实验考试

重要说明：

1. 开始答题前请仔细阅读左侧的答题说明
2. 答题时需要保持网络连接稳定
3. 上传代码截图时请注意，在看到上传成功的弹窗之前不要做其他操作

友情提示：华为的考试系统UI+UX体验绝对的垫底，实习生都写不了这么烂，各位佬千万沉住气，砸坏的鼠标是自己的！

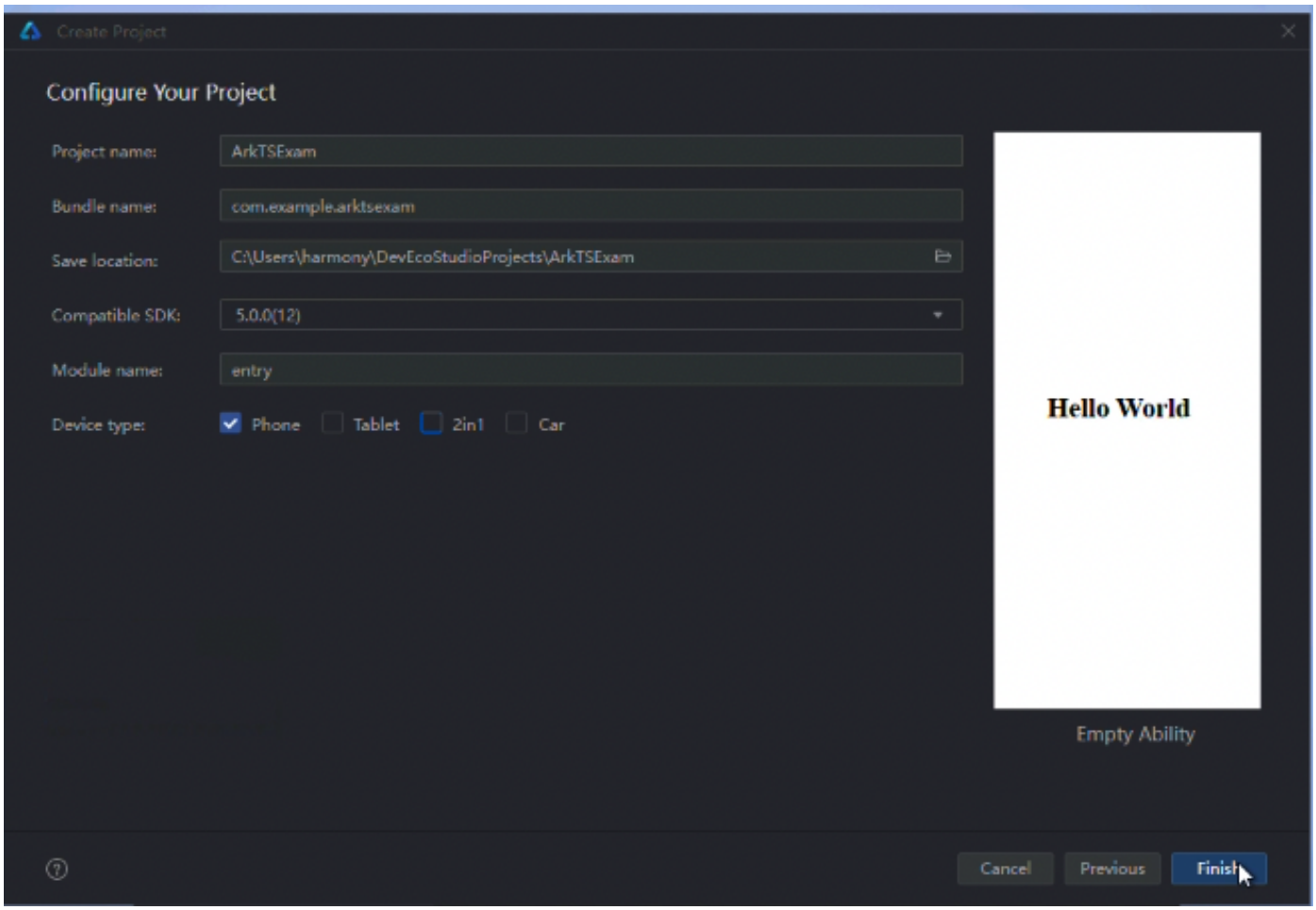
实验一：根据题目描述，使用DevEco Studio编写ArkTS语言程序完成如下图页面构建（30分）

任务1：补全数据模型数组项（5分）

得分点：能正确对指定类中创建对象进行构造初始化。

请使用如下配置自行创建API为12的应用工程：
应用工程创建时的相关配置如下表：

应用模板	**Empty Ability**
项目名称 (Project name)	ArkTSExam
Compile SDK	5.0.0(API 12)
应用模型 (Model)	Stage
开发语言 (Language)	ArkTS
设备类型(Device type)	phone



创建成功后，请打开默认生成的Index.ets文件，并复制如下代码，此时文件中已经创建好了DataSources类，请对该类使用正确的构造方法进行对象初始化。

本任务中需注意以下内容：

1. 使用错误的命名不得分
2. 使用错误的属性名不得分
3. 多或者少对象属性不得分
4. 得分条件：请填写代码后，对答案代码以及上下文代码进行完整截图。

```
class DataSources {
  id: string; // ID
  title: string; // 标题
  brief: string; // 其他描述

  // 请填写正确代码块
  constructor(id: string, title: string, brief: string) {
    this.id = id;
    this.title = title;
    this.brief = brief;
  }
}
```

任务2：创建自定义组件，并补全代码（10分）

得分点：

1. 自定义组件DataCard中数据来源应使用`任务一`中创建数据源数组DataSources。
2. 必须通过创建自定义組件DataCard构建页面。

   定义完数据模型后，请自行分析布局并优先构建组成页面UI的自定义组件DataCard。且自定义组件DataCard中数据来源应使用`任务一`中创建数据源数组Data Sources。
   DataCard自定义组件布局如下图所示：

   下文已提供自定义组件DataCard构建过程中所需具体样式属性参数。请根据这些信息构建出正确的自定义组件。

本任务中需注意以下内容：

1. 自定义组件声明错误不得分
2. 数据源没有使用DataSources 而是直接填写数据不得分
3. 容器组件选择错误不得分
4. Text组件横线处填写错误不得分
5. 得分条件：请填写代码后，对答案代码以及上下文代码进行完整截图。

```
// 请先完成以下步骤：
// 1. 在pages下创建文件DataCard.ets
// 2. 导出Index.ets中的class
// 3. 在DataCard.ets中引入class
// 4. 对照填入以下代码

import {DataSources} from './Index'

// 自定义组件构建请补全代码，填写正确装饰器以及函数
@Component
export struct DataCard{
// 使用@prop装饰器连接数据源DataSources。
@Prop dataSources: DataSources

build() {
Row() {
      Image($r('app.media.startIcon'))//系统自动提供图标
        .width(80)
        .height(80)
        .margin({ right: 20 })
```

```
// 请根据提供UI页面图样式，选择正确的容器组件
      Column(){
    //Text组件参数空缺处请补全
        Text(this.dataSources.title)
          .fontSize(20)
          .margin({ bottom: 8 })
        Text(this.dataSources.brief)
          .fontSize(16)
          .fontColor(Color.Gray)
          .margin({ bottom: 8 })
      }
      .alignItems(HorizontalAlign.Start)
      .width('80%')
      .height('100%')
    }
    .padding(20)
    .borderRadius(12)
    .backgroundColor('#FFECECEC')
    .height(120)
    .width('100%')
    .justifyContent(FlexAlign.SpaceBetween)
  }
}
```

任务3：使用渲染控制语法，构建页面（15分）

得分点：

1. 必须正确使用`任务二`中创建的DataCard组件。
2. 必须通过渲染控制语法创建页面组件DataSourcesListView。构建完成自定义组件DataCard后，请使用渲染控制语法以及子组件DataCard构建组件DataSourcesListView。
3. 如下所示，已提供自定义组件DataSourcesListView构建过程中所需具体样式属性参数，请根据这些信息构建出正确的页面。

