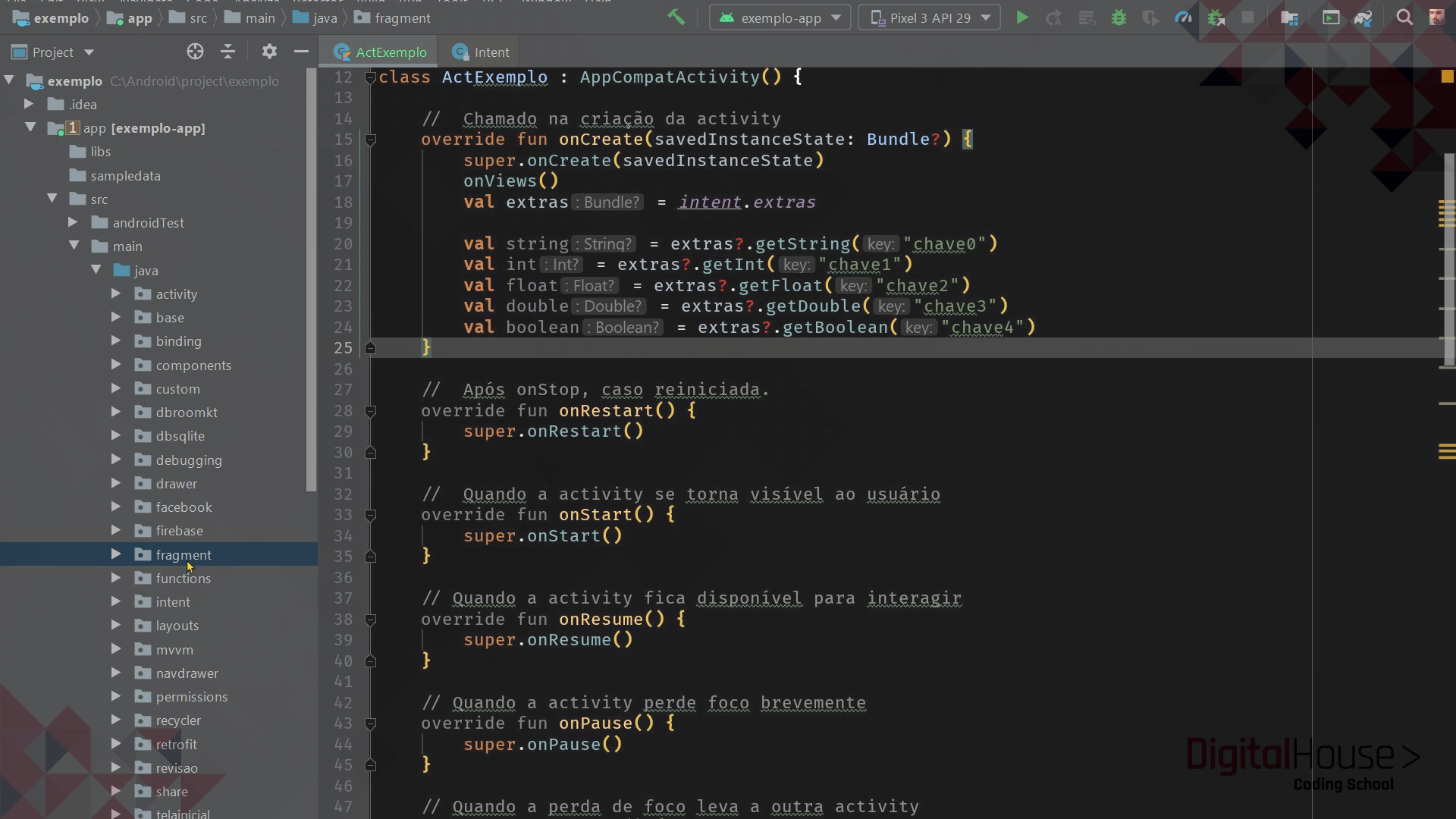
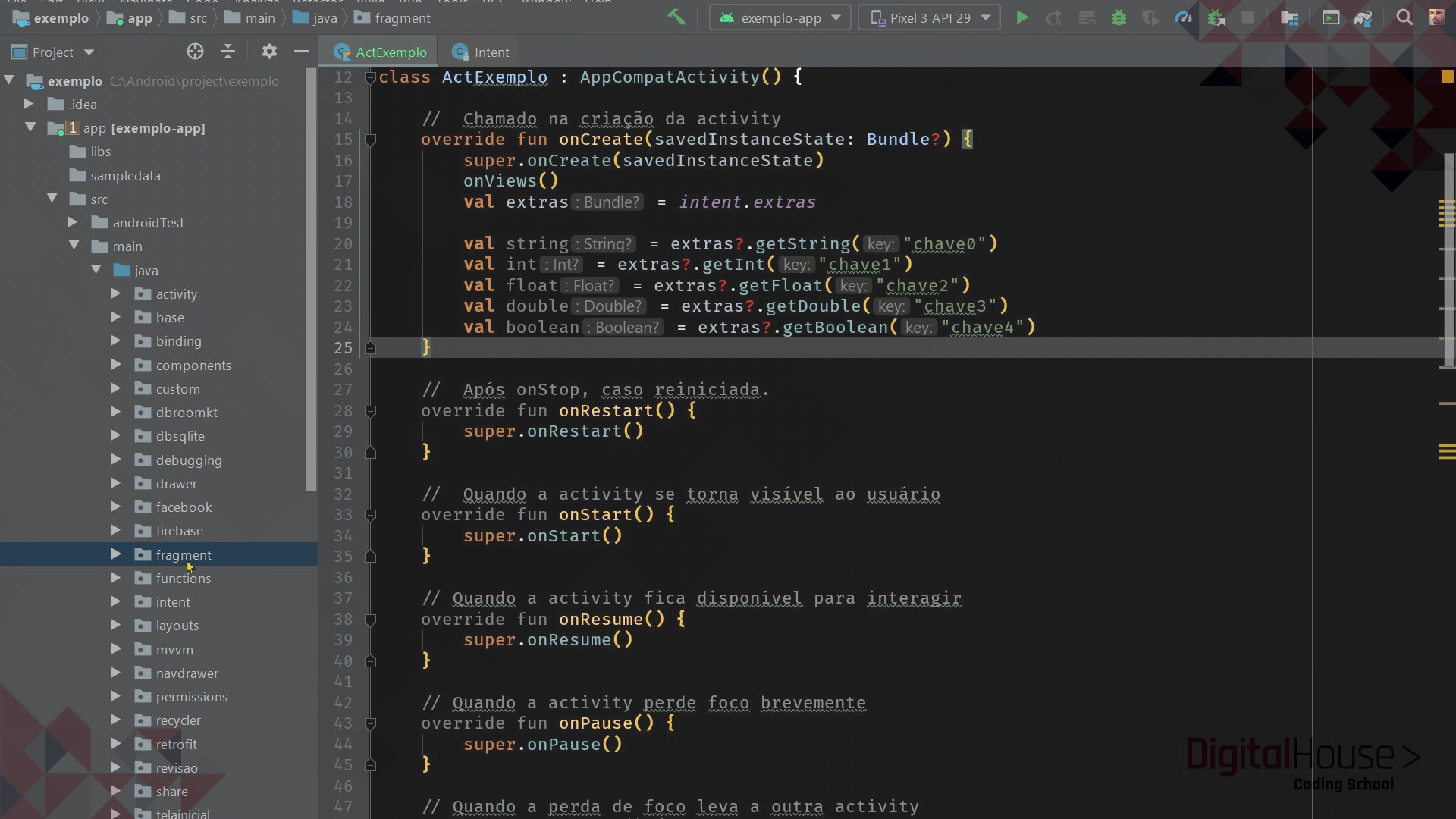
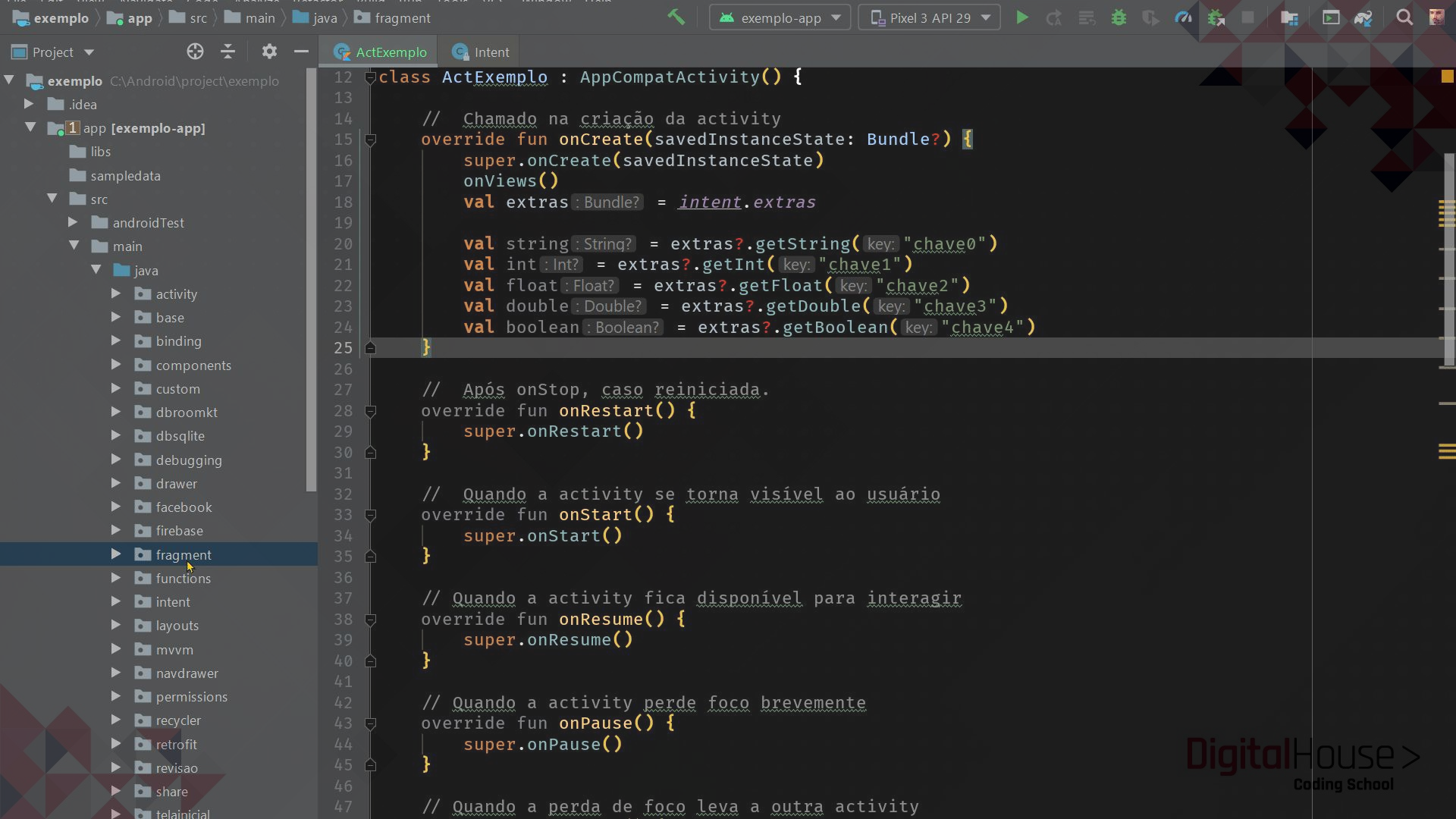
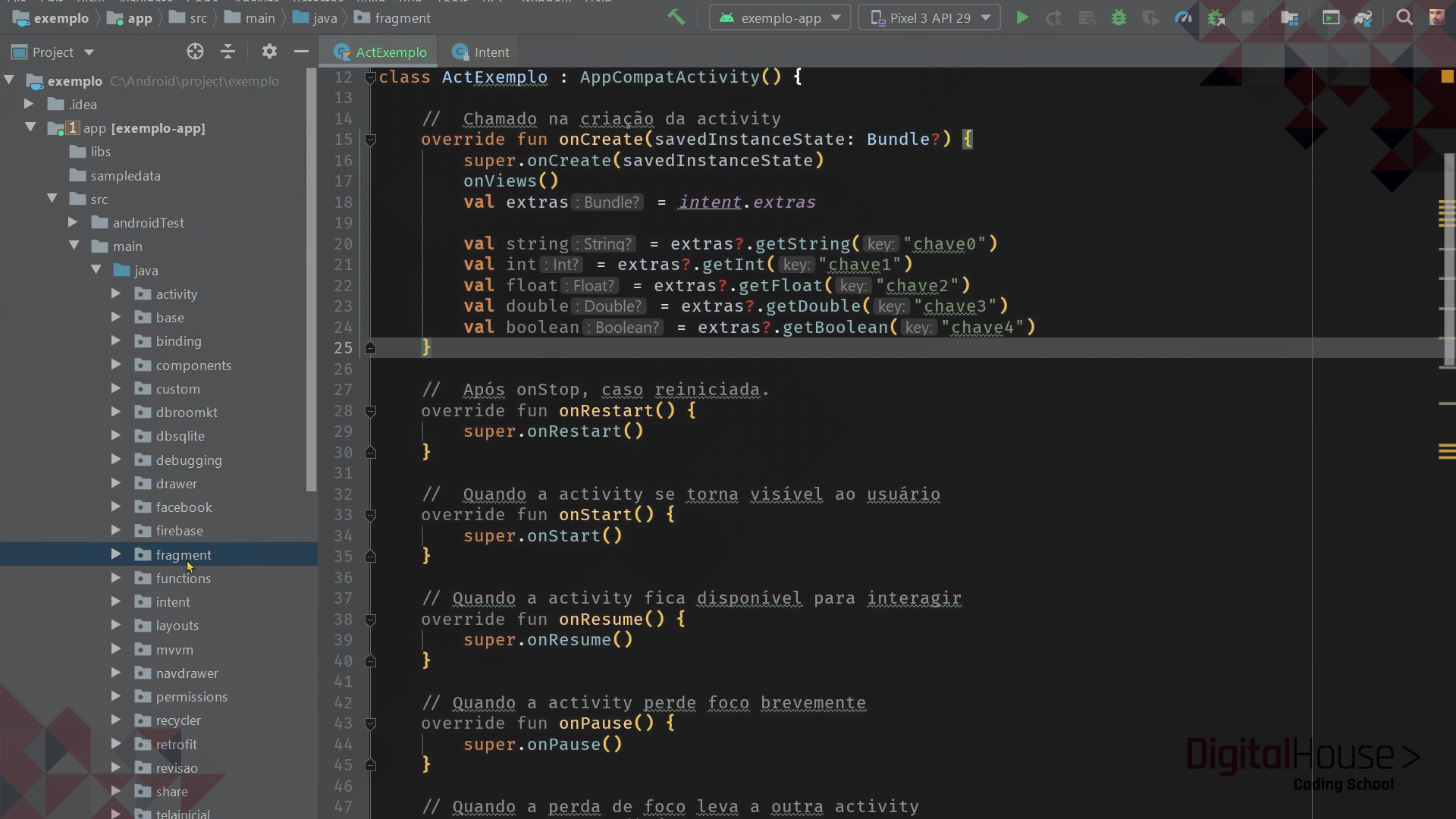
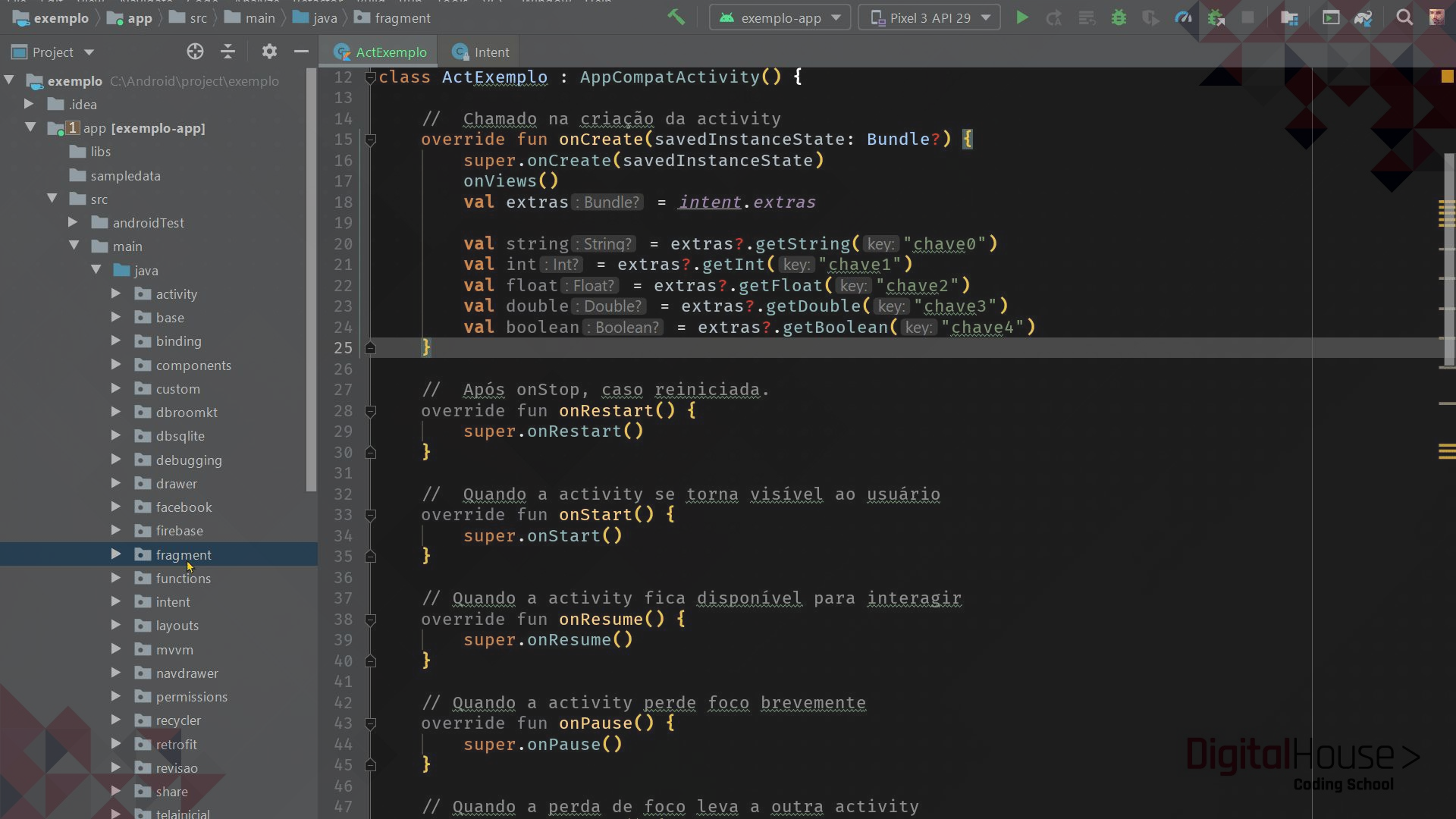
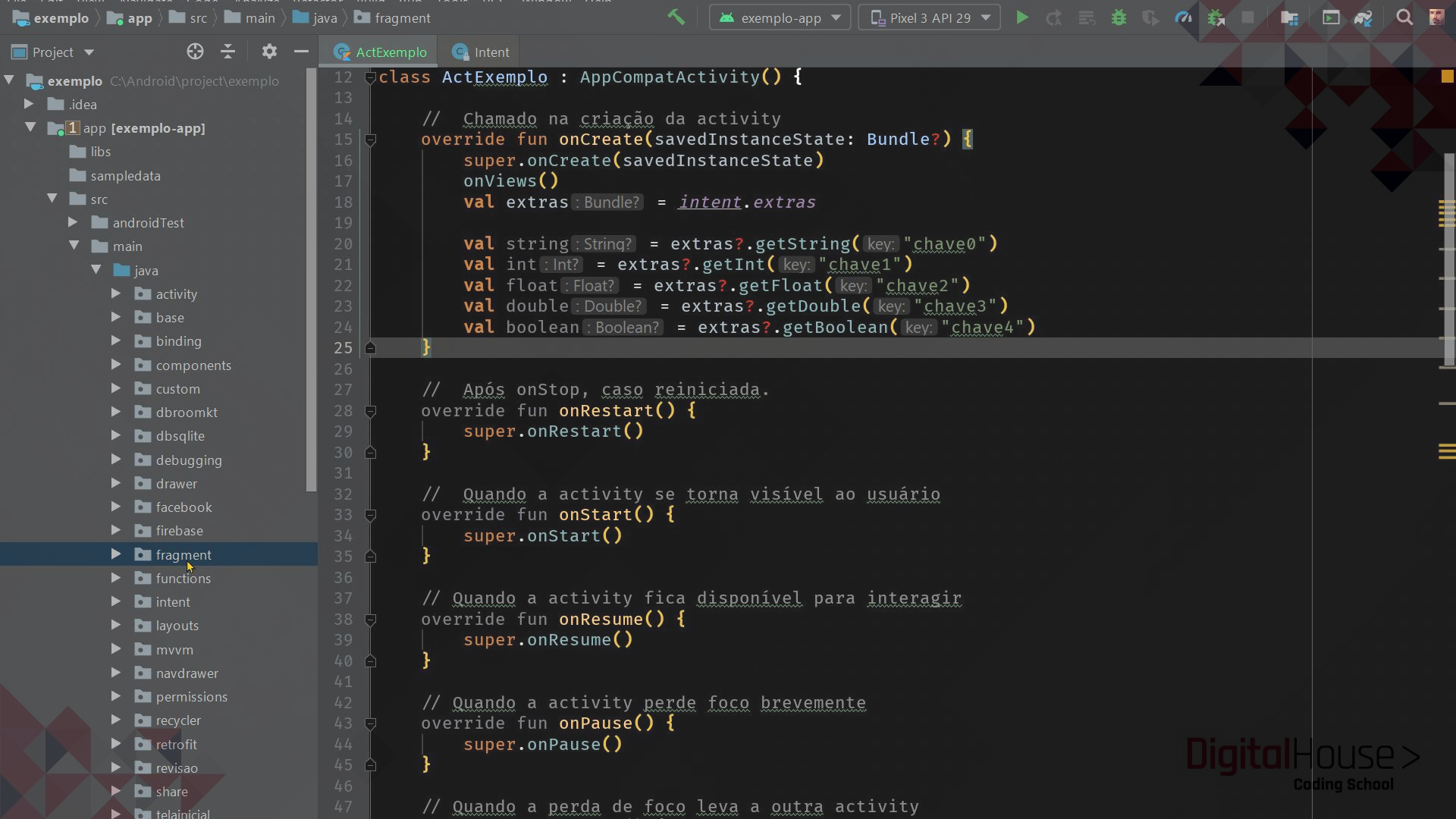
