**Приложение 3**

**МОБИЛЬНОЕ ПРИЛОЖЕНИЕ ДЛЯ ПРОСМОТРА РАСПИСАНИЯ МИЭТ**

**ТЕКСТ ПРОГРАММЫ**

**МП ПР**

Содержание

[Пользовательский интерфейс 3](#_Toc135400971)

[Окно авторизации – класс AuthScreen 3](#_Toc135400972)

[Окно расписания – класс ScheduleScreen 4](#_Toc135400973)

[Окно поиска – класс FindScreen 5](#_Toc135400974)

[Окно задач – класс TaskScreen 8](#_Toc135400975)

[Окно просмотра задачи – класс TaskInfoScreen 8](#_Toc135400976)

[Компоненты – класс InterfaceElements 9](#_Toc135400977)

[Контроллер 11](#_Toc135400978)

[Класс MainActivity 11](#_Toc135400979)

[Класс Screen 12](#_Toc135400980)

[Класс NavGraph 12](#_Toc135400981)

[Модель 13](#_Toc135400982)

[Класс Task 13](#_Toc135400983)

[Класс JSONFile 14](#_Toc135400984)

[Класс Group 14](#_Toc135400985)

[Класс Schedule 14](#_Toc135400986)

[Класс Week 15](#_Toc135400987)

[Класс Day 15](#_Toc135400988)

[Класс Lesson 16](#_Toc135400989)

**Пользовательский интерфейс**

**Окно авторизации – класс AuthScreen**

package com.n7art.rmiet  
  
import android.content.Context  
import android.widget.Toast  
import androidx.compose.runtime.\*  
import androidx.navigation.NavController  
import androidx.lifecycle.LifecycleOwner  
import androidx.lifecycle.*lifecycleScope*import com.n7art.rmiet.Controller.Screen  
import kotlinx.coroutines.CoroutineScope  
import kotlinx.coroutines.Dispatchers  
import kotlinx.coroutines.launch  
import okhttp3.\*  
import org.jsoup.Jsoup  
import java.io.IOException  
  
@Composable  
fun AuthScreen(navController: NavController) {  
 *Auth*(navController)  
}  
  
fun onLoginClick(  
 username: String,  
 password: String,  
 navController: NavController,  
 context: Context  
) {  
 val client = OkHttpClient()  
 var csrfToken = ""  
 val grequest = Request.Builder()  
 .url("https://orioks.miet.ru/user/login")  
 .get()  
 .build()  
 client.newCall(grequest).enqueue(object : Callback {  
 override fun onFailure(call: Call, e: IOException) {  
 e.printStackTrace()  
 }  
  
 override fun onResponse(call: Call, response: Response) {  
 val responseBody = response.body?.string()  
 response.body?.close()  
 if (response.isSuccessful) {  
 val doc = Jsoup.parse(responseBody)  
 csrfToken = doc.select("input[name=\_csrf]").attr("value")  
 }  
 }  
 })  
  
 val formBody = FormBody.Builder()  
 .add("\_csrf", csrfToken)  
 .add("LoginForm%5Blogin%5D", username)  
 .add("LoginForm%5Bpassword%5D", password)  
 .build()  
 val request = Request.Builder()  
 .url("https://orioks.miet.ru/user/login")  
 .post(formBody)  
 .build()  
 client.newCall(request).enqueue(object : Callback {  
 override fun onFailure(call: Call, e: IOException) {  
 e.printStackTrace()  
 }  
  
 override fun onResponse(call: Call, response: Response) {  
 if (response.isSuccessful) {  
 (context as LifecycleOwner).*lifecycleScope*.*launch*(Dispatchers.Main) **{** navController.navigate(Screen.Schedule.route) **{** popUpTo(Screen.Schedule.route) **{** inclusive = true  
 **}  
 }  
 }** } else {  
 *CoroutineScope*(Dispatchers.Main).*launch* **{** Toast.makeText(context, "Неправильный логин или пароль", Toast.*LENGTH\_LONG*)  
 .show()  
 **}** }  
 }  
 })  
}