本任务中需注意以下内容：

1. 自定义组件声明错误不得分
2. 数据源没有使用DataSources，而是直接填写数据不得分
3. 没有使用循环渲染构建页面不得分
4. 自定义组件间样式书写不正确不得分
5. 得分条件：请填写代码后，对答案代码以及上下文代码进行完整截图。

```
// 请先完成以下步骤：
// 1. 在pages下创建文件DataSourcesListView.ets
// 2. 在DataSourcesListView.ets中引入DataCard和DataSources
// 3. 对照填入以下代码

import { DataCard } from './DataCard'
```

```
import { DataSources } from './Index'

// 自定义组件构建请补全代码，填写正确装饰器以及函数
@Entry
@Component
struct DataSourcesListView {
  // 根据DataSources数据模型构建数组DataSourcesList
  @State DataSourcesList: DataSources[] = [
    new DataSources("001", "标题一", "具体描述一"),
    new DataSources("002", "标题二", "具体描述二"),
    new DataSources("003", "标题三", "具体描述三"),
    new DataSources("004", "标题四", "具体描述四"),
    new DataSources("005", "标题五", "具体描述五")
  ]

  // 自定义组件构建请补全代码，填写正确装饰器以及函数
  build() {
    Column({space: 5}) {
      List() {
        // 使用正确的循环渲染语法以及利用自定义组件DataCard构建页面
        // 注意：样式要求自定义组件之间顶部外边距距离为20
        ForEach(this.DataSourcesList, (item: DataSources) => {
          ListItem() {
            DataCard({dataSources: item})
              .margin({top: 20})
          }
        },(item: DataSources)=>item.id)
      }
      .padding(20)
      .scrollBar(BarState.Off)
    }
    .width('100%')
  }
}
```

实验二：根据题目描述，使用DevEco Studio实现启动应用内的UIAbility并获取返回结果（35分）

任务1：创建UIAbility并指定启动页面（8分）

本实验包含两个UIAbility，每个UIAbility关联一张Page页面。EntryAbility与其关联的Page页面Index.ets创建好HarmonyOS工程后默认提供。

接下来，请自行创建好一个UIAbility命名为SecondAbility，再创建好一个页面名叫Second，使其成为SecondAbility的指定启动页面。

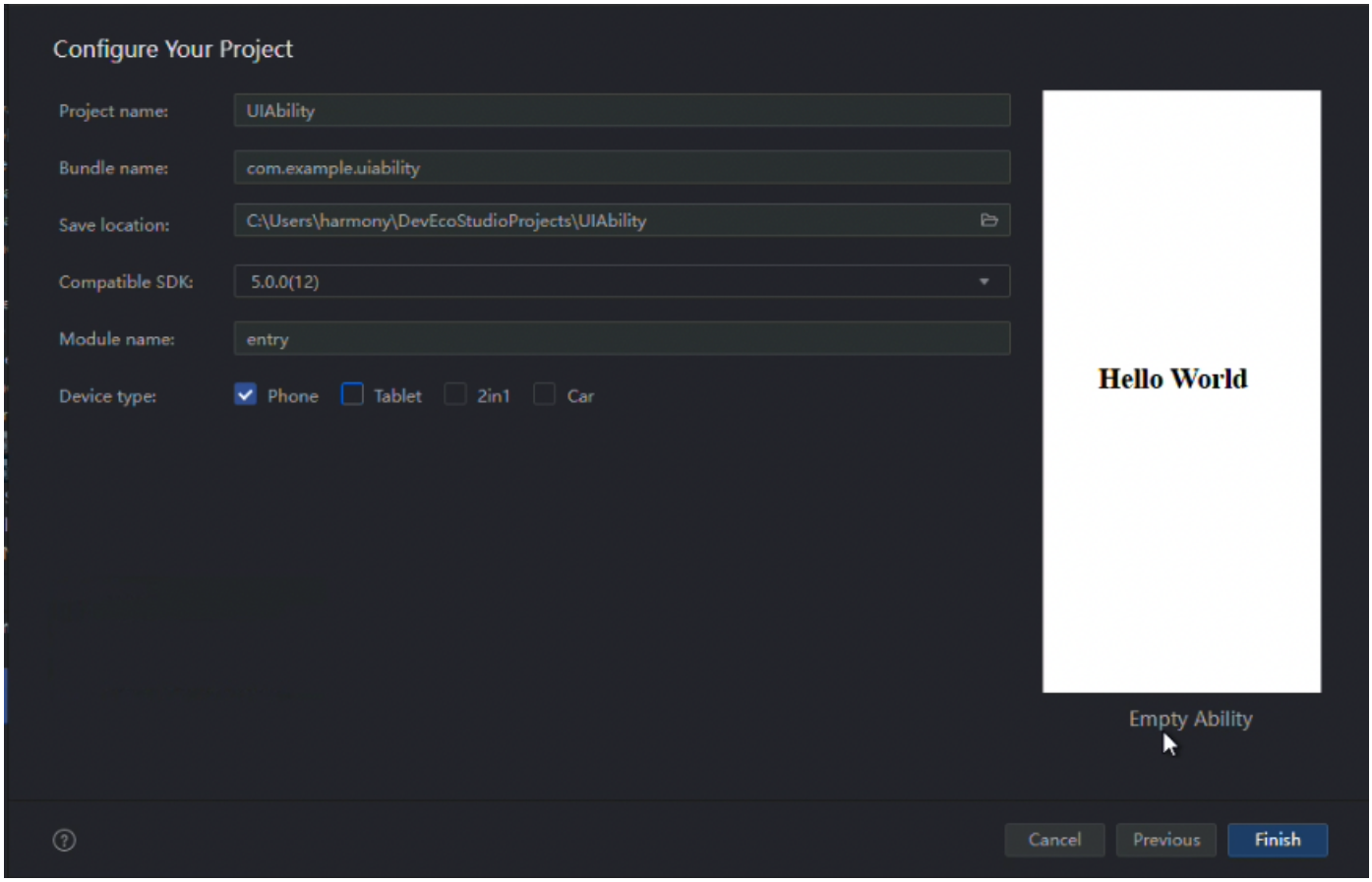
注释：该实验工程为AP112的应用工程。

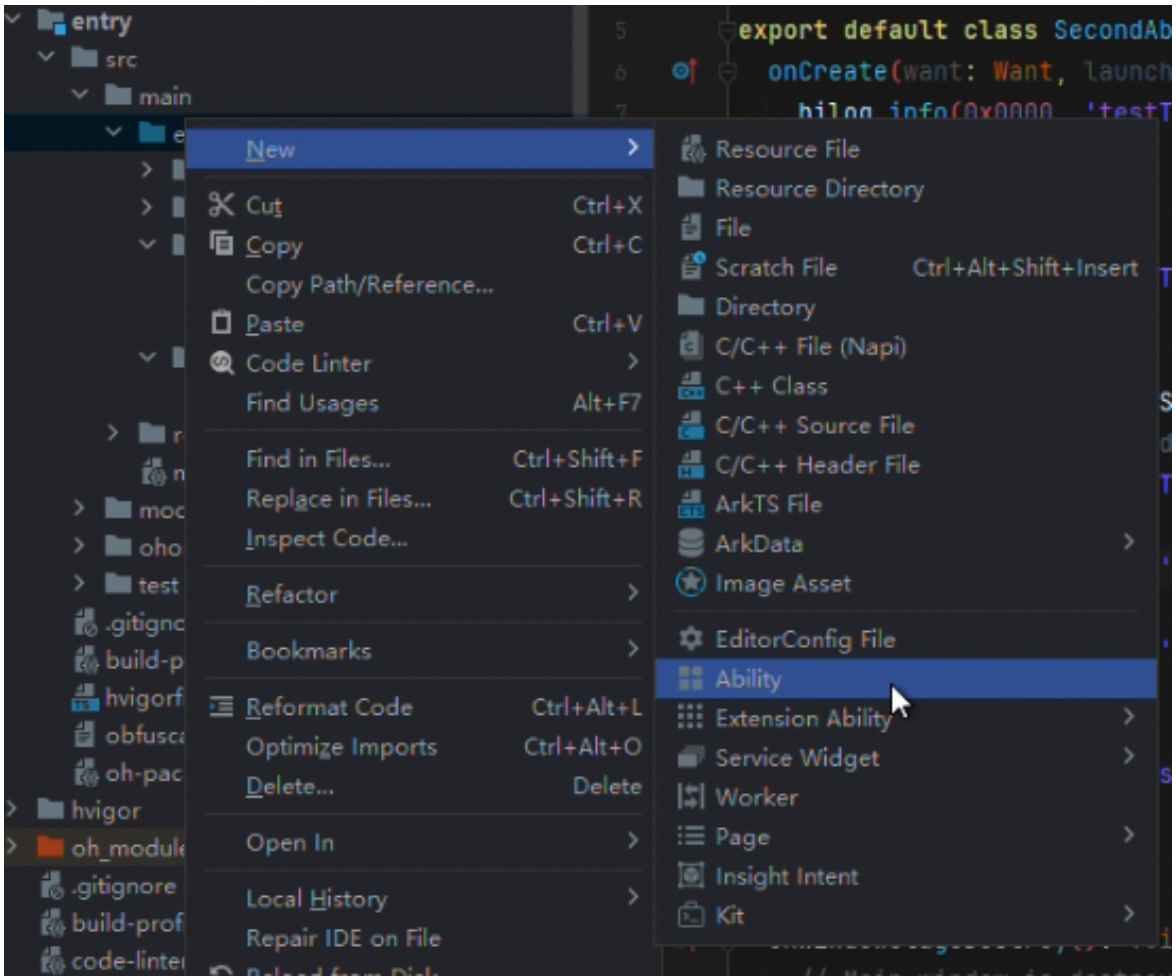
得分点：成功创建SecondAbility并指定其启动页面为Second.ets详细操作步骤，请务必截图。