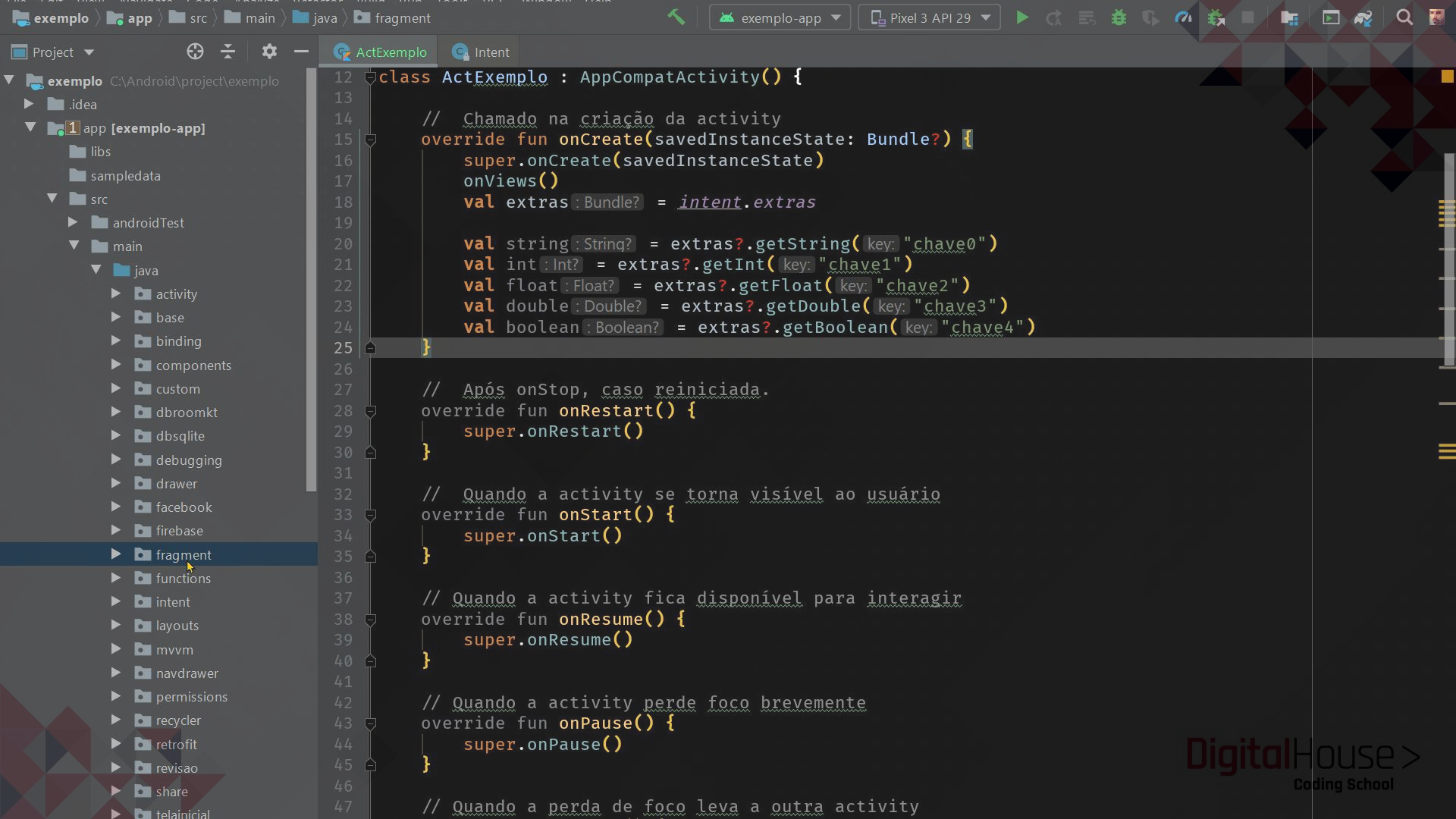
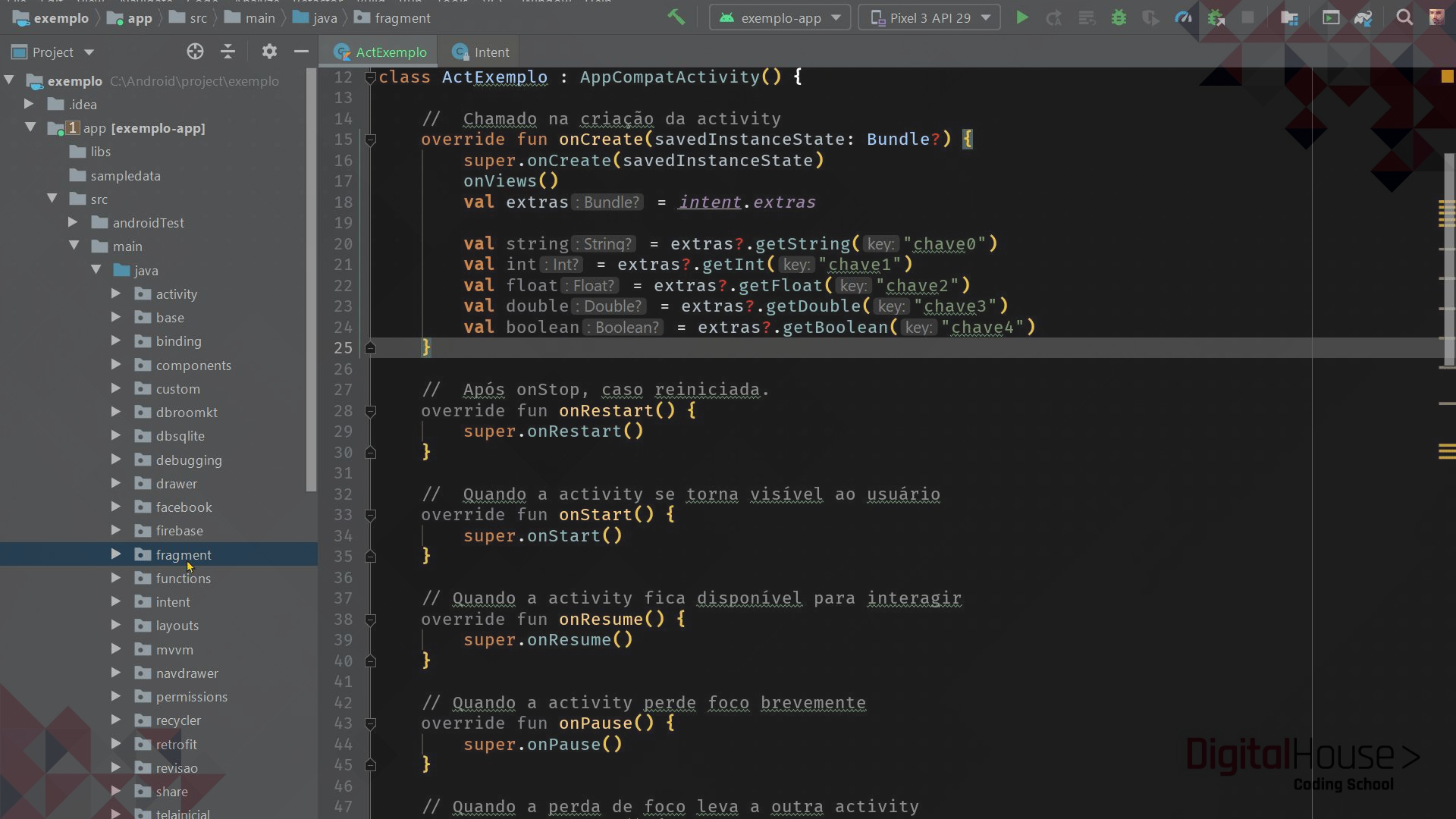
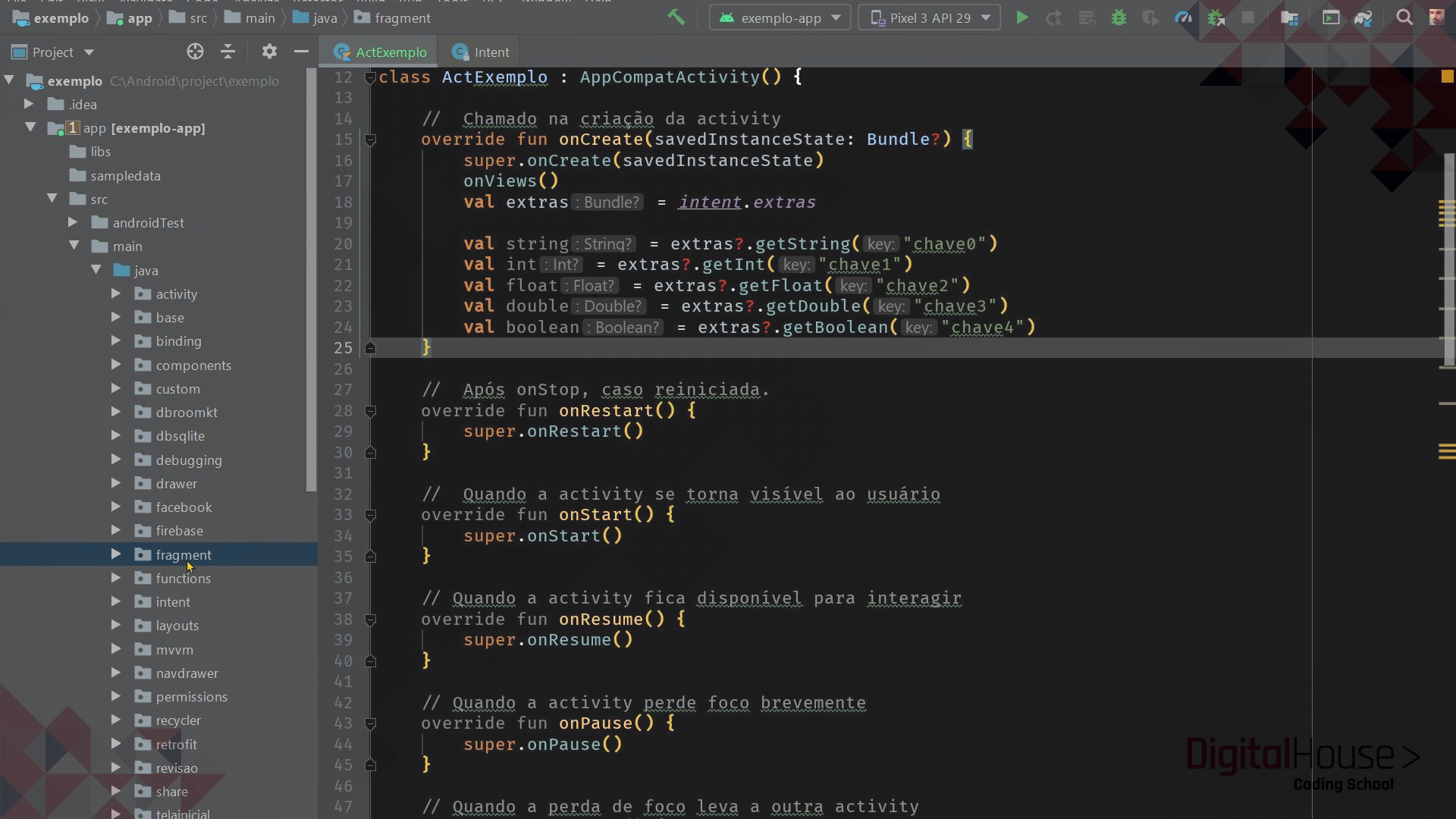
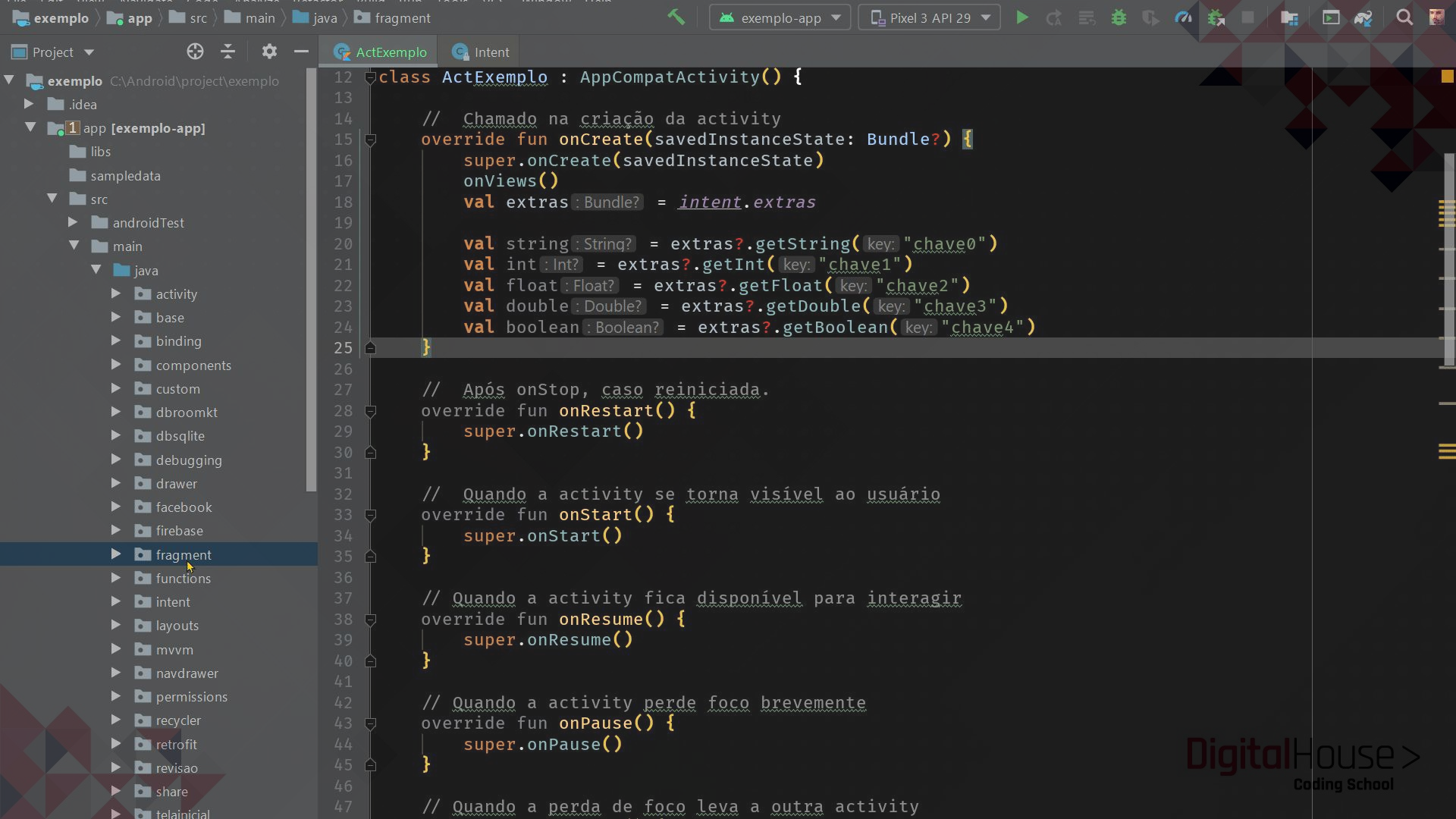
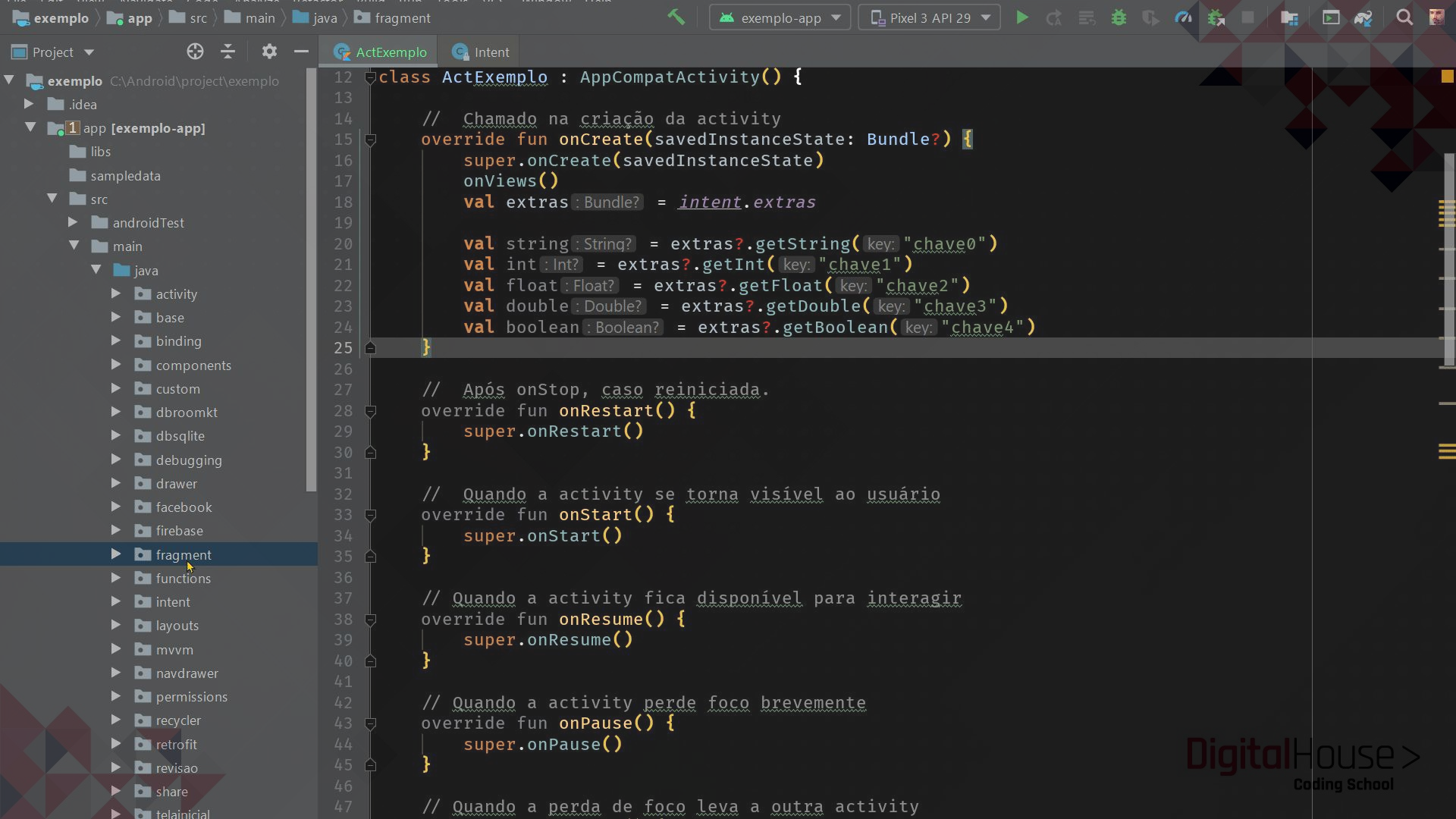
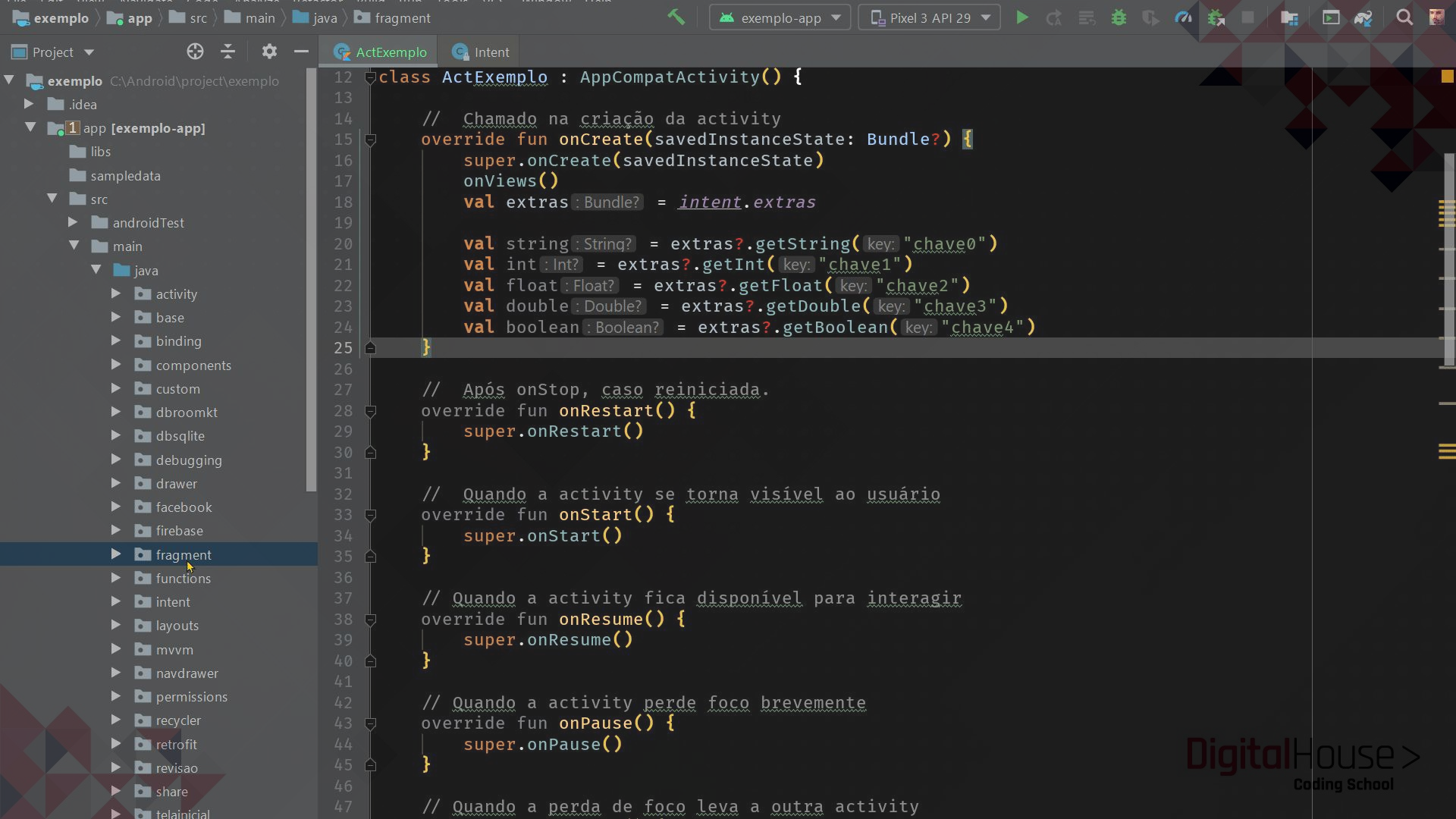