**Окно расписания – класс ScheduleScreen**

package com.n7art.rmiet  
  
import android.os.Build  
import androidx.compose.foundation.layout.\*  
import androidx.compose.runtime.\*  
import androidx.compose.ui.Alignment  
import androidx.compose.ui.Modifier  
import androidx.navigation.NavController  
import java.time.LocalDate  
  
@Composable  
fun ScheduleScreen(navController: NavController) {  
 if (Build.VERSION.*SDK\_INT* >= Build.VERSION\_CODES.*O*) {  
 var selectedDate by remember **{** *mutableStateOf*(LocalDate.now().*dayOfMonth*) **}** Box(contentAlignment = Alignment.BottomCenter) **{** Column(  
 modifier = Modifier  
 .*fillMaxSize*(),  
 verticalArrangement = Arrangement.Top  
 ) **{** Header("Артамонова Анастасия ", "ПИН-44")  
 *WeekCalendar*(  
 header = "Знаменатель 1",  
 onDaySelected = **{** dayIndex **->** selectedDate = dayIndex  
 **}** )  
 Buttons(1f, 0.7f, navController)  
 **}** FindButton(navController)  
 **}** }  
}

**Окно поиска – класс FindScreen**

package com.n7art.rmiet  
  
import android.util.Log  
import androidx.compose.foundation.background  
import androidx.compose.foundation.clickable  
import androidx.compose.foundation.layout.\*  
import androidx.compose.foundation.shape.*CircleShape*import androidx.compose.foundation.text.KeyboardActions  
import androidx.compose.foundation.text.KeyboardOptions  
import androidx.compose.material.\*  
import androidx.compose.runtime.\*  
import androidx.compose.ui.Alignment  
import androidx.compose.ui.Modifier  
import androidx.compose.ui.draw.alpha  
import androidx.compose.ui.focus.onFocusChanged  
import androidx.compose.ui.graphics.Color  
import androidx.compose.ui.res.colorResource  
import androidx.compose.ui.text.TextStyle  
import androidx.compose.ui.text.font.Font  
import androidx.compose.ui.text.font.FontFamily  
import androidx.compose.ui.text.font.FontWeight  
import androidx.compose.ui.text.input.ImeAction  
import androidx.compose.ui.text.input.KeyboardCapitalization  
import androidx.compose.ui.unit.dp  
import androidx.compose.ui.unit.sp  
import androidx.navigation.NavController  
import com.n7art.rmiet.Model.\*  
  
private val *centurygothic* = *FontFamily*(  
 *Font*(R.font.*century\_gothic\_regular*), *Font*(R.font.*century\_gothic\_bold*, FontWeight.Bold)  
)  
  
@Composable  
fun FindScreen(navController: NavController) {  
 var find by remember **{** *mutableStateOf*("") **}** var alphag by *remember* **{** *mutableStateOf*(1f) **}** var alphat by remember **{** *mutableStateOf*(0.7f) **}** var alphaw by remember **{** *mutableStateOf*(0.7f) **}** val groups = *createFile*()  
 val groupsList = *getGroupList*(groups)  
 val teachersList = *getTeacherList*(groups)  
 var findList by remember **{** *mutableStateOf*(groupsList) **}** var showDropdown by remember **{** *mutableStateOf*(false) **}** val isFocusedFind = remember **{** *mutableStateOf*(false) **}** *Box*(contentAlignment = Alignment.BottomCenter) **{** Column(  
 modifier = Modifier  
 .*fillMaxSize*()  
 .*background*(colorResource(R.color.*orioks*)),  
 horizontalAlignment = Alignment.CenterHorizontally  
 ) **{** OutlinedTextField(value = find,  
  