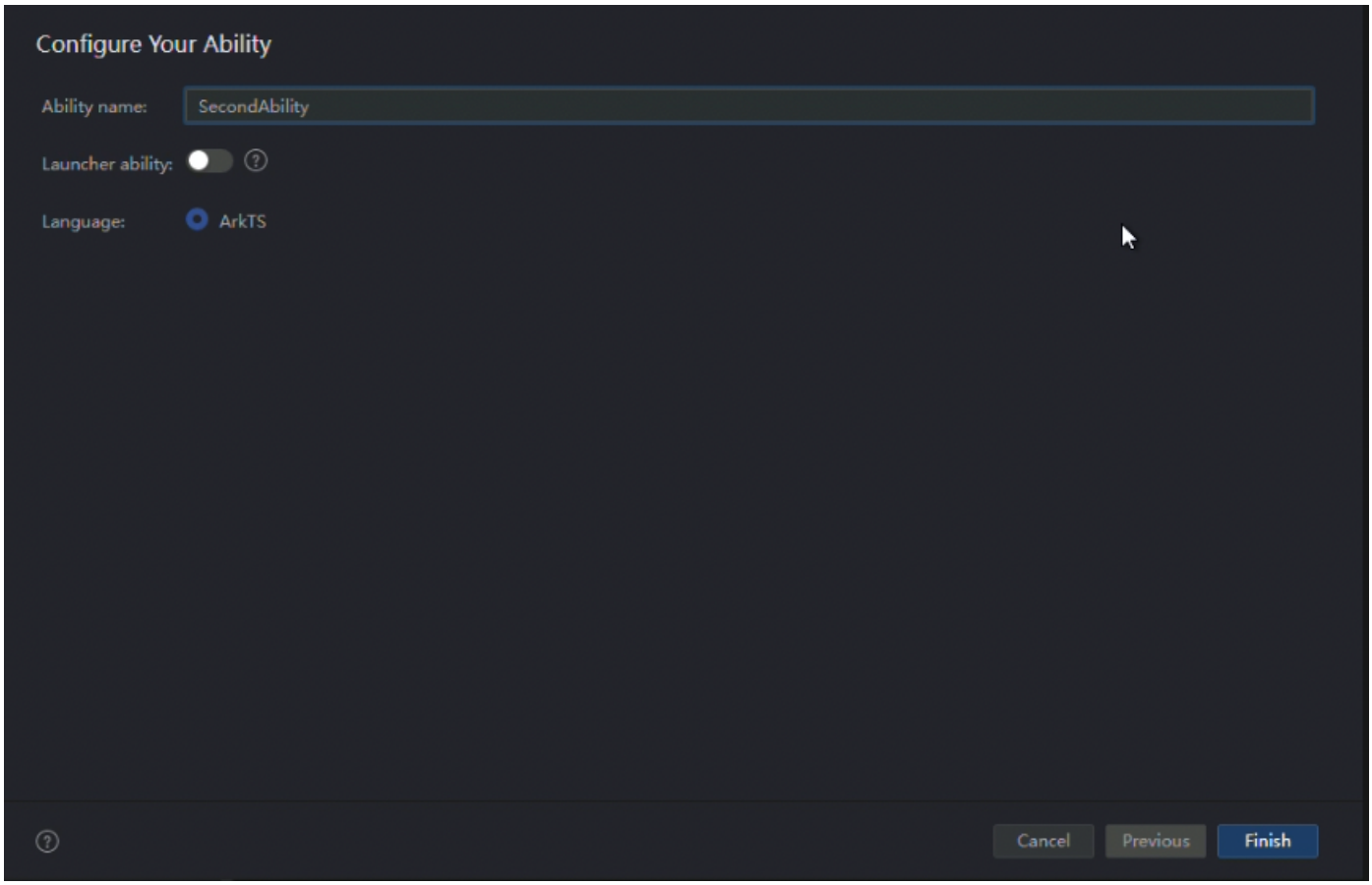
得分条件：

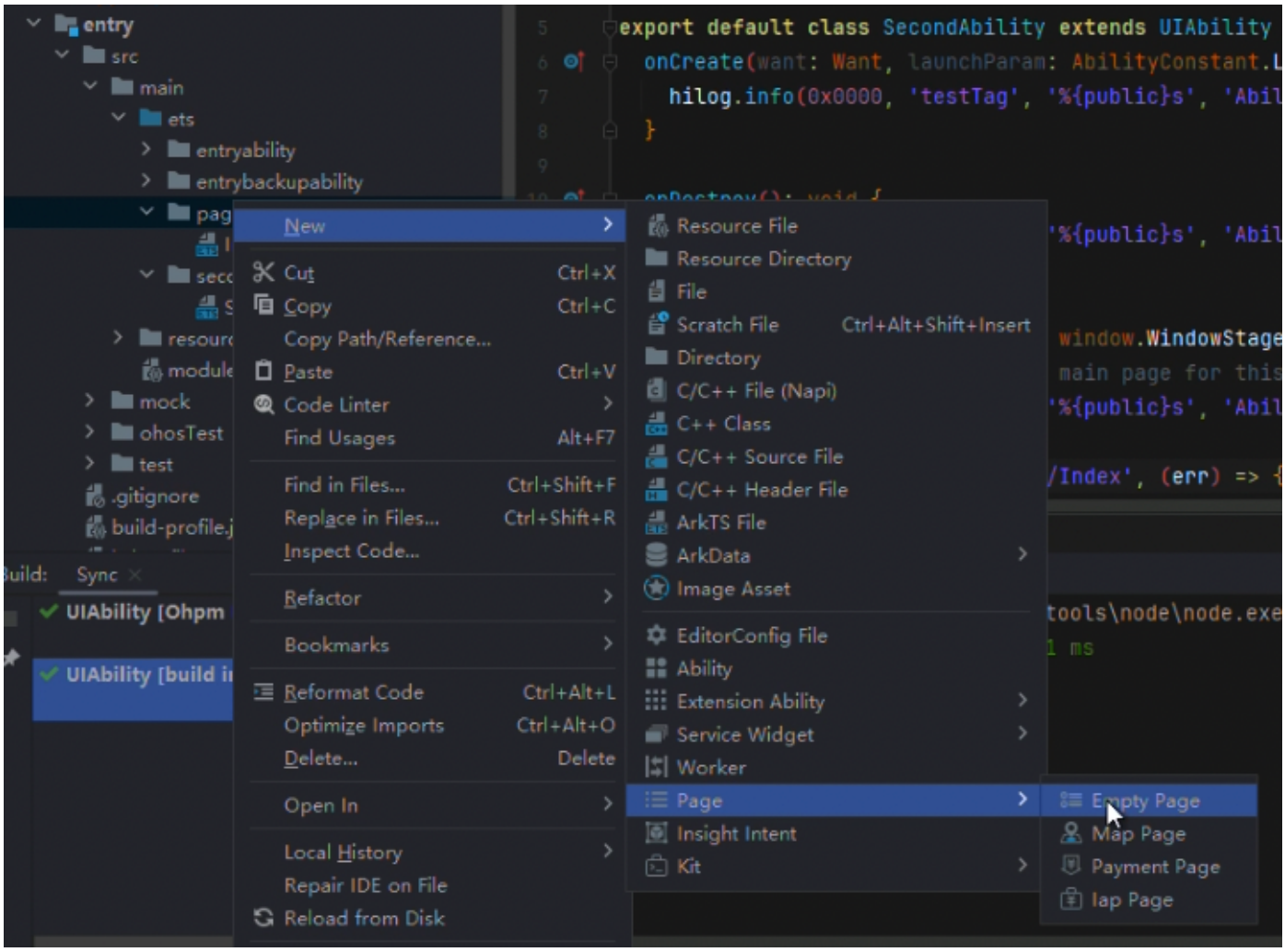
1. 创建UIAbility步骤截图（请注意必须包含UIAbility命名截图，缺失不得分。）
2. 创建Second.ets页面步骤截图（请注意必须包含页面命名截圖，缺失不得分。）
3. 书写指定启动页面代码并截图，截图需能展示在哪个文件中书写的此段代码（请注意此处代码截图需带上左侧目录结构，缺失不得分。）