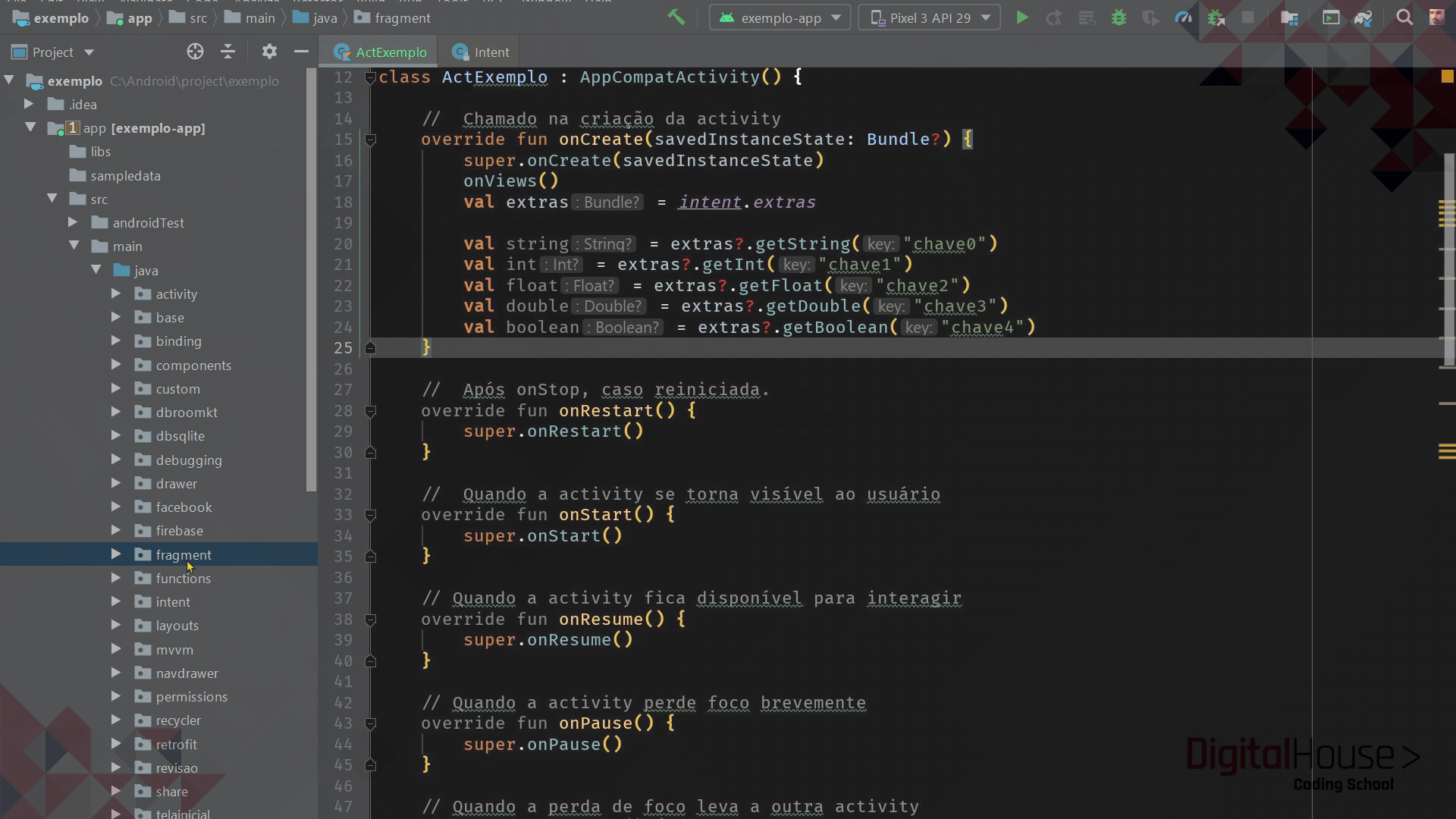
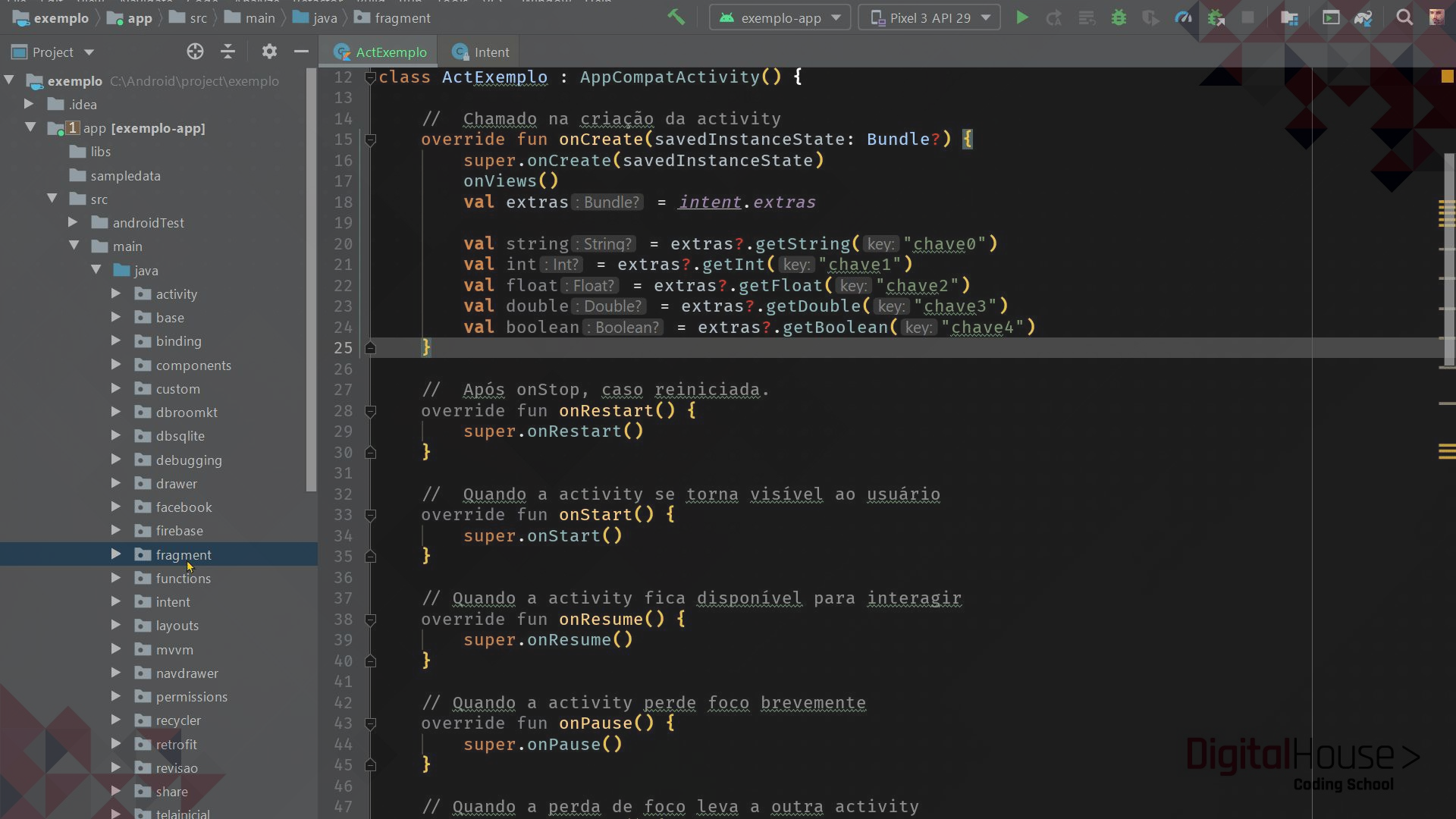
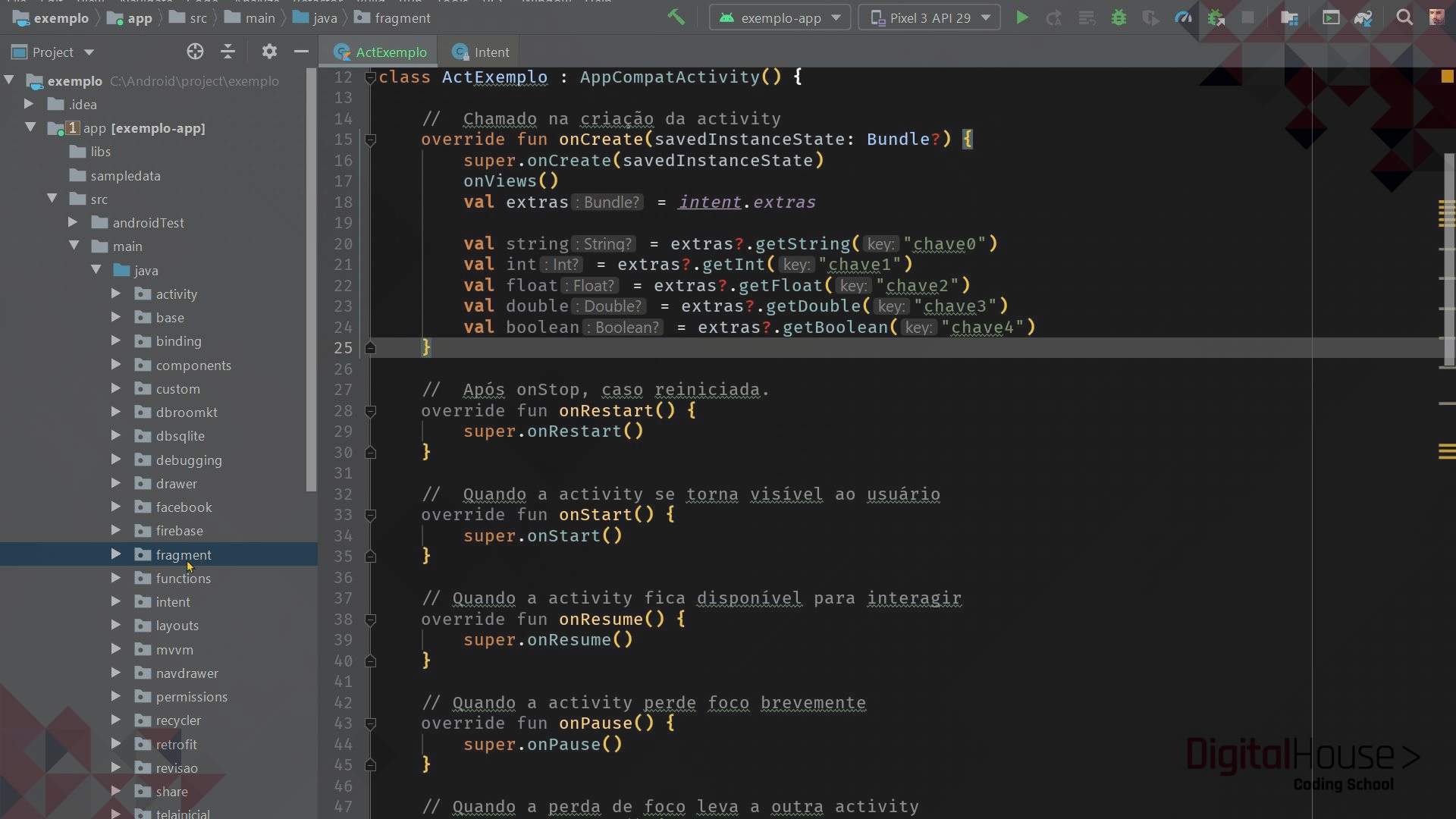
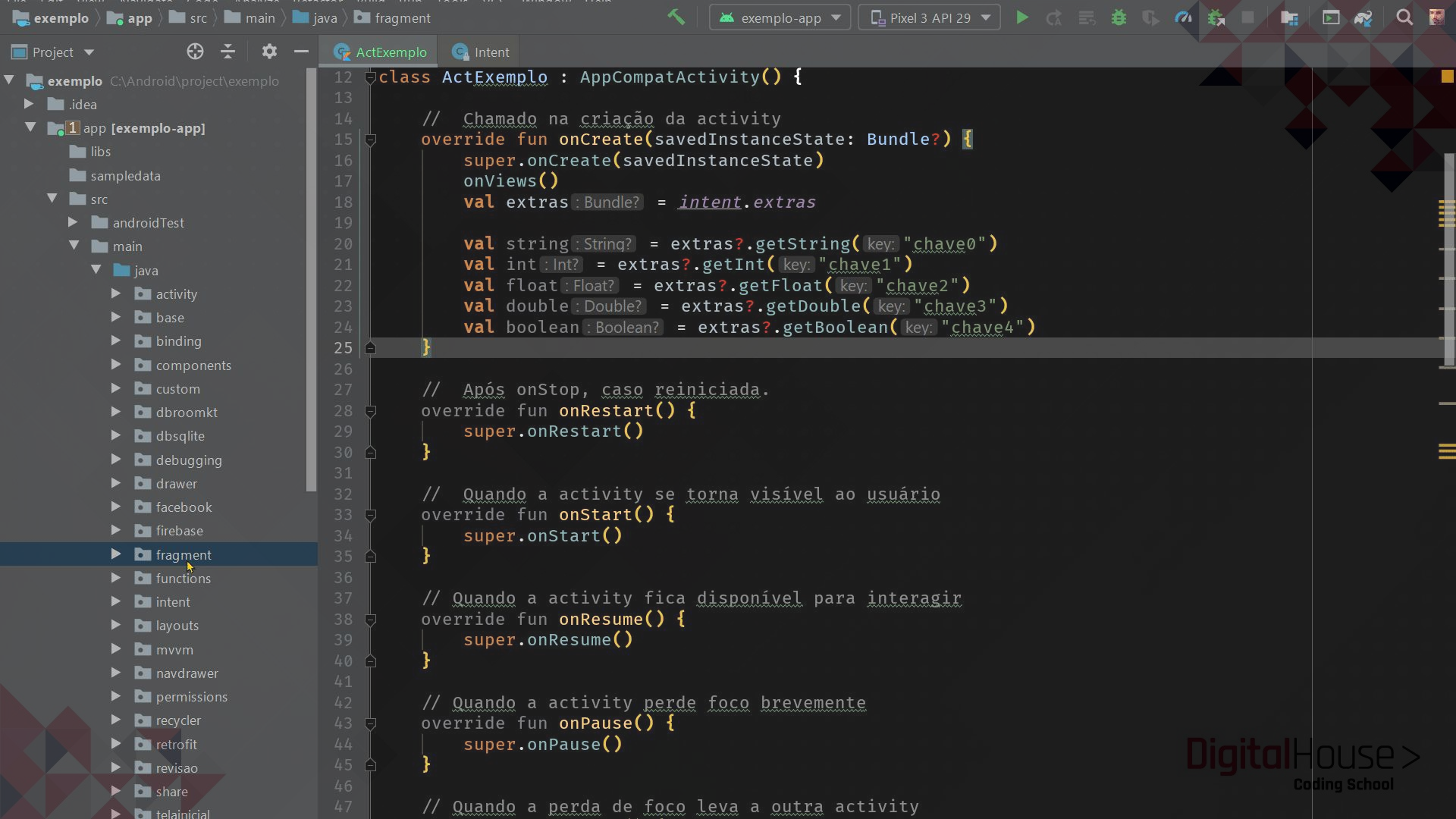
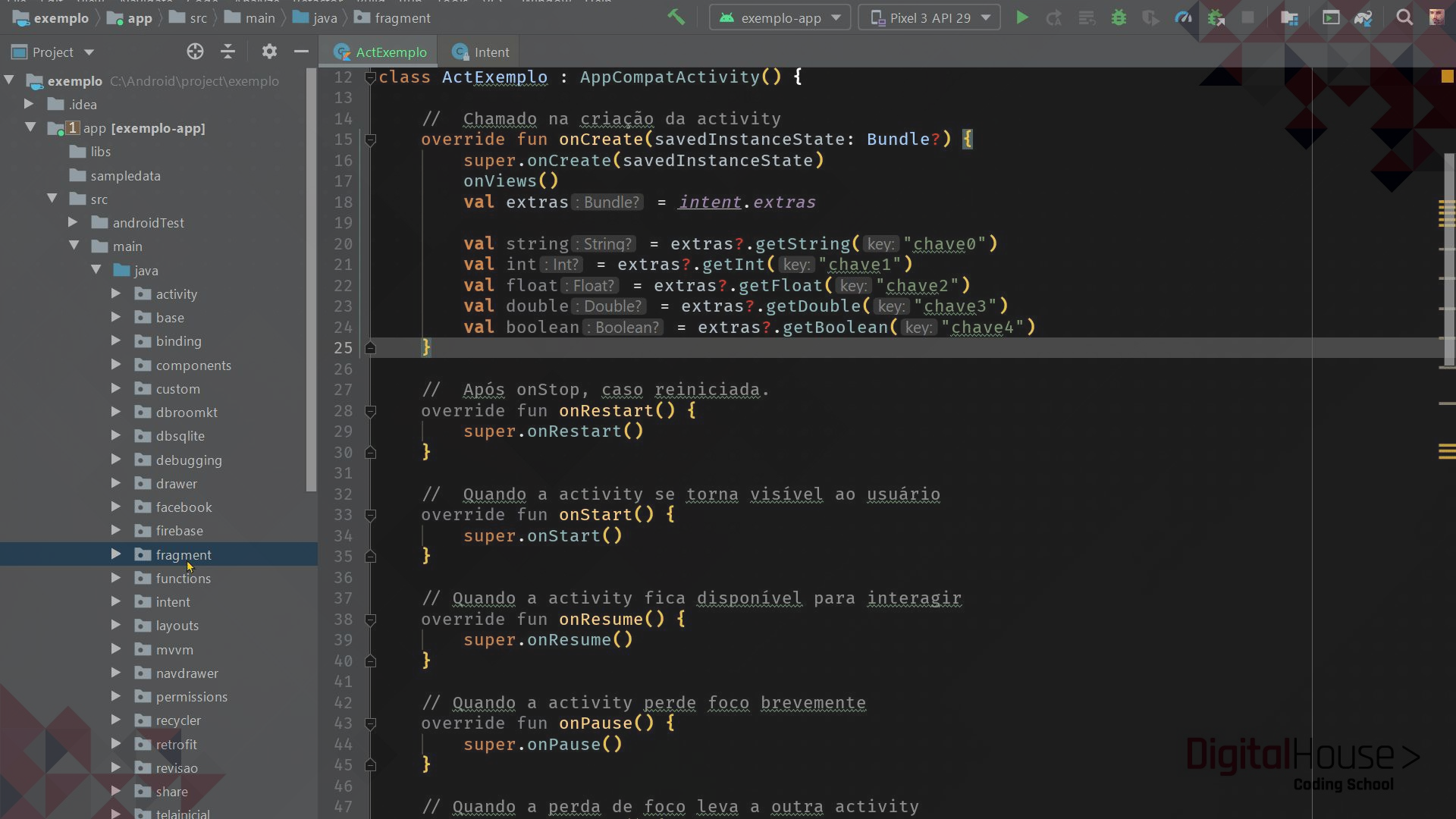
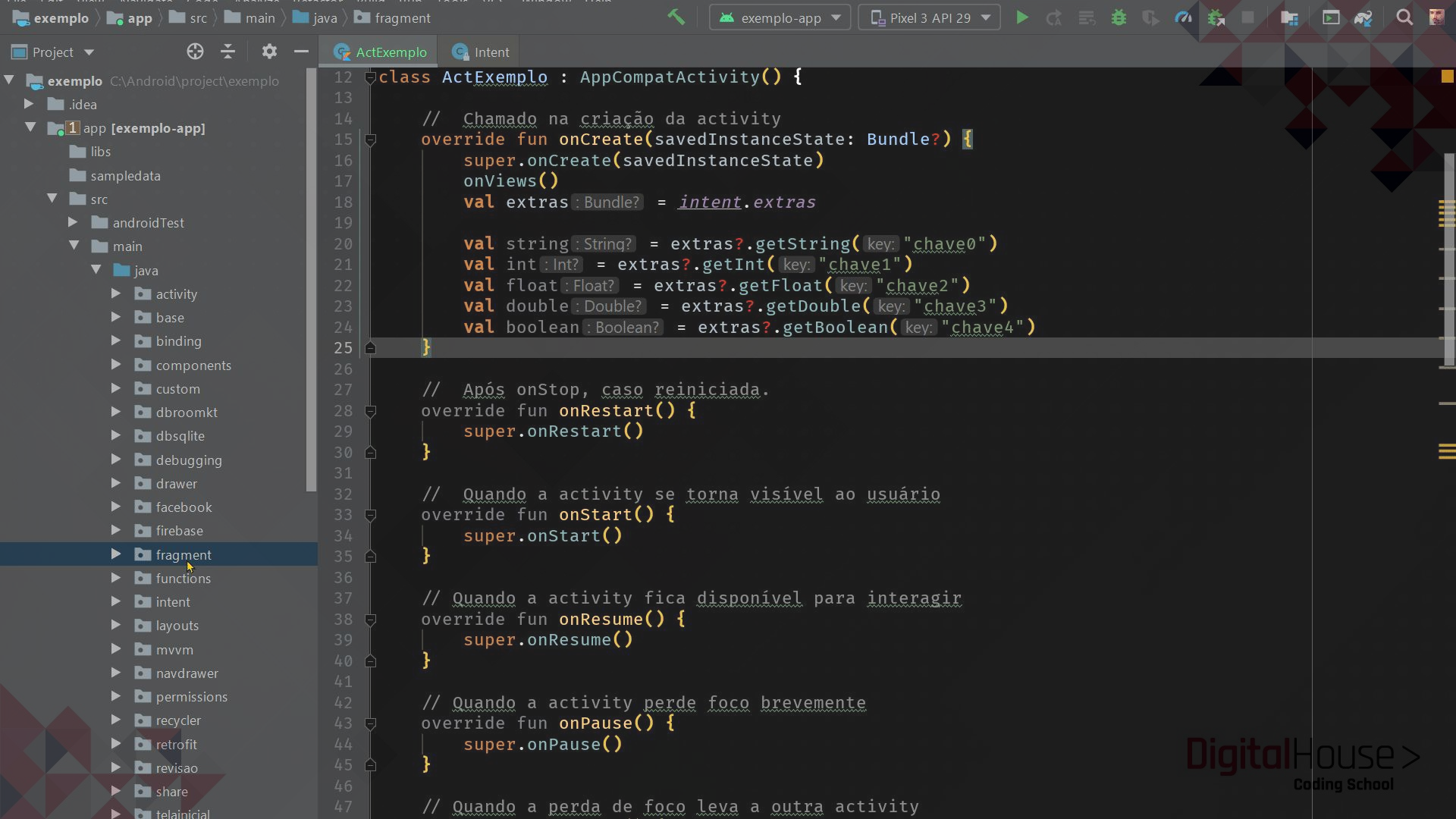