 onValueChange = **{** newText **->** find = newText.*replaceFirstChar* **{** if (**it**.*isLowerCase*()) **it**.*titlecase*() else **it**.toString()  
 **}  
 }**,  
 keyboardOptions = KeyboardOptions(  
 imeAction = ImeAction.Go, capitalization = KeyboardCapitalization.Words  
 ),  
 keyboardActions = KeyboardActions(onGo = **{}**),  
 textStyle = TextStyle(fontSize = 16.*sp*, fontFamily = *centurygothic*),  
 colors = TextFieldDefaults.outlinedTextFieldColors(  
 focusedBorderColor = Color.White, *// цвет при получении фокуса* unfocusedBorderColor = Color.White, *// цвет при отсутствии фокуса* backgroundColor = Color.White,  
 textColor = Color.Black,  
 cursorColor = Color.Black,  
 ),  
 shape = *CircleShape*,*//RoundedCornerShape(12.dp),* modifier = Modifier  
 .*padding*(10.*dp*, 15.*dp*, 10.*dp*, 10.*dp*)  
 .*fillMaxWidth*()  
 .*fillMaxHeight*(0.07f)  
 .*onFocusChanged* **{** isFocusedFind.value = **it**.isFocused  
 Log.i("CLICK", **it**.isFocused.toString())  
 showDropdown = **it**.isFocused  
 **}**)  
 Row(  
 modifier = Modifier.*fillMaxWidth*(), horizontalArrangement = Arrangement.SpaceAround  
 ) **{** *Text*("группа",  
 fontSize = 16.*sp*,  
 fontWeight = FontWeight.Normal,  
 fontFamily = *centurygothic*,  
 color = *colorResource*(R.color.*white*),  
 modifier = Modifier  
 .*alpha*(alphag)  
 .*clickable* **{** alphag = 1f  
 alphat = 0.7f  
 alphaw = 0.7f  
 findList = groupsList  
 **}**)  
 Text("преподаватель",  
 fontSize = 16.*sp*,  
 fontWeight = FontWeight.Normal,  
 fontFamily = *centurygothic*,  
 color = colorResource(R.color.*white*),  
 modifier = Modifier  
 .*alpha*(alphat)  
 .*clickable* **{** alphag = 0.7f  
 alphat = 1f  
 alphaw = 0.7f  
 findList = teachersList  
 **}**)  
 *Text*("окно",  
 fontSize = 16.*sp*,  
 fontWeight = FontWeight.Normal,  
 fontFamily = *centurygothic*,  
 color = *colorResource*(R.color.*white*),  
 modifier = Modifier  
 .*alpha*(alphaw)  
 .*clickable* **{** alphag = 0.7f  
 alphat = 0.7f  
 alphaw = 1f  
 **}**)  
 **}  
 }** *Box*(  
 modifier = Modifier.*fillMaxSize*(), contentAlignment = Alignment.Center  
 ) **{** DropdownMenu(modifier = Modifier  
 .*fillMaxWidth*(0.8f)  
 .*fillMaxHeight*(0.5f),  
 expanded = showDropdown,  
 onDismissRequest = **{** showDropdown = true **}**) **{** findList.*forEach* **{** option **->** *DropdownMenuItem*(onClick = **{** find = option  
 showDropdown = false  
 **}**) **{** Text(option)  
 **}  
 }  
 }  
 }** FindButtonToFind()  
 **}**}  
  
fun getGroupList(groups: List<Group>): List<String> {  
 return groups.*map* **{ it**.name **}**}  
  
fun getTeacherList(groups: List<Group>): List<String> {  
 val teachersList: MutableList<String> = *mutableListOf*()  
 for (group: Group in groups) for (week: Week in group.schedule.WeekToList()) for (day: Day in week.DayToList()) for (lesson: Lesson in day.LessonToList()) if (lesson != null && !lesson.teacher.*contains*(  
 "Преподаватель"  
 ) && !teachersList.contains(  
 lesson.teacher  
 )  
 ) teachersList.add(lesson.teacher)  
 return teachersList  
}

**Окно задач – класс TaskScreen**

package com.n7art.rmiet  
  
import androidx.compose.foundation.layout.\*  
import androidx.compose.runtime.Composable  
import androidx.compose.ui.Alignment  
import androidx.compose.ui.Modifier  
import androidx.navigation.NavController  
import com.n7art.rmiet.Model.Task  
  