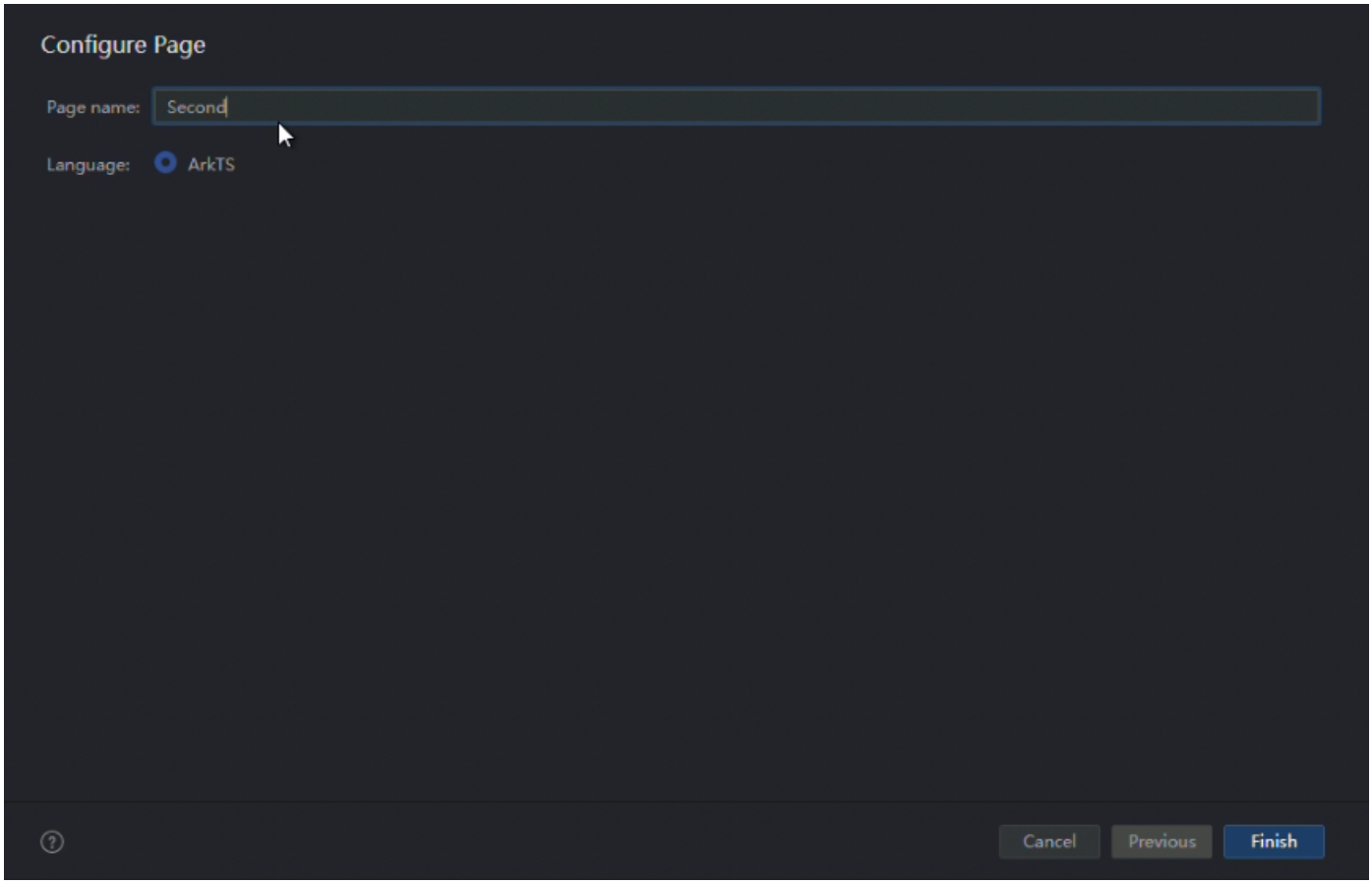
请新建一个project完成，并按照以下顺序进行操作，截图时请根据得分条件确认是否需要侧边目录

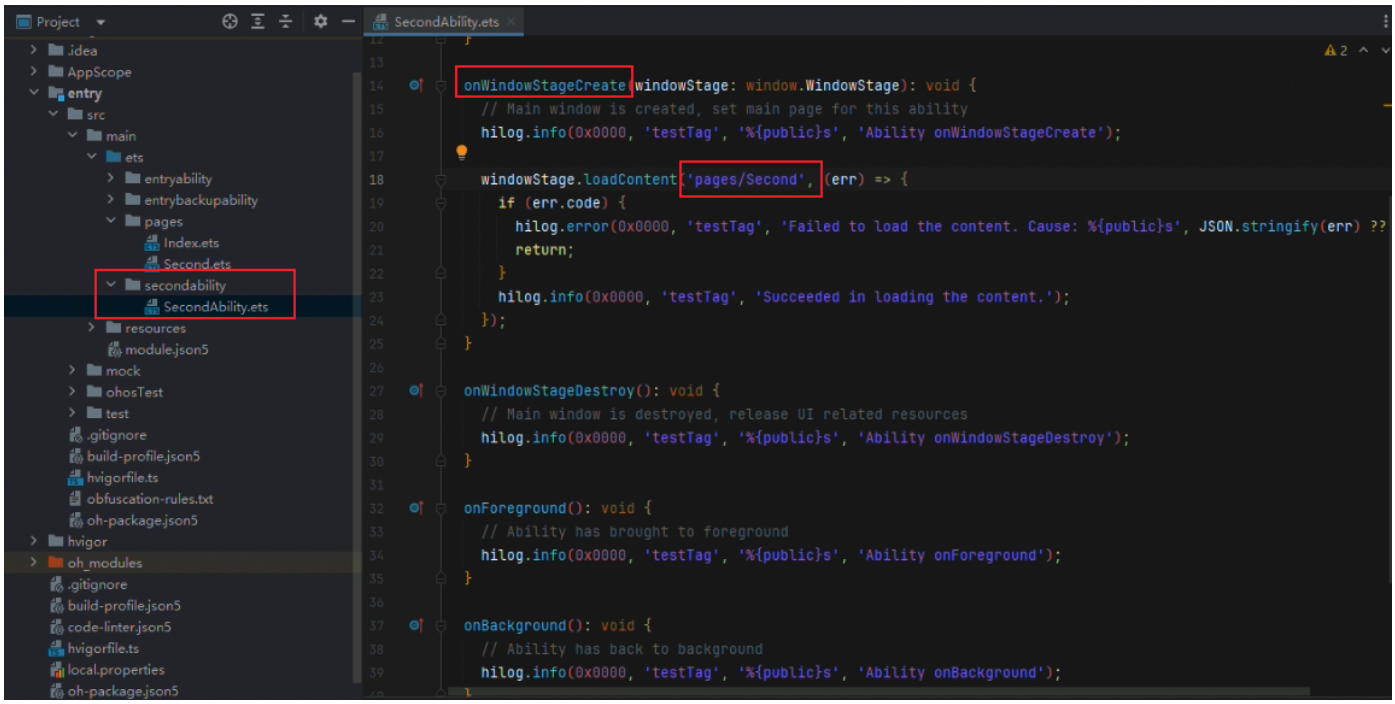



entry
src
main
export default class SecondAb
onCreate(want: Want, launch
New
Cut Ctrl+X
Copy Ctrl+C
Copy Path/Reference...
Paste Ctrl+V
Code Linter
Find Usages Alt+F7
Find in Files... Ctrl+Shift+F
Replace in Files... Ctrl+Shift+R
Inspect Code...
Refactor
Bookmarks
Reformat Code Ctrl+Alt+L
Optimize Imports Ctrl+Alt+O
Delete... Delete
Open In
Local History
Repair IDE on File
Resource File
Resource Directory
File
Scratch File Ctrl+Alt+Shift+Insert
Directory
C/C++ File (Napi)
C++ Class
C/C++ Source File
C/C++ Header File
ArkTS File
ArkData
Image Asset
EditorConfig File
Ability
Extension Ability
Service Widget
Worker
Page
Insight Intent
Kit