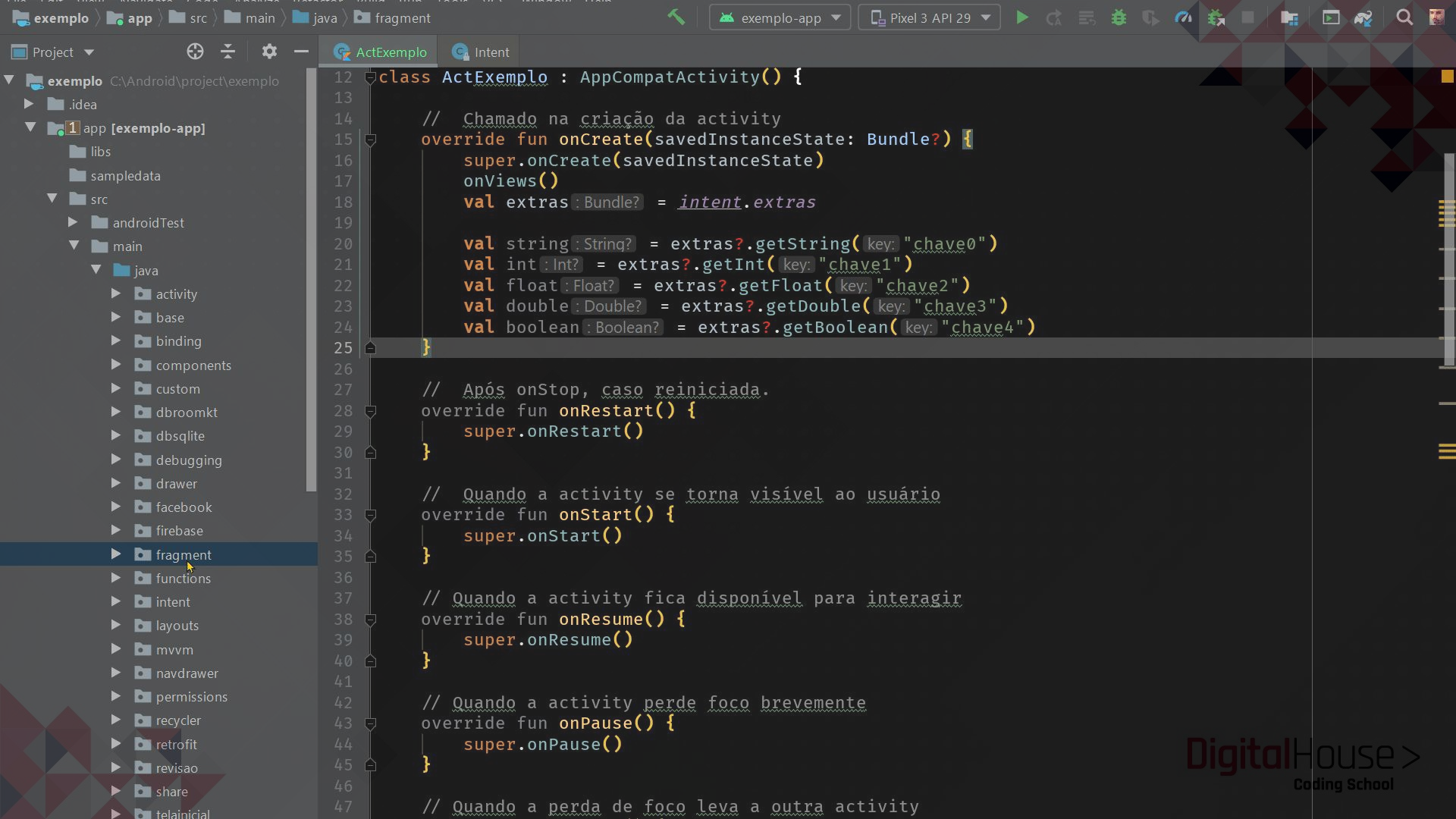
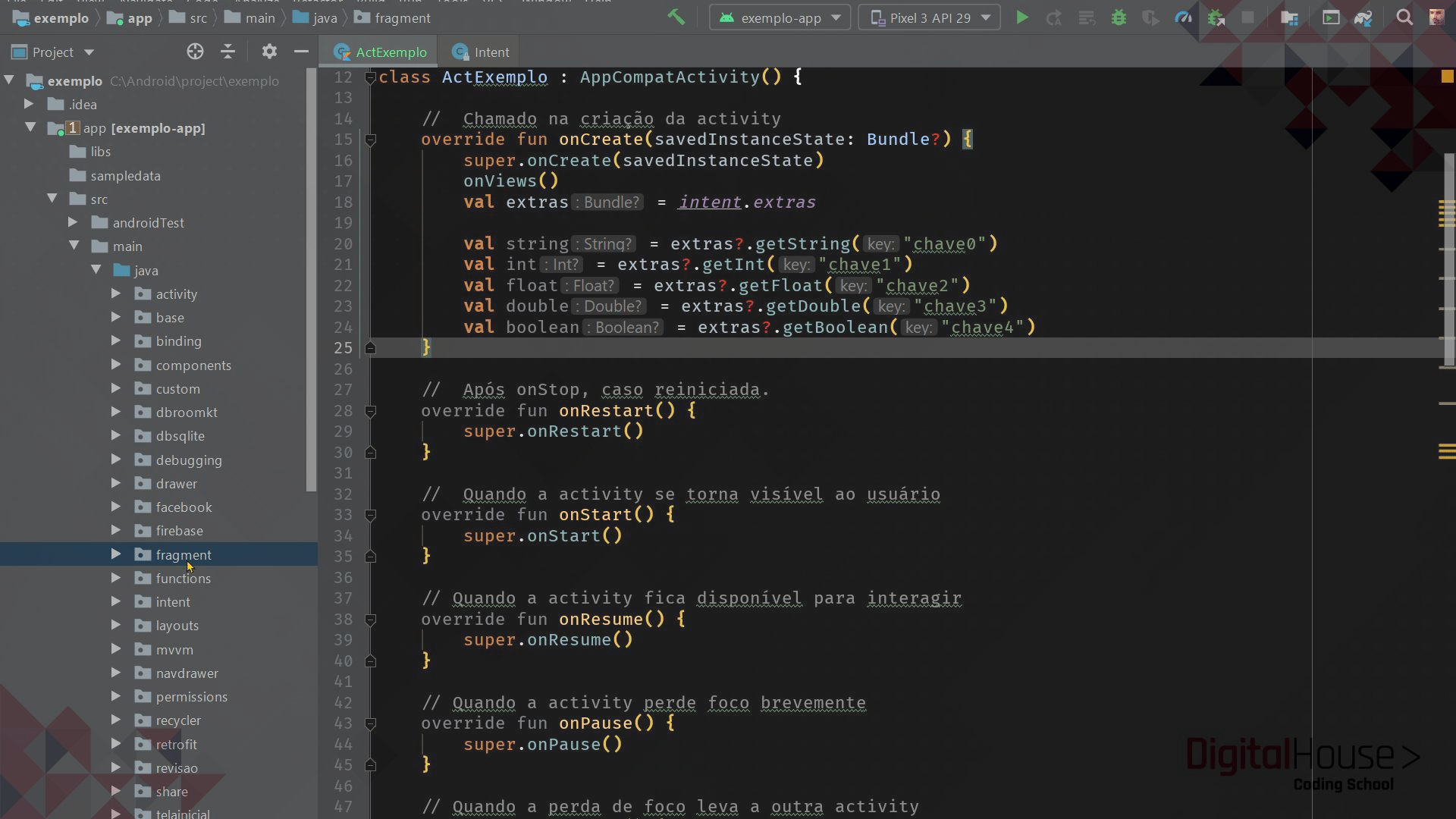
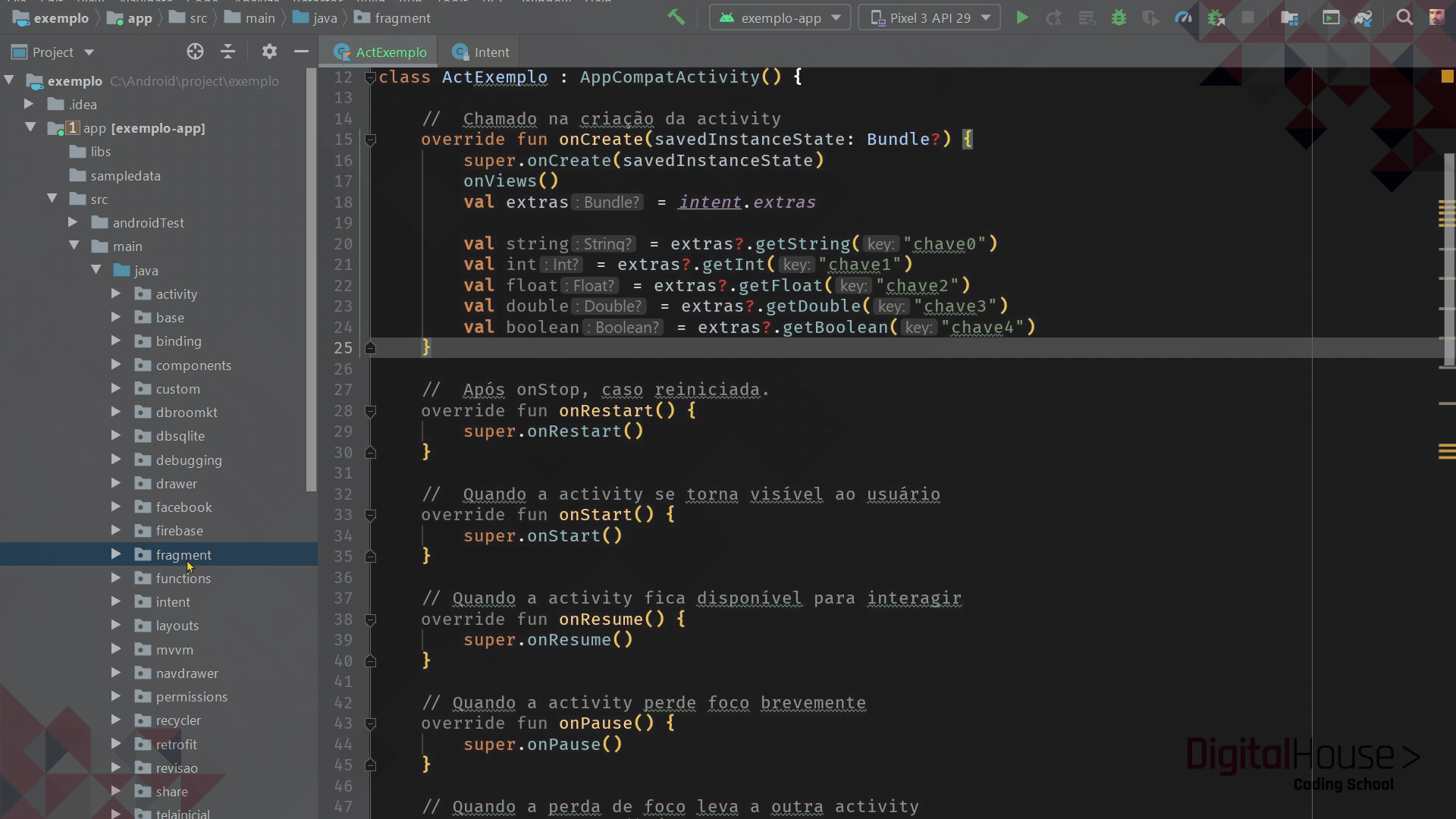
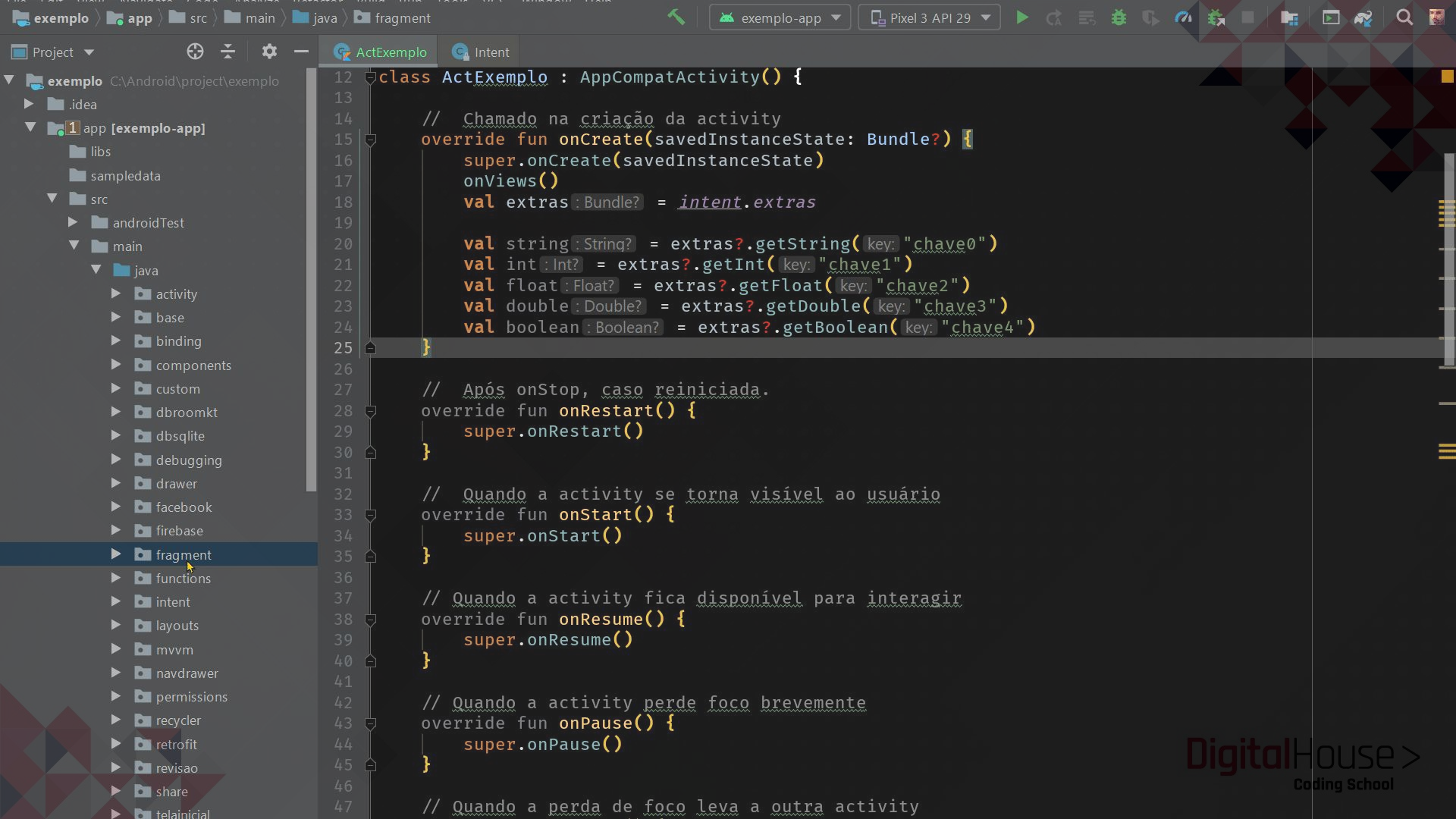
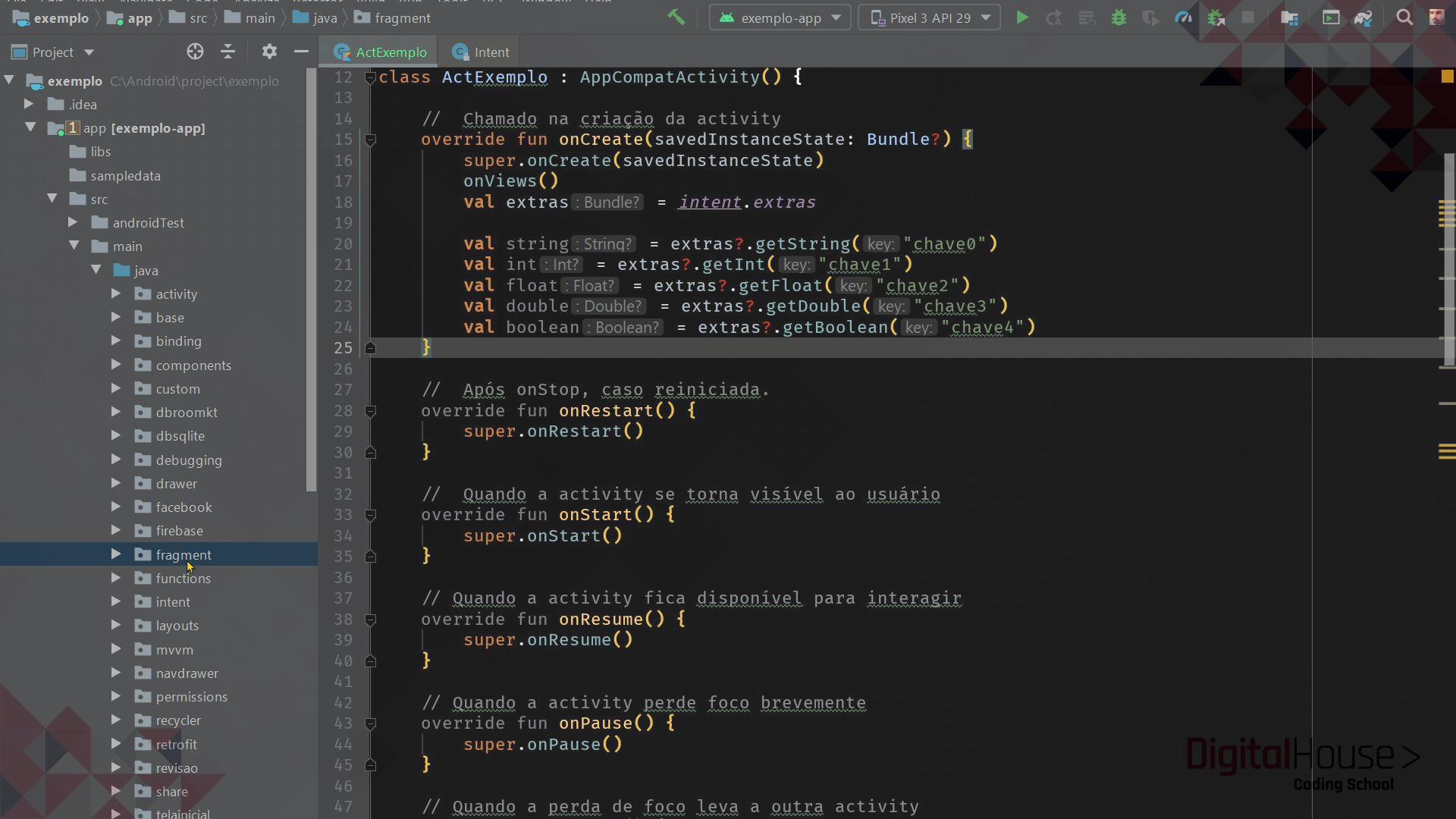
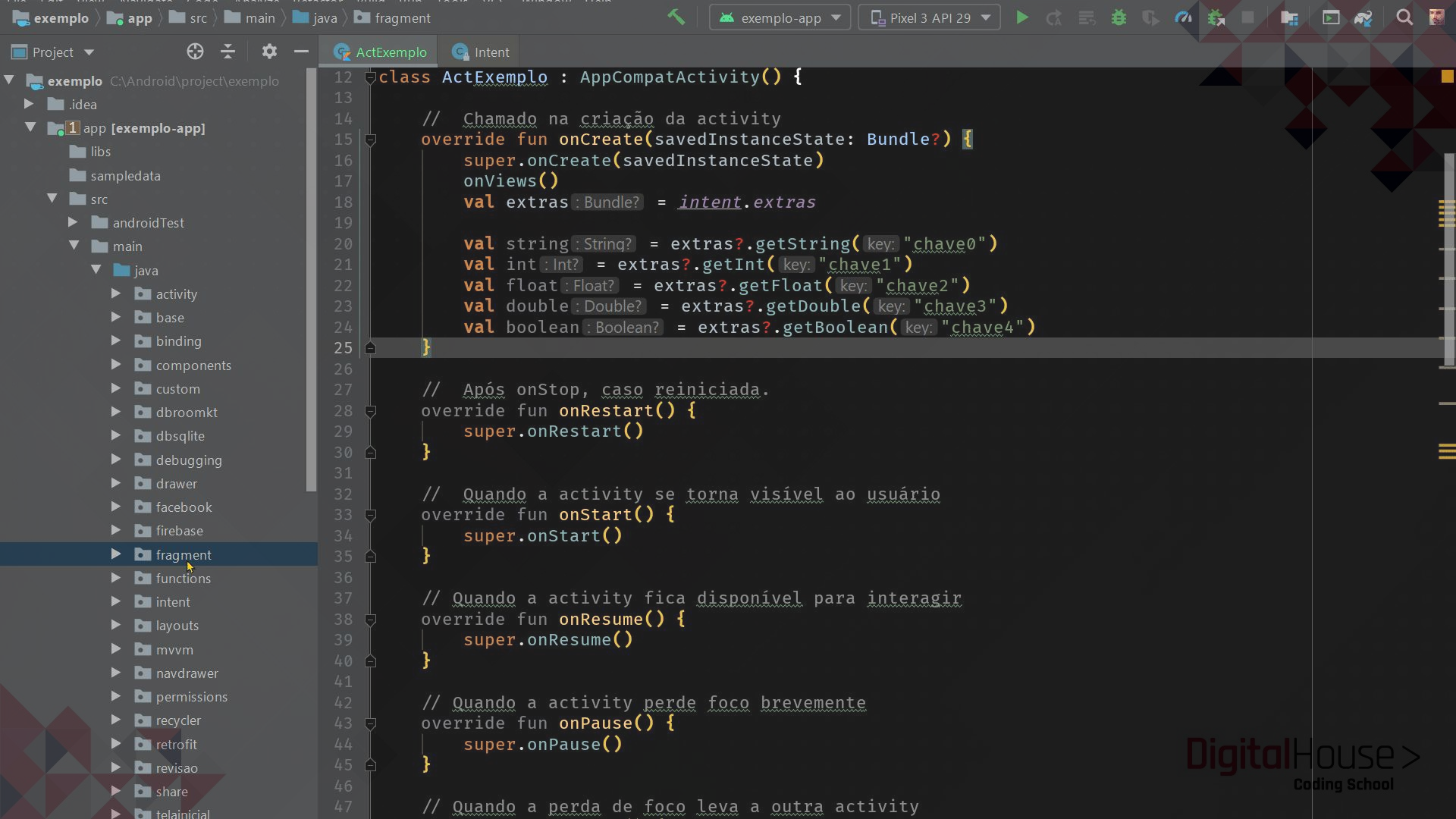