@Composable  
fun TaskScreen(navController: NavController, tasks: List<Task>, user: String, group: String) {  
 Box(contentAlignment = Alignment.BottomCenter) **{** *Column*(  
 modifier = Modifier  
 .*fillMaxSize*(),  
 verticalArrangement = Arrangement.Top  
 ) **{** Header(user, group)  
 AddNewTask()  
 Tasks(tasks, navController)  
 *Buttons*(0.7f, 1f, navController)  
 **}** FindButton(navController)  
 **}**}

**Окно просмотра задачи – класс TaskInfoScreen**

package com.n7art.rmiet  
  
import androidx.compose.foundation.layout.Arrangement  
import androidx.compose.foundation.layout.Column  
import androidx.compose.foundation.layout.fillMaxSize  
import androidx.compose.runtime.Composable  
import androidx.compose.ui.Modifier  
import androidx.navigation.NavController  
import com.n7art.rmiet.Model.Task  
  
@Composable  
fun TaskInfoScreen(navController: NavController, task: Task?) {  
 Column(  
 modifier = Modifier  
 .*fillMaxSize*(),  
 verticalArrangement = Arrangement.Top  
 ) **{** HeaderBack(navController)  
 task?.*let* **{** TaskInfo(**it**) **}  
 }**}

**Компоненты – класс InterfaceElements**

package com.n7art.rmiet  
  
import android.os.Build  
import android.util.Log  
import androidx.compose.foundation.BorderStroke  
import androidx.compose.foundation.Image  
import androidx.compose.foundation.background  
import androidx.compose.foundation.clickable   
import androidx.compose.foundation.lazy.itemsIndexed  
import androidx.compose.foundation.shape.*CircleShape*import androidx.compose.foundation.shape.RoundedCornerShape   
import androidx.compose.ui.Alignment  
import androidx.compose.ui.Modifier  
import androidx.compose.ui.composed   
import androidx.compose.ui.geometry.Offset  
import androidx.compose.ui.graphics.Color  
import androidx.compose.ui.graphics.*RectangleShape*   
import androidx.compose.ui.text.TextStyle  
import androidx.compose.ui.text.font.FontFamily  
import androidx.compose.ui.text.font.FontWeight  
import androidx.compose.ui.text.font.Font   
  
  
private val *centurygothic* = *FontFamily*(  
 *Font*(R.font.*century\_gothic\_regular*),  
 *Font*(R.font.*century\_gothic\_bold*, FontWeight.Bold)  
)  
  
@Composable  
fun Header(name: String, group: String) {  
 *//вычисление инициалов* val words = name.*split*(" ")  
 val initials = words[0][0] + words[1].*substring*(0, 1)  
 *//две колонки:имя,фамилия,группа; аватар  
 Card*(  
 modifier = Modifier  
 .*fillMaxWidth*()  
 .*fillMaxHeight*(0.09f),  
 shape = *RectangleShape*,  
 elevation = 10.*dp* ) **{** *Row*(  
 modifier = Modifier  
 .*background*(colorResource(R.color.*orioks*)),  
 horizontalArrangement = Arrangement.SpaceBetween,  
 verticalAlignment = Alignment.CenterVertically,  
 ) **{** *//две строки:имя,фамилия; группа* Column(  
 modifier = Modifier  
 .*padding*(10.*dp*)  
 .*weight*(0.85f)  
  
 ) **{** Text(  
 name,  
 fontSize = 16.*sp*,  
 fontWeight = FontWeight.Normal,  
 fontFamily = *centurygothic*,  
 color = Color.White  
 )  
 *Text*(  
 group,  
 fontSize = 16.*sp*,  
 fontWeight = FontWeight.Normal,  
 fontFamily = *centurygothic*,  
 color = Color.White,  
 modifier = Modifier.*alpha*(0.9f)  
 )  
 **}** *//для наслоения квадрата на круг* Box(  
 modifier = Modifier  
 .*padding*(2.*dp*)  
 .*fillMaxSize*()  
 .*weight*(0.17f),  
 contentAlignment = Alignment.TopEnd  
  