Configure Your Ability
Ability name: SecondAbility
Launcher ability:
Language: ArkTS
Cancel
Previous
Finish


entry
src
main
ets
entryability
entrybackupability
New
Cut Ctrl+X
Copy Ctrl+C
Copy Path/Reference...
Paste Ctrl+V
Code Linter
Find Usages Alt+F7
Find in Files... Ctrl+Shift+F
Replace in Files... Ctrl+Shift+R
Inspect Code...
Refactor
Bookmarks
Reformat Code Ctrl+Alt+L
Optimize Imports Ctrl+Alt+O
Delete... Delete
Open In
Local History
Repair IDE on File
Reload from Disk
Resource File
Resource Directory
File
Scratch File Ctrl+Alt+Shift+Insert
Directory
C/C++ File (Napi)
C++ Class
C/C++ Source File
C/C++ Header File
ArkTS File
ArkData
Image Asset
EditorConfig File
Ability
Extension Ability
Service Widget
Worker
Page
Insight Intent
Kit
Empty Page
Map Page
Payment Page
Iap Page
export default class SecondAbility extends UIAbility
onCreate(want: Want, launchParam: AbilityConstant.L
hilog.info(0x0000, 'testTag', '%{public}s', 'Abil
Sync
UIAbility [Ohpm
UIAbility [build i


Configure Page
Page name: Second
Language: ArkTS
Cancel
Previous
Finish



任务2：启动应用内的SecondAbility并传递参数（15分）

本实验代码中，已经将Index.ets页面的基本UI界面提前构建完毕。接下来，请补全Apply(方法，使其可以完成启动SecondAbility并传递参数的功能。

最终实现如下功能效果：Index页面中存在一个文本输出框，当用户在其中输出文本，并点击提交按钮。应用会将用户所输入文本在Second页面上显示。

```
// 1. 请填入正确导入代码
import { common, Want } from '@kit.AbilityKit';
import Prompt from '@system.prompt';
import { BusinessError} from '@kit.BasicServicesKit';

const RESULT_CODE:number = 1000;

@Entry
@Component
struct Index {
  @State newData: string = '';
  // 是否提交
  @State isApply: boolean = false;

  build() {
    Column() {
      // 2. 添加正确的事件从而完善TextInput组件,确保将用户输入信息
      TextInput({
        placeholder: "请输入信息",
        text: this.newData,
      }).onChange((value) => {
        this.newData = value;
      })
```

```
24
25        Button('提交申请', { type: ButtonType.Normal })
26          .onClick(() => {
27            if (!this.isApply){
28              this.Apply();
29            } else {
30              Prompt.showToast({ message: '信息已答复，不能重复提交' });
31            }
32          })
33      }
34    }
35
36    Apply() {
37      // 3. 获取上下文
38      let context = getContext(this) as common.UIAbilityContext
39      // 4. 创建want对象，将要传递的自定义参数放入其中
40      let wantInfo: Want = {
41        deviceId: '', // 为空代表本设备
42        bundleName: 'com.example.uiability', // 应用名
43        abilityName: 'SecondAbility', // 目标Ability
44        parameters: {
45          // 传递的参数
46          info: this.newData
47        }
48      }
49      if (this.newData !== '' && this.newData !== ' ') {
50        context.startAbilityForResult(wantInfo).then((data) => {
51          if(data?.resultCode === RESULT_CODE) {
52            let result = data.want?.parameters?.info;
53          } else {
54            console.info('未能获取到返回结果');
55          }
56        })
57      }
58    }
59  }
```

任务3：接收EntryAbility传递的参数（6分）

在正确的生命周期回调中接收EntryAbility在启动SecondAbility时传递过来的自定义参数（即文本输入框信息），并将接收到的参数存入AppStorage对象中。将参数存放在AppStorage对象中后，即可在Second.ets页面中显示改文本。

得分点：

1. 成功启动接收EntryAbility专递的参数，需要选择正确的文件中的正确的生命周期回调函数中填写正确代码
2. 得分条件：填空处代码，连带上下文进行截图用于评审得分。（请注意此处代码截图需带上左侧目录结构，缺失不得分。）

```
export default class SecondAbility extends UIAbility {
  onCreate(want: Want, launchParam: AbilityConstant.LaunchParam): void {
    // 1. 请输入正确代码
    // 导入
    // 接收EntryAbility传递过来的参数
    let SecondAbilityInfo = want
    if(SecondAbilityInfo?.parameters?.info){
      // 解析参数
      let info = SecondAbilityInfo.parameters.info as string
      // 看解析好的参数信息存到AppStorage中，方便在批假页面使用
      AppStorage.setOrCreate('info', info)
    }
  }
```

任务4：参数在Second页面中展示。(6分）

得分点：

1. 成功在Second页面中获取EntryAbility传递过来并存放在AppStorage中的参数，以进行页面显示。
2. 得分条件：填空处代码，连带上下文进行截图用于评审得分。

```
// 获取EntryAbility传递过来存储在AppStorage中的参数
let INFO = AppStorage.get('info') as string

@Entry
@Component
struct Second {
  // 确保每次从EntryAbility传递过来的参数都能刷新到UI
  @StorageLink('info') info: string = INFO
  build() {
    Column(){
      // 页面标题
      Text(this.info)
    }
  }
}
```