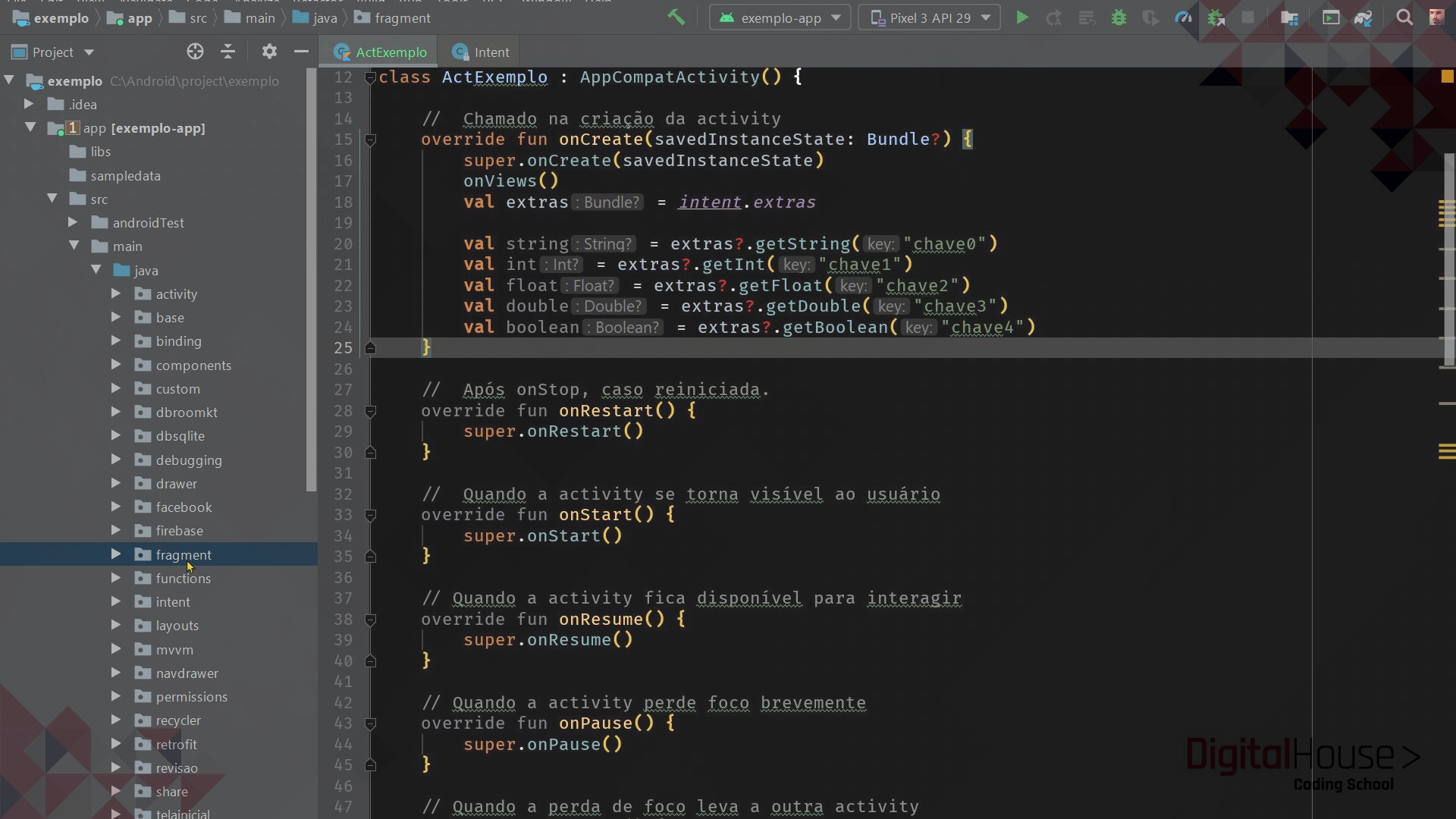
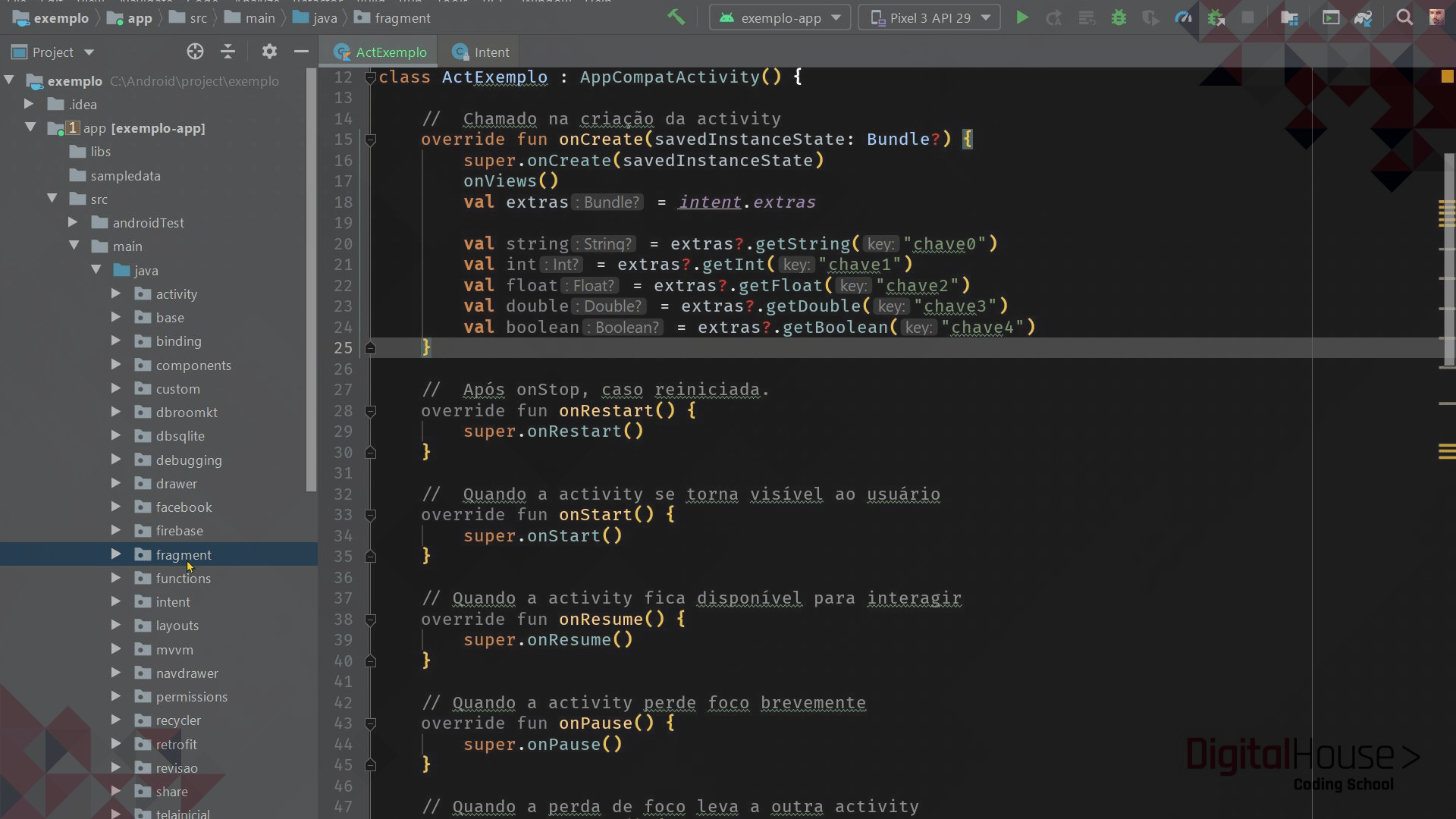
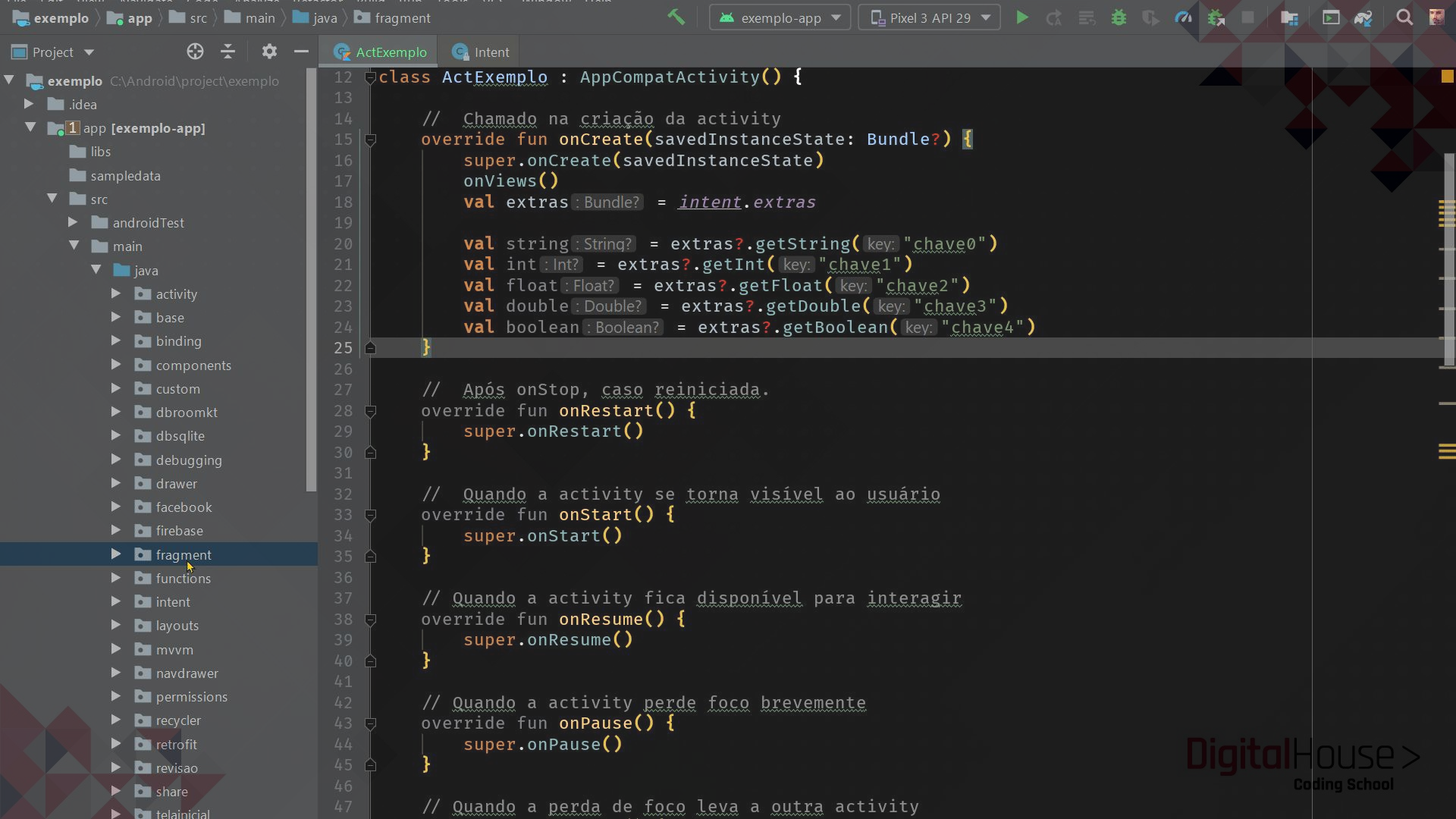
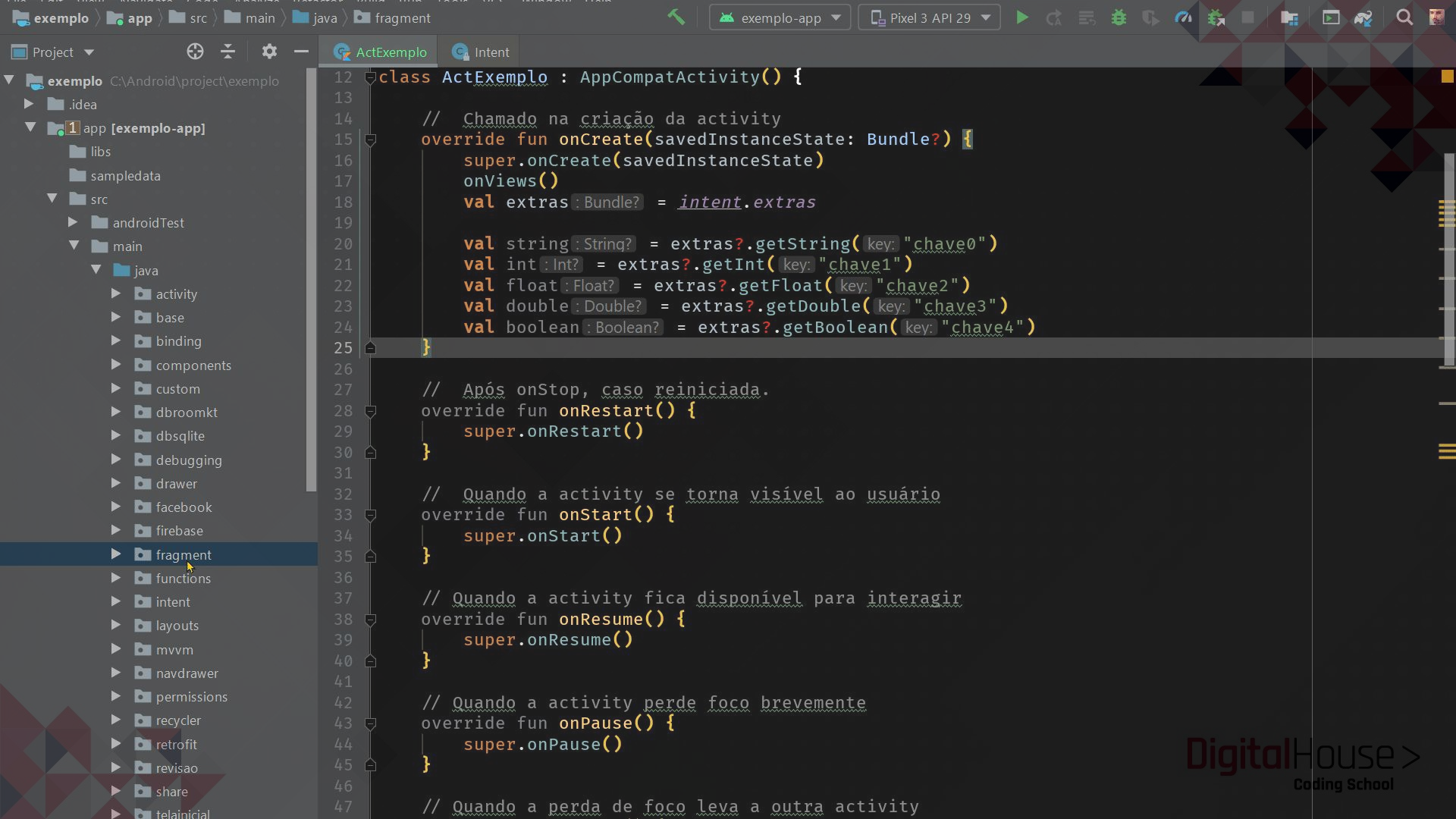
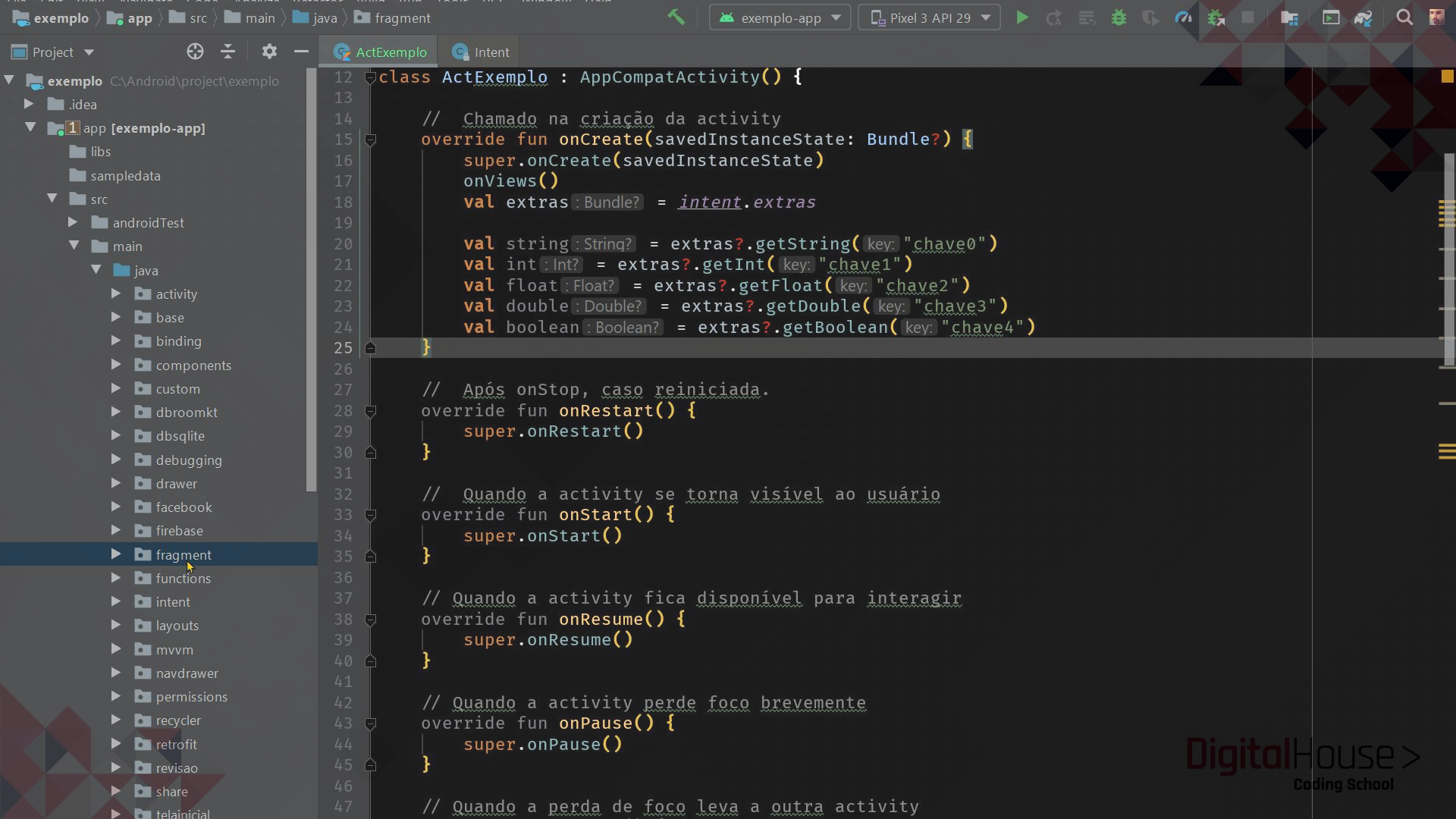
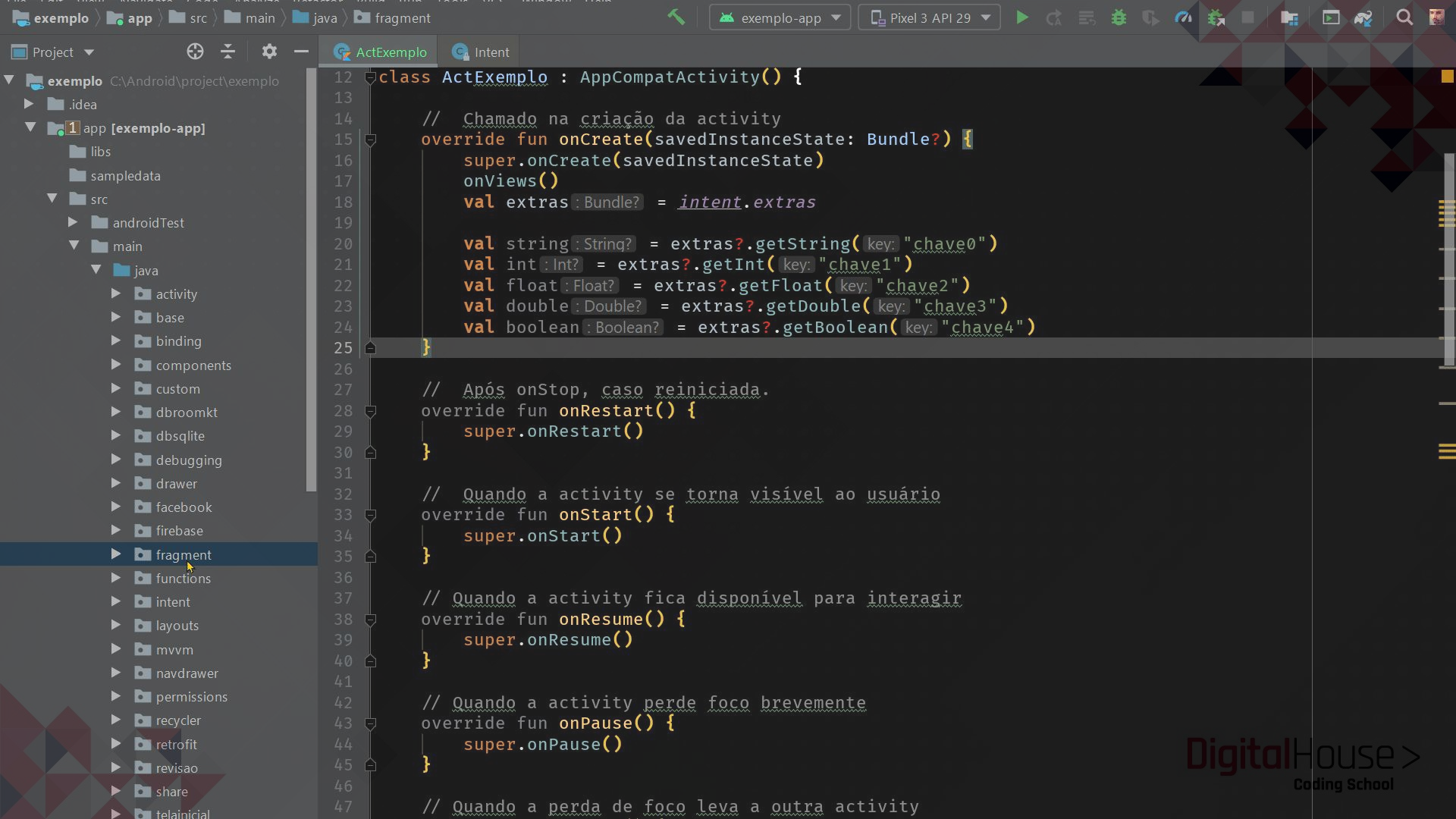
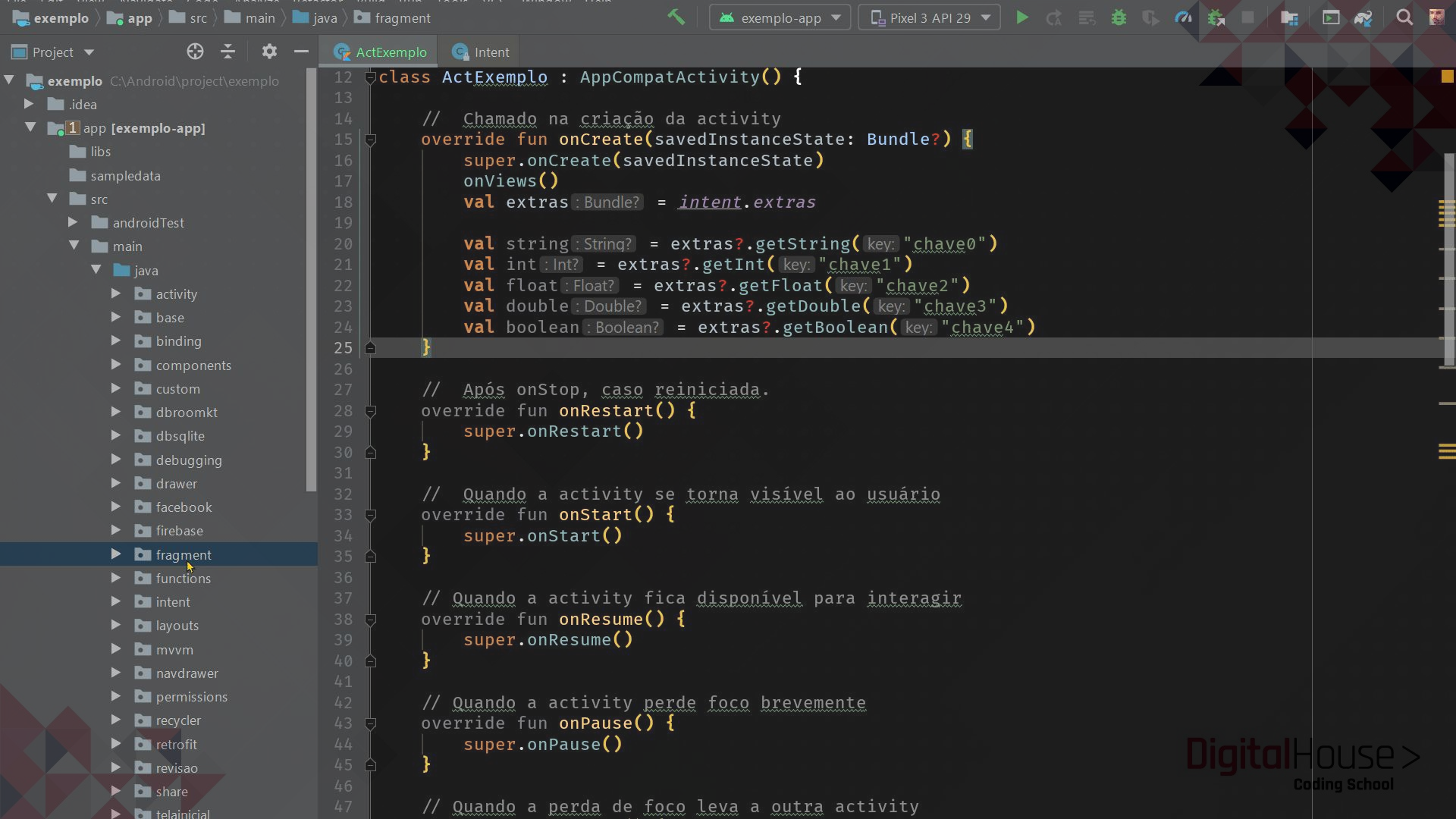
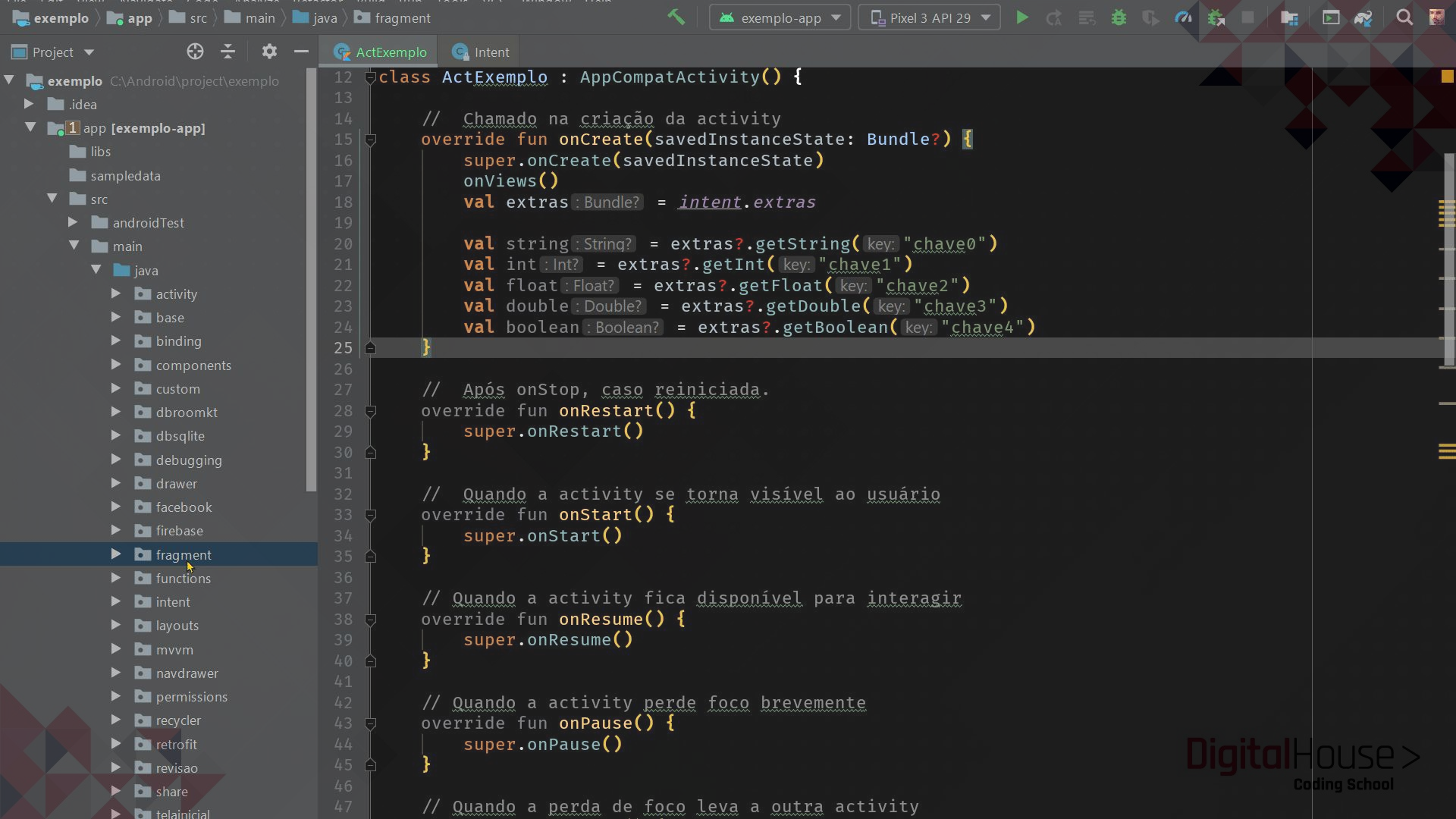
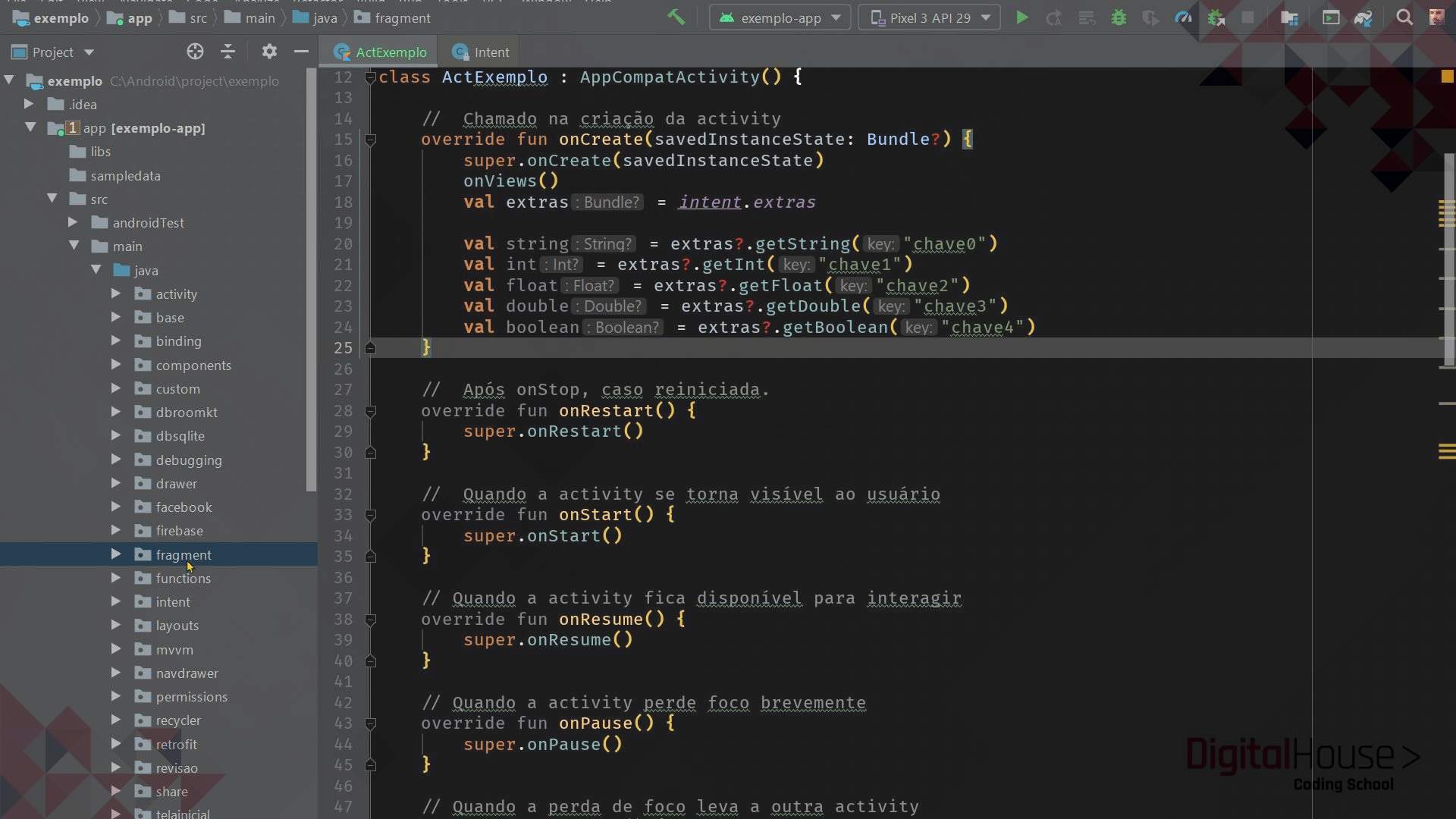
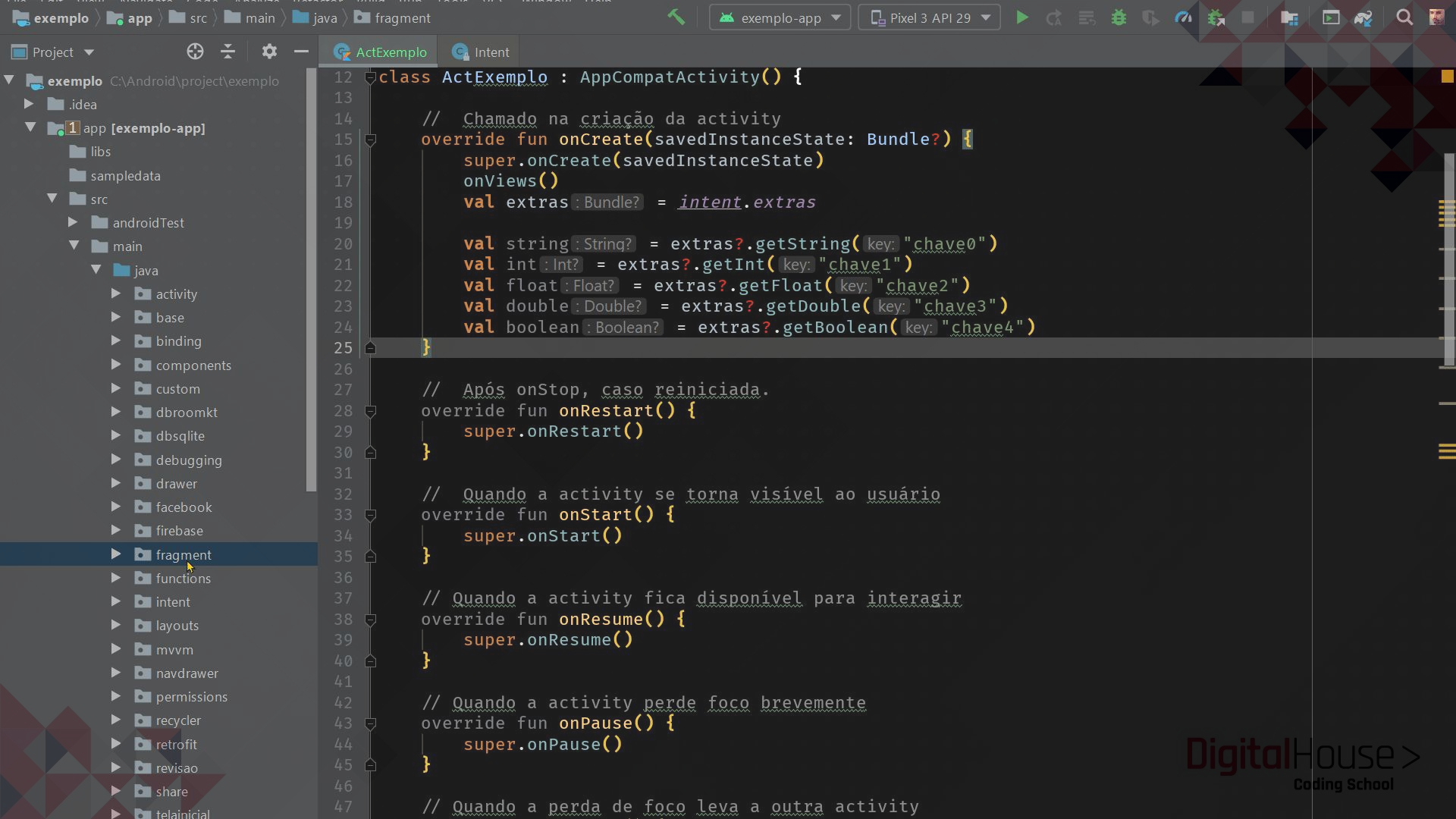
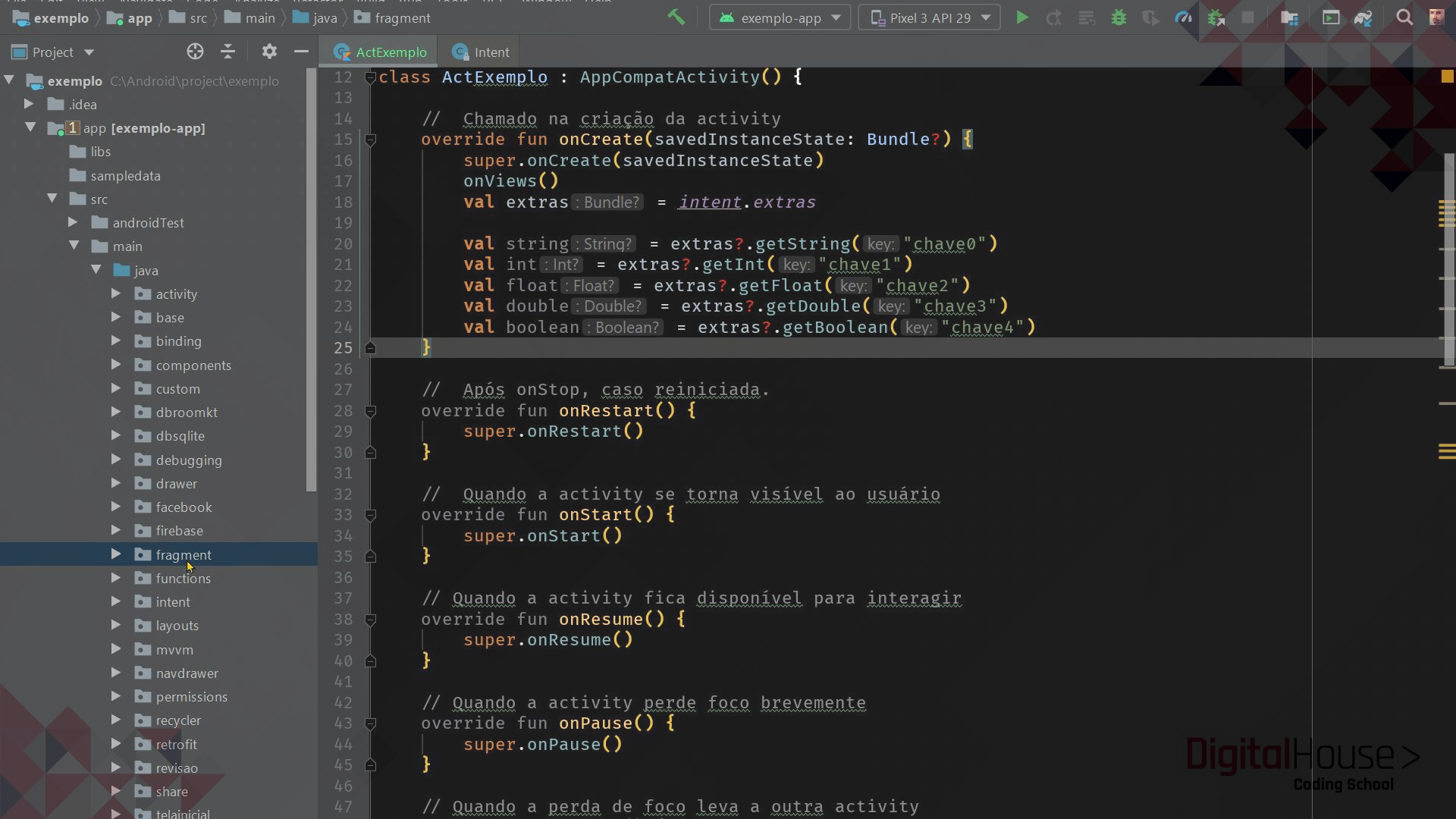
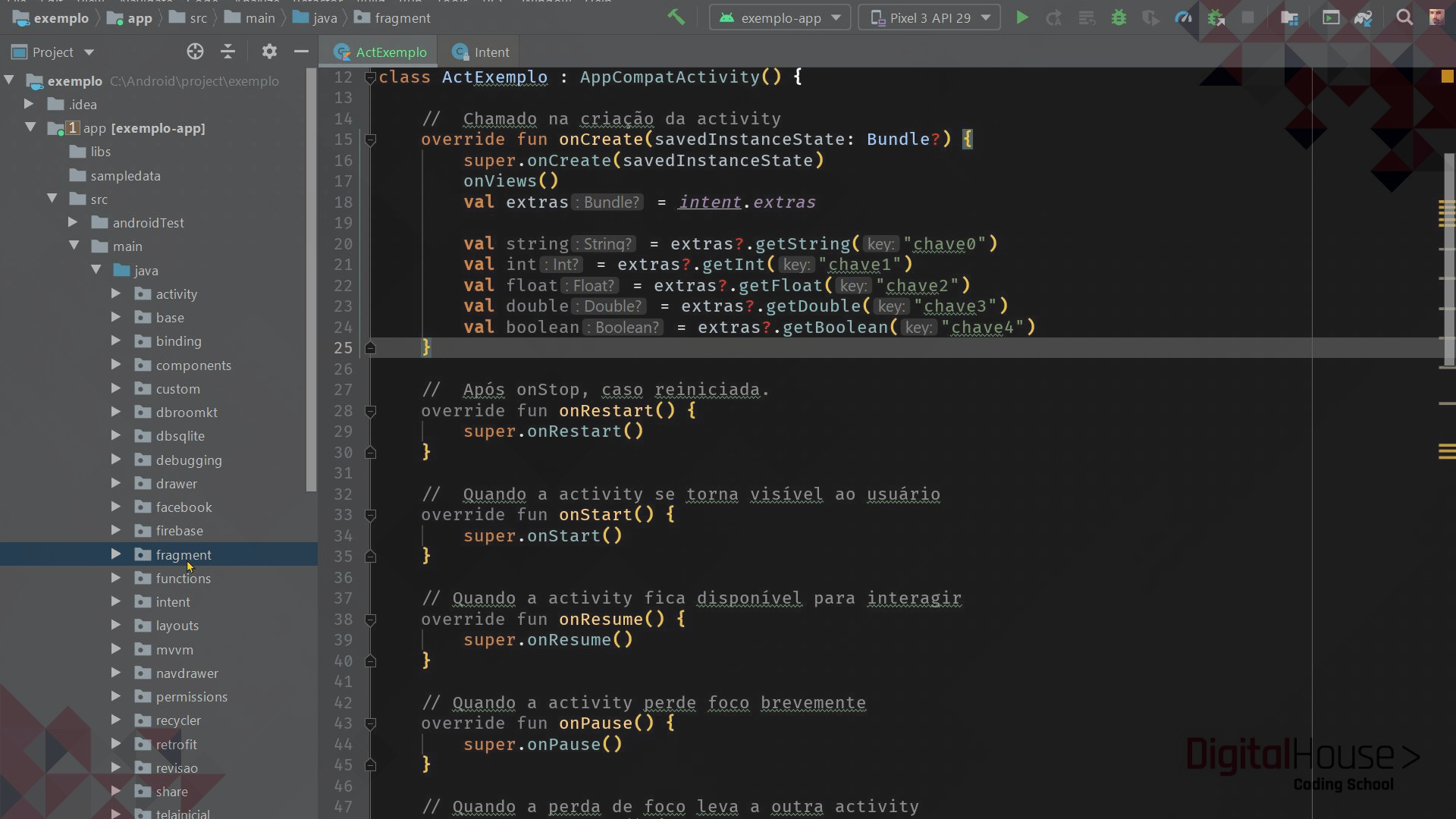