实验三：开发ArkTS卡片（35分）

任务1：创建ArkTS卡片（5分）

得分点：

能在Harmonyos应用工程中正确创建基于ArkTS UI的服务卡片，卡片创建成功后，提供应用工程目录的截图。

本任务中需注意以下内容：

1. 本考试任务需要考生自行创建一个HarmonyOS应用工程，创建应用工程时的相关配置如下：

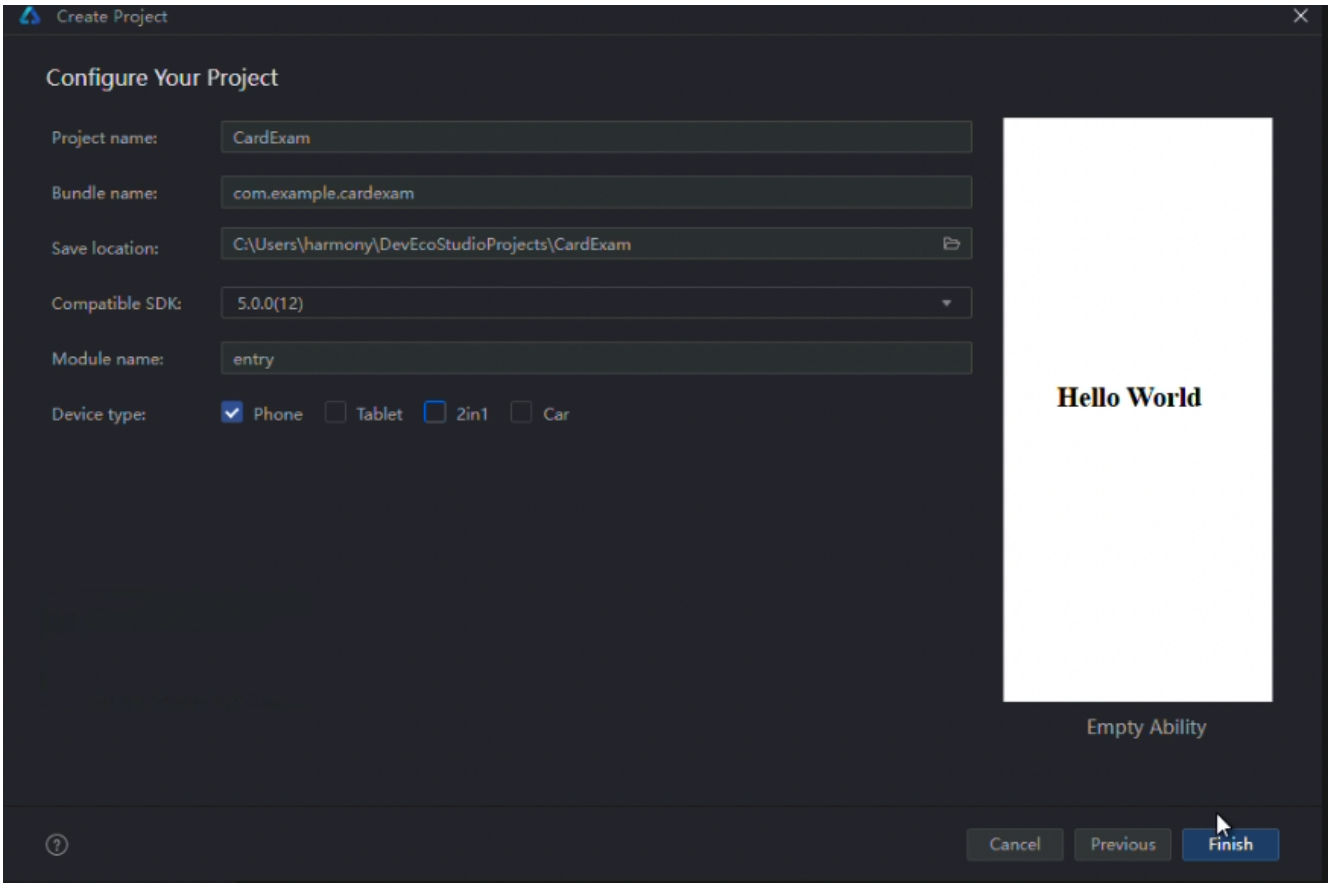
应用模板	**Empty Ability**
项目名称 (Project name)	CardExam
Compile SDK	5.0.0(API 12)
应用模型 (Model)	Stage
开发语言 (Language)	ArkTS
设备类型(Device type)	phone

2. 本考试任务需要考生自行创建ArkTS卡片，创建ArkTS卡片时的相关配置如下表：

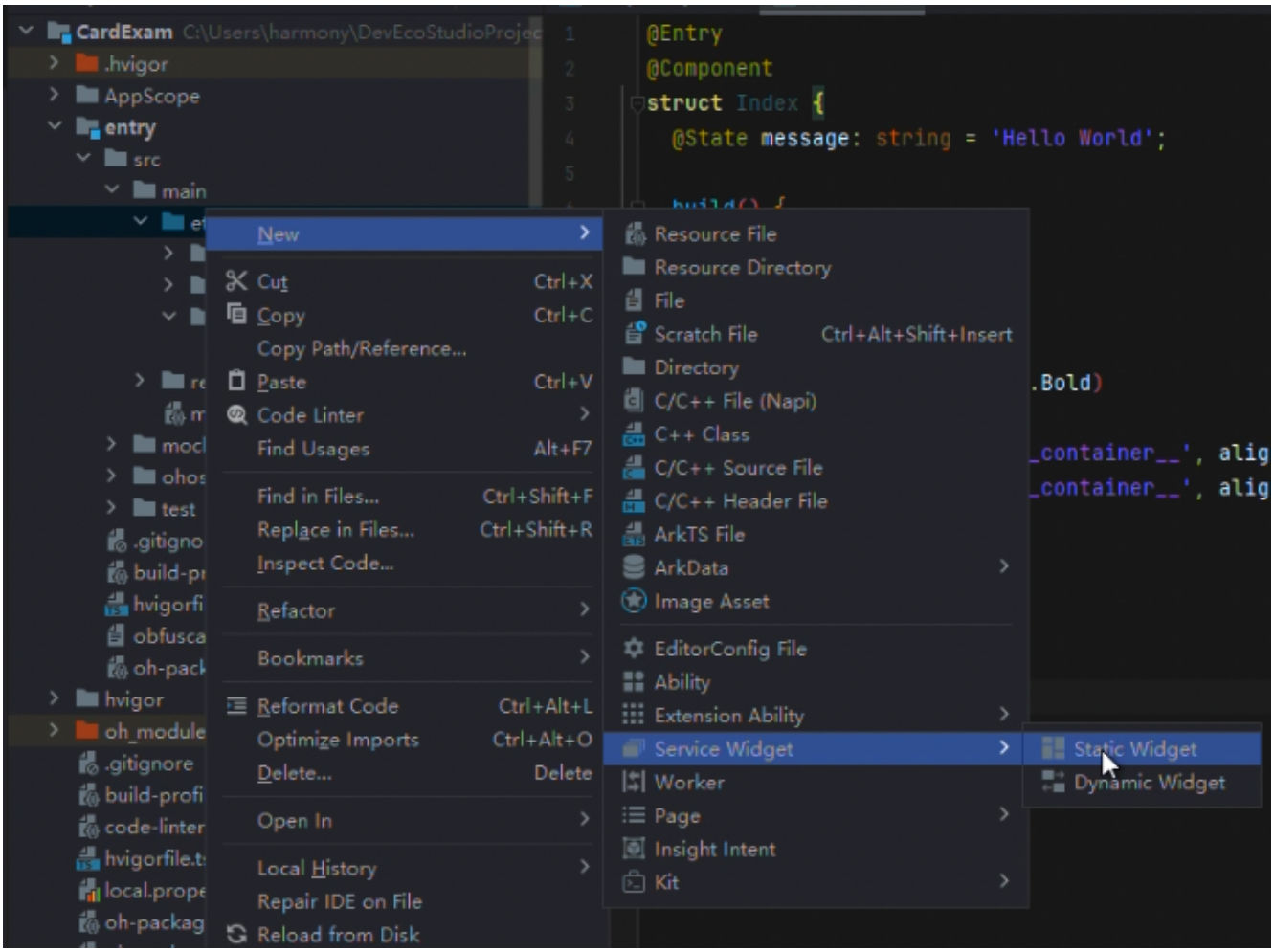
卡片模板	**Hello World**
卡片名称	Exam
开发语言	ArkTS
卡片规格	2x2