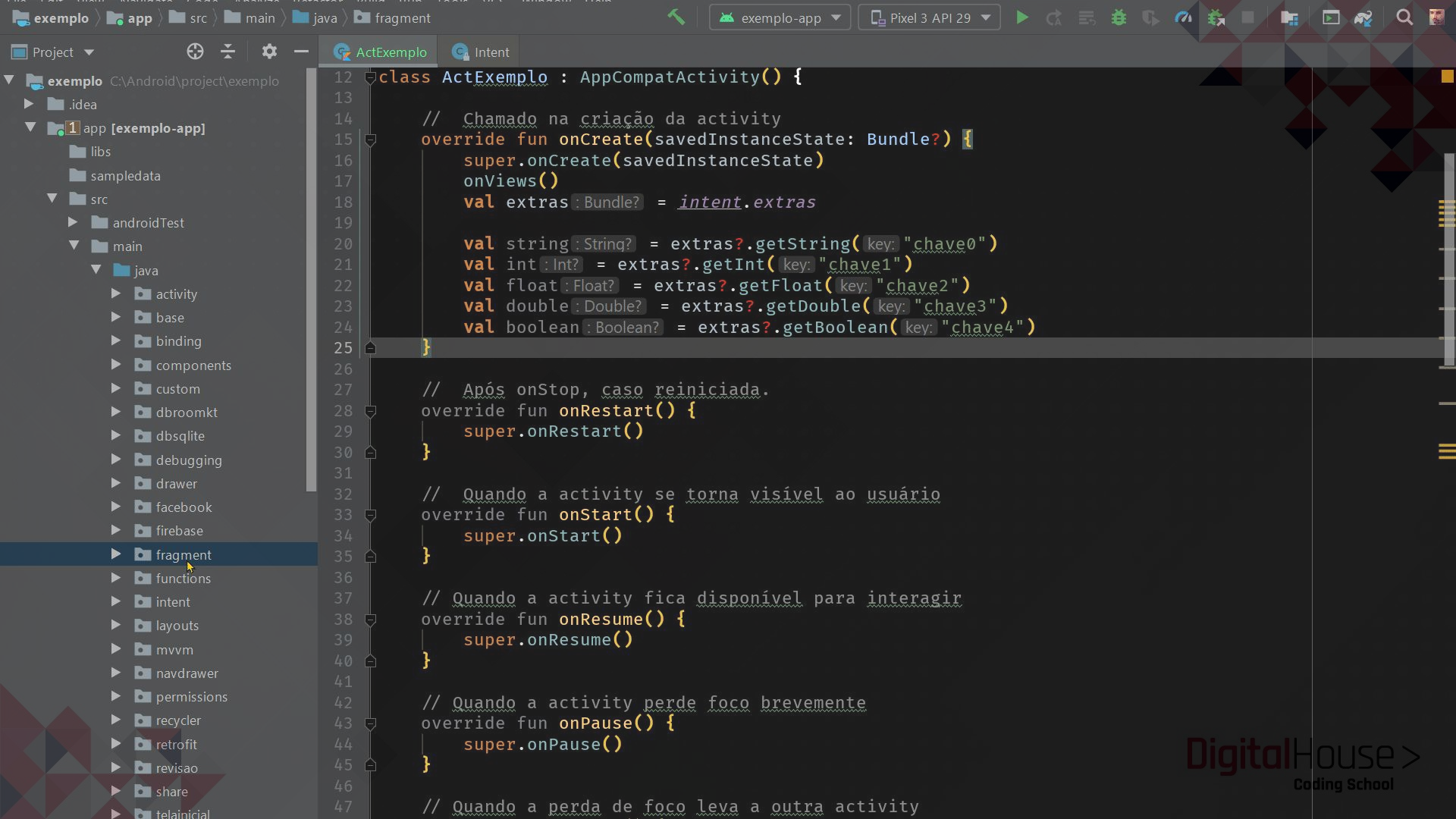
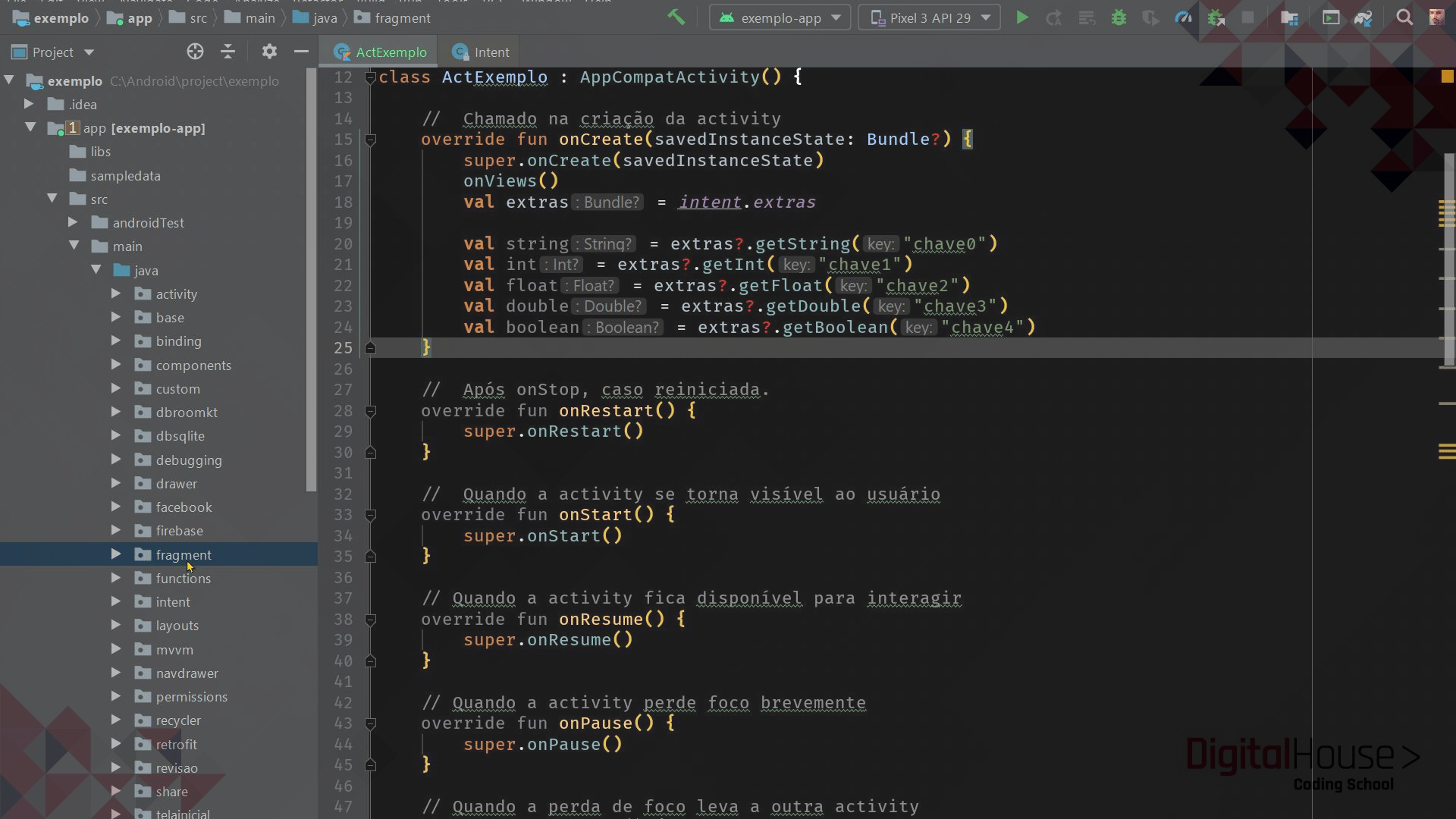
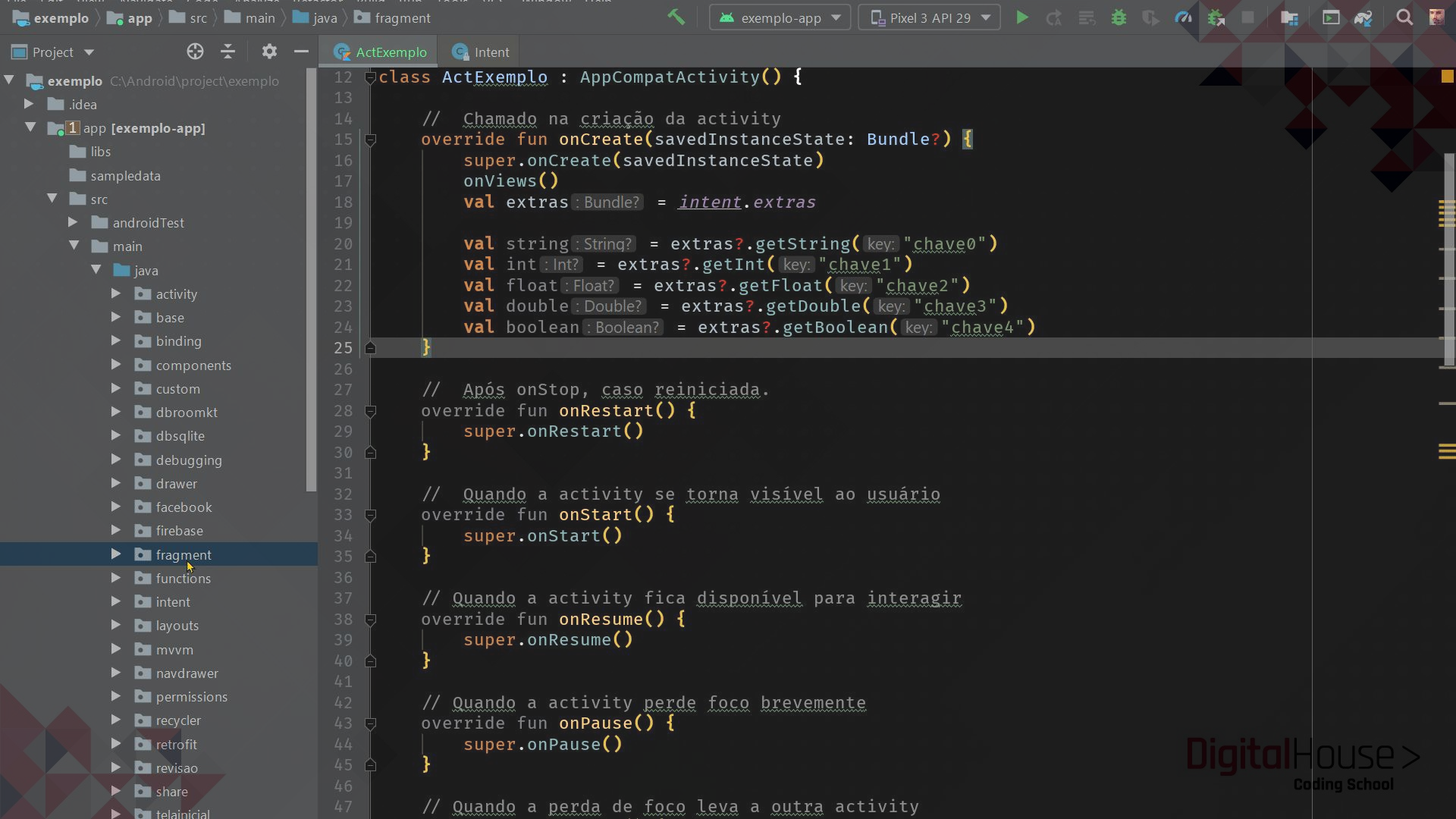
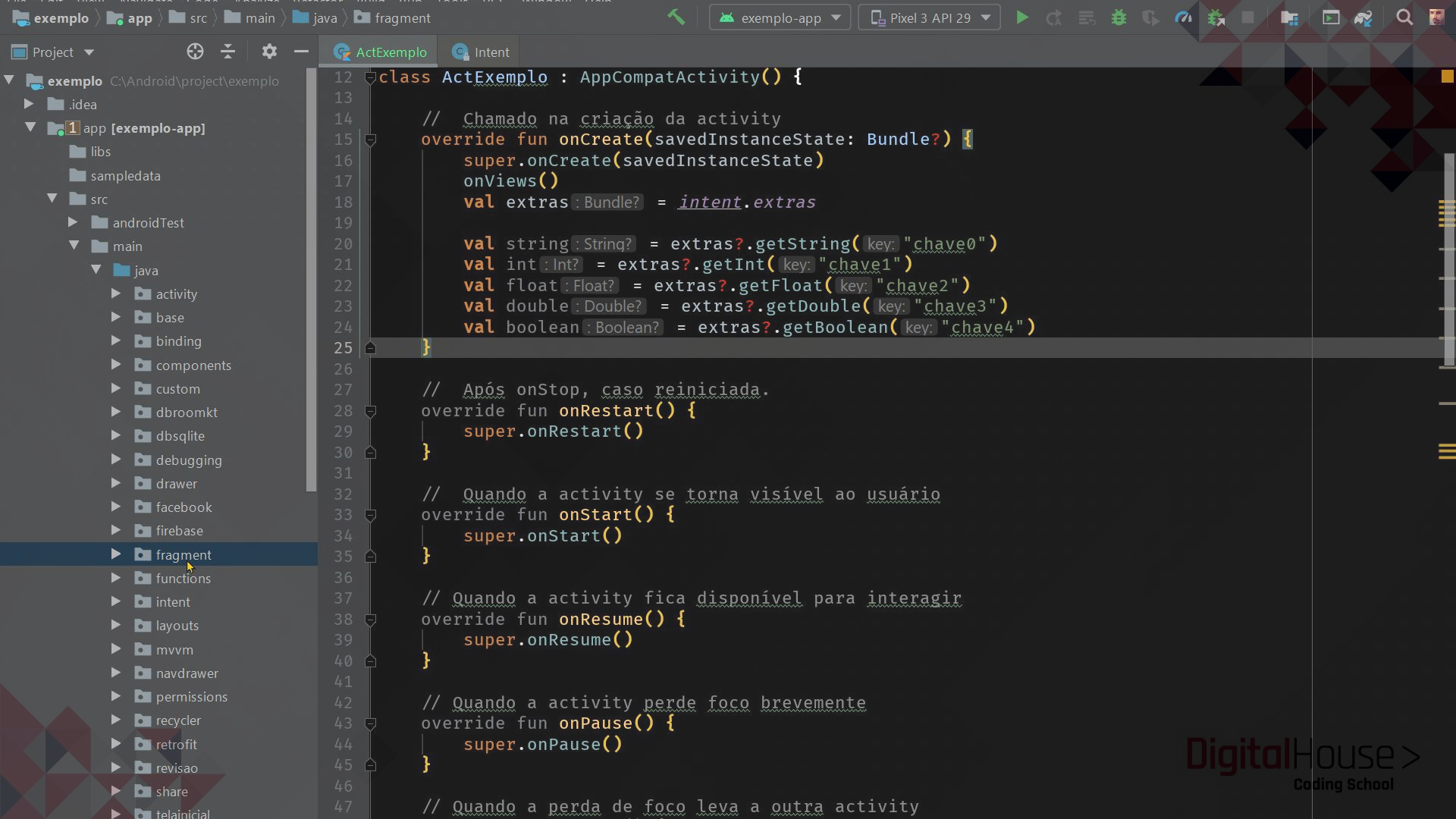
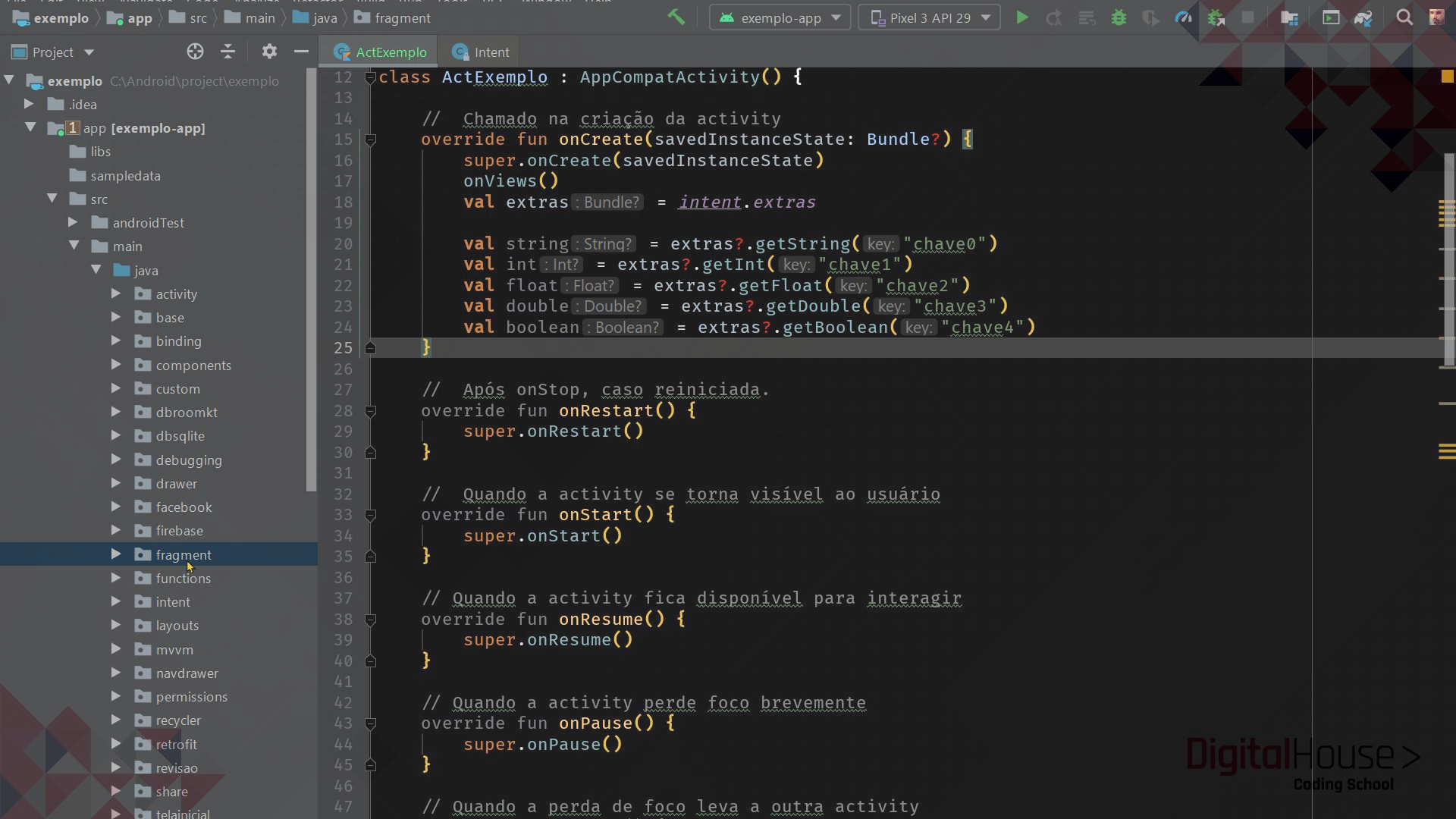
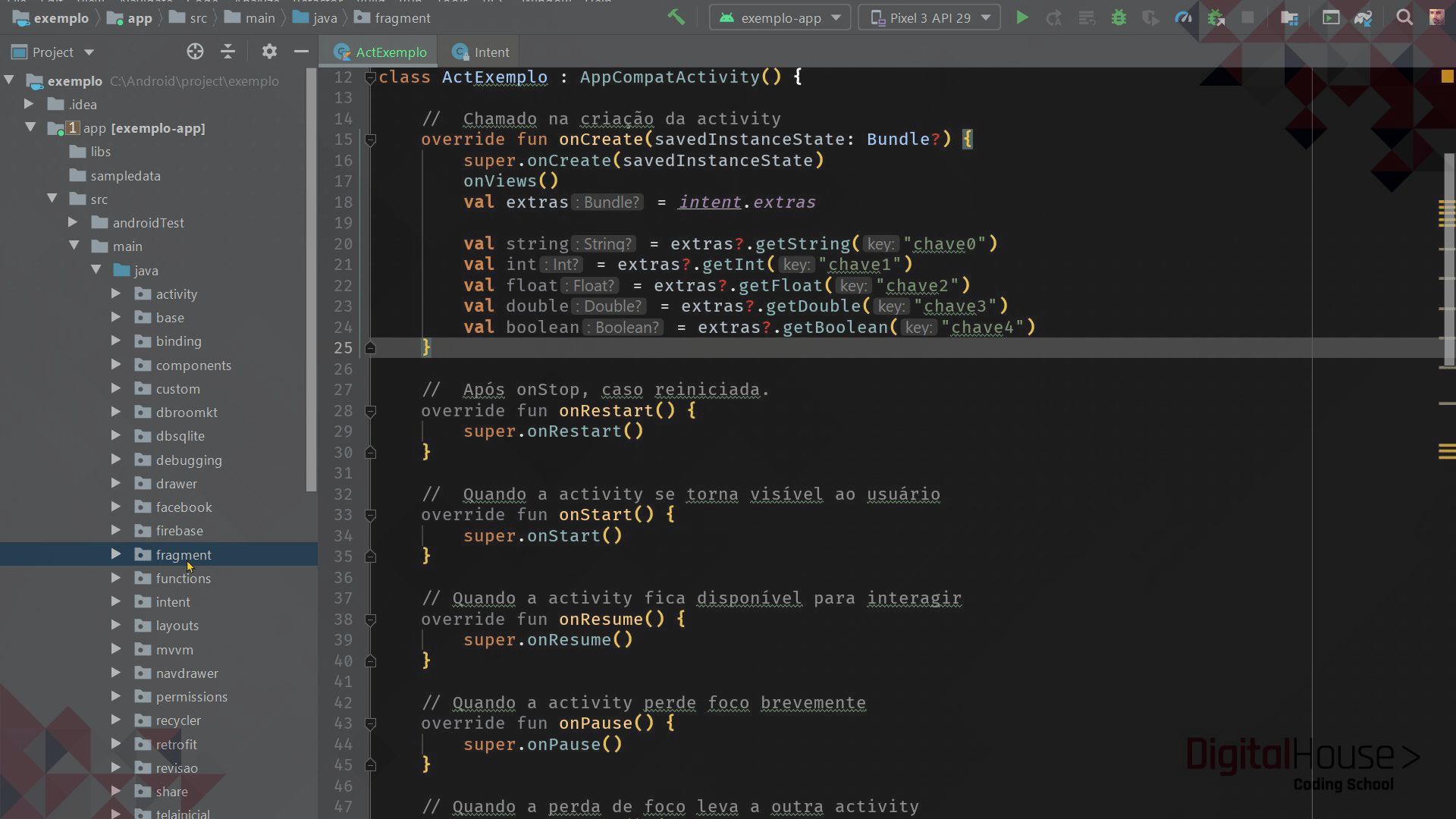
# Ciclo de vida

**onStop()** - Assim como no ciclo de vida da Activity, uma vez que o Fragment não está mais visível, há uma chance dele ser encerrado.

**onDestroyView()** - O onDestroyView é correspondente ao onDestroy da Activity e é chamado imediatamente antes do Fragment ser destruído. Ele funciona independente da Activity pai.

**onDestroy()** - chamado para fazer a limpeza final do estado do fragmento.

**onDetach()** - O onDetach é a última coisa que acontece no ciclo de vida, mesmo após o seu Fragment ser tecnicamente destruído.



## Diferenças

Activity	Fragment
Só pode rodar uma por vez	Pode ter várias rodando ao mesmo tempo
Funciona por si só	Precisa de uma Activity para ser apresentado
Código único	Possibilidade de reutilização

# Instanciando um Fragment numa Activity

Para colocarmos um **fragmento de tela** dentro de uma **Activity** precisamos iniciar uma **transação**, fazer o **replace** e **commitar**. Tudo isso através da classe **FragmentManager**:

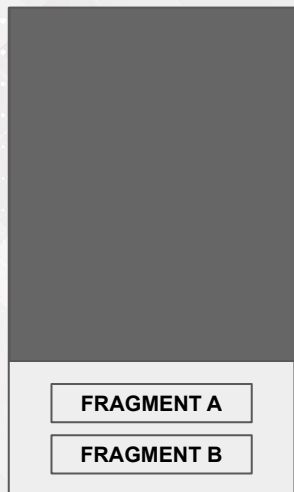
```
val manager = supportFragmentManager
val transaction = manager.beginTransaction()
transaction.add(R.id.fragmentContainer, MeuFragment())
transaction.commit()
```

# Exercício 1

## Transições entre Fragments

### Checklist:

- Criar um layout conforme o protótipo ao lado
- O cinza escuro representa a área onde o Fragment deve ser inserido
- Iniciar sem nenhum fragment
- Ao clicar no botão, abrir o Fragment referente a ele (terá dois fragments)
- Garantir que alternar entre eles irá funcionar sem quebrar o app





# Comunicação entre Fragments

Home

Home

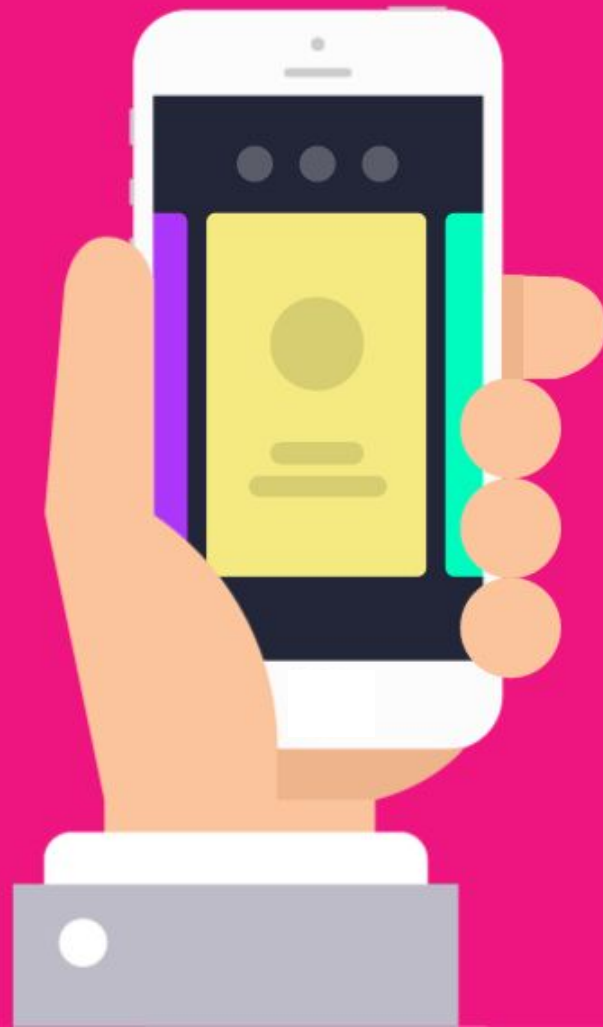





# A Interface


Criaremos uma interface, que será implementada por nossa Activity.

No `onAttach` de cada `Fragment`, com a referência à `Activity`, faremos o casting e trataremos como sendo nossa interface.





```
interface ActivityContract {  
    fun setTextVermelho(texto: String)  
    fun setTextVerde(texto: String)  
    fun setTextAzul(texto: String)  
}
```

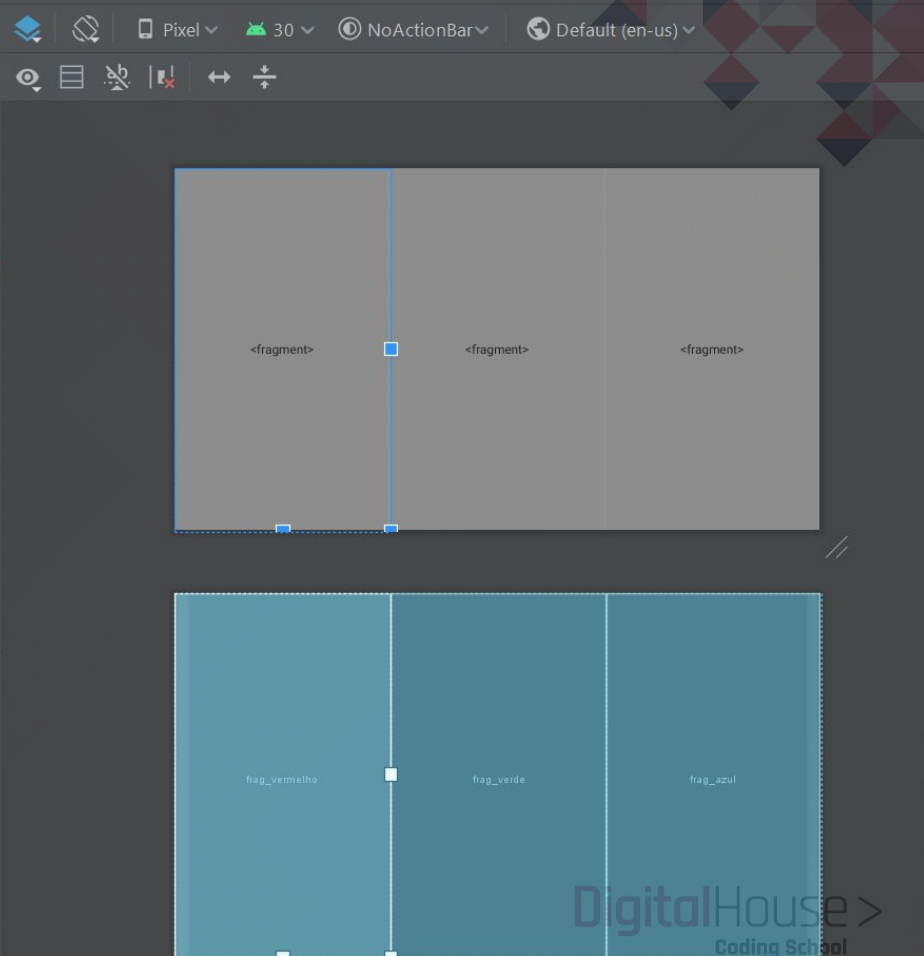


```
class ActComunicaFragKotlin : AppCompatActivity(), ActivityContract {  
  
    private var fragVermelho: FragVermelho? = null  
    private val fragVerde by lazy { findFrag<FragVerde>(R.id.frag_verde) }  
    private lateinit var fragAzul: Fragment  
  
    override fun onCreate(bundle: Bundle?) {  
        super.onCreate(bundle)  
        setContentView(R.layout.act_comunica_frag)  
        fragVermelho = findFrag(R.id.frag_vermelho)  
        fragAzul = findFrag(R.id.frag_azul)  
    }  
  
    private fun <T> findFrag(fragID: Int): T =  
        supportFragmentManager.findFragmentById(fragID) as T  
  
    override fun setTextVermelho(texto: String): Unit =  
        fragVermelho!!.setTextVermelho(texto)  
  
    override fun setTextVerde(texto: String): Unit =  
        fragVerde.setTextVerde(texto)  
  
    override fun setTextAzul(texto: String): Unit =  
        (fragAzul as FragAzul).setTextAzul(texto)  
}
```

```

1 <?xml version="1.0" encoding="utf-8"?>
2 <LinearLayout
3     xmlns:android="http://schemas.android.com/apk/res/android"
4     android:layout_width="match_parent"
5     android:layout_height="match_parent"
6     android:baselineAligned="false"
7     android:orientation="horizontal"
8     >
9
10    <fragment|
11        android:id="@+id/frag_vermelho"
12        android:name="fragment.FragVermelho"
13        android:layout_width="0dp"
14        android:layout_height="match_parent"
15        android:layout_weight="1"
16    />
17
18    <fragment
19        android:id="@+id/frag_verde"
20        android:name="fragment.FragVerde"
21        android:layout_width="0dp"
22        android:layout_height="match_parent"
23        android:layout_weight="1"
24    />
25
26    <fragment
27        android:id="@+id/frag_azul"
28        android:name="fragment.FragAzul"
29        android:layout_width="0dp"
30        android:layout_height="match_parent"
31        android:layout_weight="1"
32    />
33
34 </LinearLayout>

```



Nos fragmentos, podemos referenciar o contexto a partir do `onAttach`. Ou seja, a partir do momento que o fragmento foi vinculado a uma `activity`, é possível referenciar tal `activity` e tratá-la como `ActivityContract`.

```
class FragVermelho : Fragment() {  
    private lateinit var activity: ActivityContract  
  
    override fun onAttach(context: Context) {  
        super.onAttach(context)  
        if (context is ActivityContract) activity = context  
    }  
}
```



No `onViewCreated`, podemos configurar as Views. Observe que o click executa uma função da activity que comunicará com outro fragmento.

```
override fun onViewCreated(view: View, savedInstanceState: Bundle?) {  
    super.onViewCreated(view, savedInstanceState)  
    editText = view.findViewById(R.id.et_vermelho)  
    button = view.findViewById(R.id.button_vermelho)  
    textVermelho = view.findViewById(R.id.texto_vermelho)  
  
    button.setOnClickListener { it: View!  
        activity.setTextVerde(editText.text.toString())  
    }  
}
```

## Na activity:

```
override fun setTextVermelho(texto: String): Unit =  
    fragVermelho!!.setTextVermelho(texto)
```

```
override fun setTextVerde(texto: String): Unit =  
    fragVerde.setTextVerde(texto)
```

```
override fun setTextAzul(texto: String): Unit =  
    (fragAzul as FragAzul).setTextAzul(texto)
```



No fragmento:

```
fun setTextVermelho(texto: String?) {  
    textVermelho.text = texto  
}
```

## Exercício 2

### Passagem de informação entre Fragments

#### Checklist:

- Criar layout dos Fragments
- Inserir eles na activity
- Criar lógica
  - O usuário digitará seu nome e ano de nascimento, ao clicar em calcular irá ser exibido:  
“Fulano, você tem X ano(s)”
- Como bônus tratar pluralização da string ano; Se o usuário digitou uma data futura mostrar a mensagem:  
“Fulano ainda não nasceu”;
- Mostrar sempre que possível, feedback de erro

Nome

Ano Nasc

CALCULAR

XXX, você tem Y ano(s)

Fragment A

Fragment B

#### DICA PARA OBTER ANO ATUAL:

```
Calendar.getInstance().get(Calendar.YEAR)
```