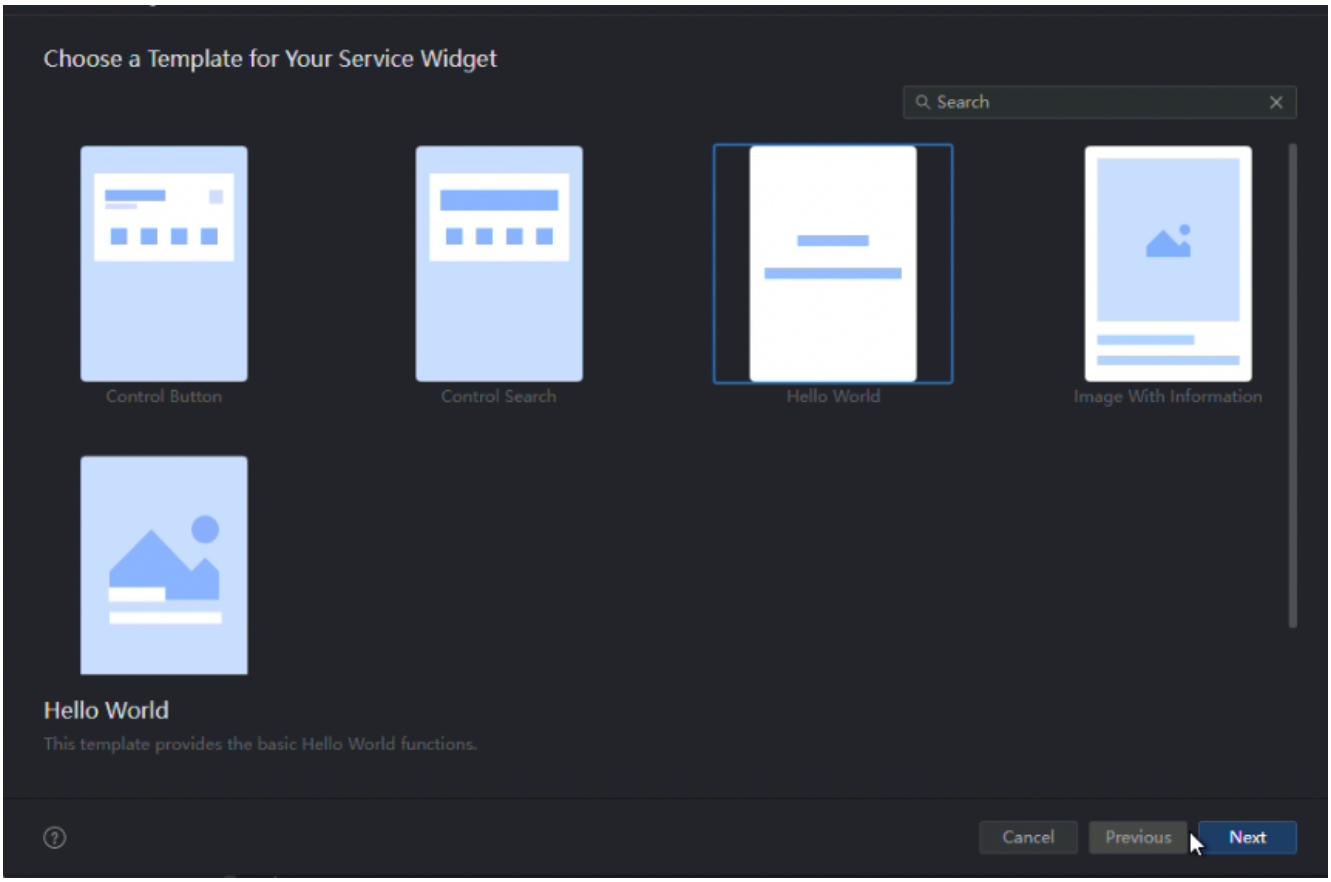
详细操作步骤：

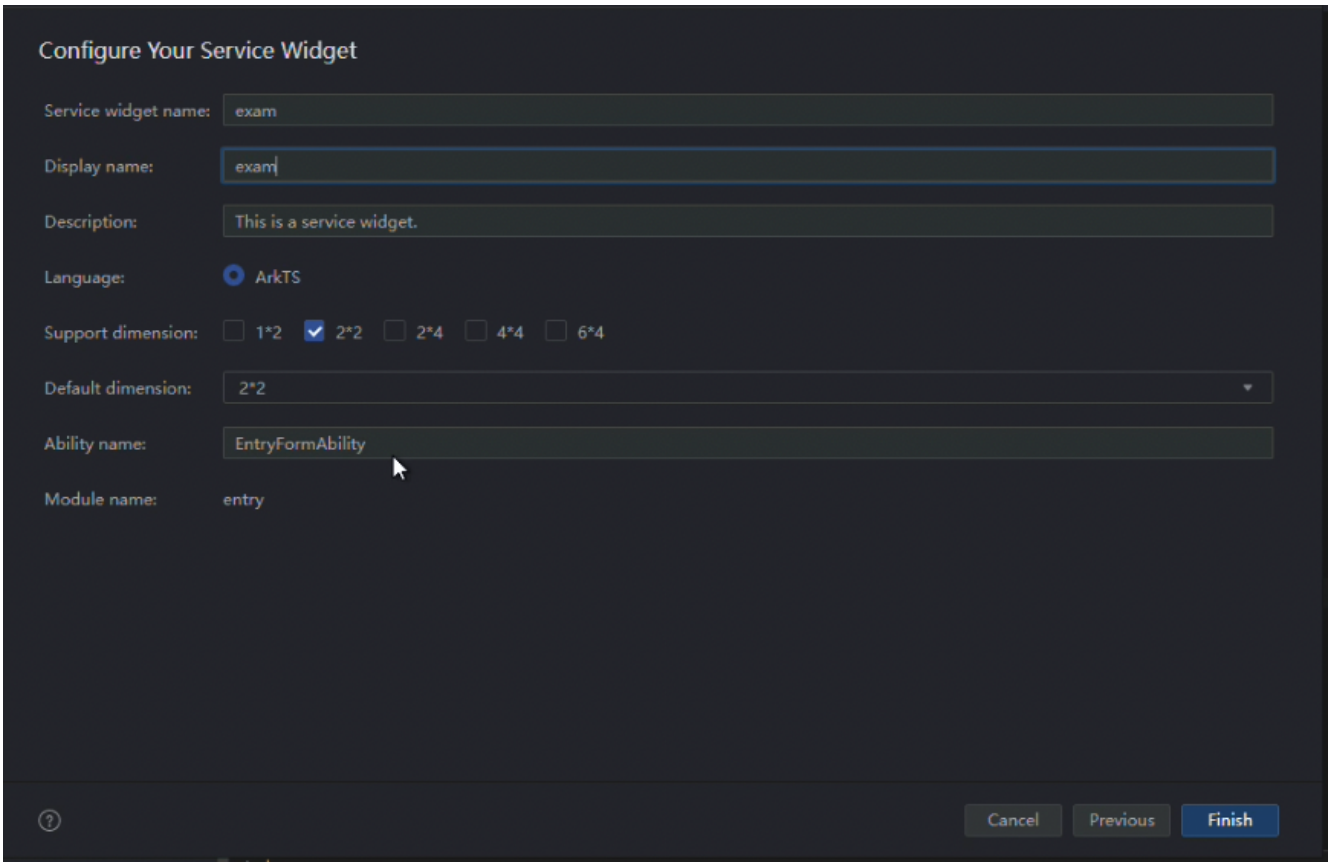
1. 使用开发工具DevEco Studio，创建一个名为CardExam的HarmonyOS应用工程。



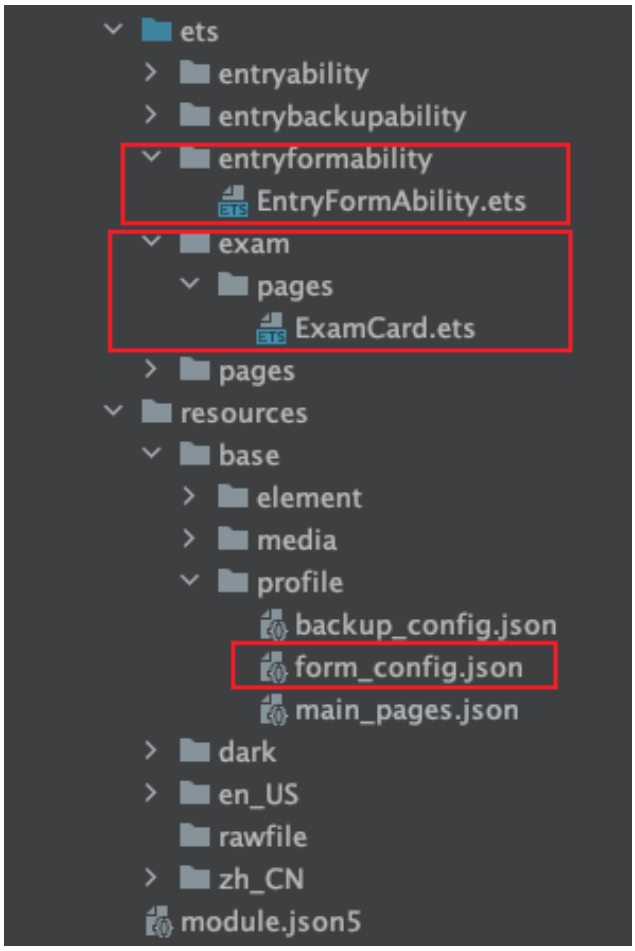
2. CardExam应用工程创建完毕后，为该应用创建一张ArkTS卡片。







3. 截图上传需要包含红框中标示的文件



任务2：使用router事件跳转到EntryAbility（22分）

得分点：

在EntryAbility.ts文件中补全onCreate()、onNewWant()、 onWindowStageCreate()三个生命周期回调函数中的代码，以实现使用router事件跳转到EntryAbility，并根据传递的参数拉起指定页面。

详细操作步骤：

1. 在ets/pages 目录下新建一张名为Detail 的Page页面，将下列代码复制到新建的Detail.ets文件中，覆盖Detail.ets文件中的原有代码：

```
// Detail.ets
@Entry
@Component
struct Detail {
  @State message: string = '这是详情页'

  build() {
    Row() {
      Column() {
        Text(this.message)
          .fontSize(50)
          .fontWeight(FontWeight.Bold)
      }
```

```
      .width('100%')
    }
    .height('100%')
  }
}
```

2. 将下列代码复制到ExamCard.ets文件中，覆盖ExamCard.ets文件中的原有代码：

```
let storage = new LocalStorage()

@Entry(storage)
@Component
struct ExamCard {
  //演示message刷新的文本
  @LocalStorageProp('refresh') refresh: string = 'message刷新前'
  readonly FULL_WIDTH_PERCENT: string = '100%';
  readonly FULL_HEIGHT_PERCENT: string = '100%';
  readonly IMAGE_HEIGHT_PERCENT: string = '80%';
  readonly TEXT_MARGIN_LEFT: number = 40;
  readonly BUTTON_MARGIN_LEFT: number = 30;
  readonly TEXT_MARGIN_BOTTOM: number = 10;
  readonly BUTTON_RADIUS: number = 8;
  readonly BUTTON_WIDTH: number = 90;
  readonly BUTTON_HEIGHT: number = 40;
  readonly BUTTON_OPACITY: number = 0.8;

  readonly ACTION_TYPE: string = 'router';
  readonly ABILITY_NAME: string = 'EntryAbility';
  readonly ACTION_TYPE2: string = 'message';

  build() {
    Row() {
      Column() {
        //演示message刷新
        Text(this.refresh)
          .fontSize(14)
          .fontColor(Color.Red)
          .fontWeight(FontWeight.Bold)
          .fontStyle(FontStyle.Italic)
          .margin({ left: this.TEXT_MARGIN_LEFT, bottom:
this.TEXT_MARGIN_BOTTOM })
        Blank()
        //刷新按钮，点击按钮刷新推荐文本内容
        Button('刷新', { type: ButtonType.Normal })
          .borderRadius(this.BUTTON_RADIUS)
          .opacity(this.BUTTON_OPACITY)
          .backgroundColor(0x4282ea)
          .width(this.BUTTON_WIDTH)
          .height(this.BUTTON_HEIGHT)
```

```
          .margin({ left: this.BUTTON_MARGIN_LEFT })
          .onClick(() => { //点击刷新按钮,通过message事件刷新卡片数据
            postCardAction(this, {
              'action': this.ACTION_TYPE2,
              'params': {
                'msgTest': 'messageEvent'
              }
            });
          })
      }
      .width(this.FULL_WIDTH_PERCENT)
      .height(this.FULL_HEIGHT_PERCENT)
      .alignItems(HorizontalAlign.Start)
    }
    .height(this.FULL_HEIGHT_PERCENT)
    .alignItems(VerticalAlign.Top)
    .padding(12)
    .backgroundImageSize({ width: this.FULL_WIDTH_PERCENT, height:
this.FULL_WIDTH_PERCENT })
    .onClick(() => { //使用router事件，拉起EntryAbility并跳转到对应的页面
      postCardAction(this, {
        "action": this.ACTION_TYPE,
        "abilityName": this.ABILITY_NAME,
        "params": {
          'targetPage': 'detail' // 在EntryAbility中处理这个信息
        }
      });
    })
  }
}
```

3. 将下列代码复制到EntryAbility.ts文件中，覆盖EntryAbility.ts文件中的原有代码，并根据注释补全缺失代码，实现在Entrybility中接收router事件并获取参数，根据传递的params不问，选择拉起不同的页面。

```
import { AbilityConstant, UIAbility, Want } from '@kit.AbilityKit';
import { hilog } from '@kit.PerformanceAnalysisKit';
import { window } from '@kit.ArkUI';

export default class EntryAbility extends UIAbility {
  private  selectPage:string = '';
  private currentWindowStage:window.WindowStage|null = null;
  // 如果UIAbility是首次启动，在收到卡片Router事件后会触发onCreate生命周期回调
  onCreate(want: Want, launchParam: AbilityConstant.LaunchParam): void {
    if (want?.parameters?.params ) {
      let params:Record<string,Object> = JSON.parse(want.parameters.params as
string);//补全代码，以实现获取获取router事件中传递的targetPage参数
      this.selectPage = params.targetPage as string;//补全代码，以实现获取获取
router事件中传递的targetPage参数
    }
```

```
  }
  // 如果UIAbility是非首次启动，在收到卡片Router事件后会触发onNewWant生命周期回调
  onNewWant(want: Want, launchParam: AbilityConstant.LaunchParam) {
    if (want?.parameters?.params ) {
      let params:Record<string,Object> = JSON.parse(want.parameters.params as string);//补全代码，以实现获取获取router事件中传递的targetPage参数
      this.selectPage = params.targetPage as string;//补全代码，以实现获取获取router事件中传递的targetPage参数
    }
    if (this.currentWindowStage !== null) {
      this.onWindowStageCreate(this.currentWindowStage);
    }
  }

  onDestroy(): void {
    hilog.info(0x0000, 'testTag', '%{public}s', 'Ability onDestroy');
  }

  onWindowStageCreate(windowStage: window.WindowStage) {
    let targetPage:string;
    // 根据传递的targetPage不同，选择拉起不同的页面
    switch (this.selectPage) {
      case 'detail':
        targetPage = 'pages/Detail';//补全代码，拉起详情页面
        break;
      default:
        targetPage = 'pages/Index';//补全代码，拉起应用主页
    }
    if (this.currentWindowStage === null) {
      this.currentWindowStage = windowStage;
    }
    windowStage.loadContent(targetPage, (err, data) => {//补全代码，执行具体的拉起页面操作
      if (err.code) {
        hilog.error(0x0000, 'testTag', 'Failed to load the content. Cause: %{public}s', JSON.stringify(err) ?? '');
        return;
      }
      hilog.info(0x0000, 'testTag', 'Succeeded in loading the content. Data: %{public}s', JSON.stringify(data) ?? '');
    });
  }

  onWindowStageDestroy(): void {
    // Main window is destroyed, release UI related resources
    hilog.info(0x0000, 'testTag', '%{public}s', 'Ability onWindowStageDestroy');
  }

```

```
  onForeground(): void {
    // Ability has brought to foreground
    hilog.info(0x0000, 'testTag', '%{public}s', 'Ability onForeground');
  }

  onBackground(): void {
    // Ability has back to background
    hilog.info(0x0000, 'testTag', '%{public}s', 'Ability onBackground');
  }
}
```

4. 补全代码后，对补全的代码进行截图，截图示例如下：

   这里就按照要求截取对应的代码上传即可，略过了

任务3：通过message事件刷新卡片内容（8分）

得分点：

在EntryFormAbility.ts文件中补全onFormEvent()生命周期回调函数中的代码。

详细操作步骤：

1. 将下列代码复制到EntryFormAbility.ts中，覆盖EntryFormAbility.ts原有的代码，并根据注释补全缺失代码，以实现通过message事件刷新卡片的内容。

```
import { formBindingData, FormExtensionAbility, formInfo, formProvider } from '@kit.FormKit';
import { Want } from '@kit.AbilityKit';
import { BusinessError } from '@kit.BasicServicesKit';

interface GeneratedObjectLiteralInterface_1 {
  'refresh': string;
}

export default class EntryFormAbility extends FormExtensionAbility {
  onAddForm(want: Want) {
    // Called to return a FormBindingData object.
    let formData = '';
    return formBindingData.createFormBindingData(formData);
  }

  onCastToNormalForm(formId: string) {
    // Called when the form provider is notified that a temporary form is successfully
    // converted to a normal form.
  }

  onUpdateForm(formId: string) {
    // Called to notify the form provider to update a specified form.
```

```
  }

  onFormEvent(formId:string, message:string) {
    class FormDataClass {
      'refresh':string='message刷新后' //要刷新的具体内容，和卡片布局中对应
    }
    let formData = new FormDataClass();
    //补全代码，绑定要刷新的内容
    let formInfo = formBindingData.createFormBindingData(formData)
    //补全代码，调用相关接口刷新卡片内容
    formProvider.updateForm(formId, formInfo).then((data) => {
      console.info('FormAbility updateForm success.' + JSON.stringify(data));
    }).catch((error:BusinessError) => {
      console.error('FormAbility updateForm failed: ' + JSON.stringify(error));
    })
  }

  onRemoveForm(formId: string) {
    // Called to notify the form provider that a specified form has been
destroyed.
  }

  onAcquireFormState(want: Want) {
    // Called to return a {@link FormState} object.
    return formInfo.FormState.READY;
  }
};
```