 ) **{** *//квадрат в правом верхнем углу  
 Column*(  
 modifier = Modifier  
 .*fillMaxHeight*(0.5f)  
 .*aspectRatio*(1f, true)  
 .*background*(Color.White),  
 verticalArrangement = Arrangement.Top  
  
 ) **{}** *//Круг(box) с инициалами* Box(  
 modifier = Modifier  
 .*background*(Color.White, *CircleShape*)  
 .*fillMaxHeight*()  
 .*aspectRatio*(1f, true)  
 ) **{** Text(  
 initials,  
 Modifier.*align*(Alignment.Center),  
 fontSize = 26.*sp*,  
 fontWeight = FontWeight.ExtraBold,  
 fontFamily = *centurygothic* )  
 **}  
 }  
 }  
 }**}  
  
@Composable  
fun Buttons(alpha1: Float, alpha2: Float, navController: NavController) {  
 *Card*(  
 modifier = Modifier  
 .*fillMaxWidth*()  
 .*fillMaxHeight*(1f),  
 shape = *RectangleShape*,  
 elevation = 10.*dp* ) **{** Row(  
 modifier = Modifier  
 .*fillMaxSize*(),  
 horizontalArrangement = Arrangement.SpaceBetween  
 ) **{** ScheduleButton(navController, alpha1)  
 TaskButton(navController, alpha2)  
 **}  
 }**}  
fun Modifier.bottomBorder(strokeWidth: Dp, color: Color, alpha: Float) = *composed*(  
 factory = **{** val density = *LocalDensity*.current  
 val strokeWidthPx = density.*run* **{** strokeWidth.*toPx*() **}** Modifier.*drawBehind* **{** val width = size.width  
 val height = size.height - strokeWidthPx / 2  
  
 drawLine(  
 color = color,  
 start = *Offset*(x = 0f, y = height),  
 end = *Offset*(x = width, y = height),  
 strokeWidth = strokeWidthPx,  
 alpha = alpha  
 )  
 **}  
 }**)

**Контроллер**

**Класс MainActivity**

package com.n7art.rmiet.Controller  
  
import android.os.Bundle  
import androidx.activity.ComponentActivity  
import androidx.activity.compose.setContent  
import androidx.navigation.NavHostController  
import androidx.navigation.compose.rememberNavController  
import com.n7art.rmiet.SetupNavGraph  
  
class MainActivity : ComponentActivity() {  
 lateinit var navController: NavHostController  
 override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 *setContent* **{** navController = *rememberNavController*()  
 SetupNavGraph(navController)  
 **}** }  
}

**Класс Screen**

package com.n7art.rmiet.Controller  
  
sealed class Screen(val route: String){  
 object Schedule: Screen(route = "schedule")  
 object Task: Screen(route = "task")  
 object TaskInfo: Screen(route = "taskinfo/{task}")  
 object Auth: Screen(route = "auth")  
 object Find: Screen(route = "find")  
}

**Класс NavGraph**

package com.n7art.rmiet  
  
import androidx.compose.runtime.Composable  
import androidx.navigation.NavHostController  
import androidx.navigation.NavType  
import androidx.navigation.compose.NavHost  
import androidx.navigation.compose.composable  
import androidx.navigation.navArgument  
import com.n7art.rmiet.Controller.Screen  
import com.n7art.rmiet.Model.Task  
  
@Composable  
fun SetupNavGraph(navController: NavHostController) {  
 *NavHost*(  
 navController = navController,  
 startDestination = Screen.Auth.route  
 ) **{** *composable*(  
 route = Screen.Auth.route  
 ) **{** *AuthScreen*(navController)  
 **}** *composable*(  
 route = Screen.Schedule.route  
 ) **{** ScheduleScreen(navController)  
 **}** *composable*(  
 route = Screen.Task.route  
 ) **{** TaskScreen(navController)  
 **}** *composable*(  
 route = Screen.TaskInfo.route,  
 arguments = *listOf*(  
 *navArgument*("task") **{** type = NavType.ParcelableType(Task::class.*java*)  
 **}** )  
 ) **{** val task = **it**.arguments?.getParcelable<Task>("task")  
 TaskInfoScreen(navController, task)  
 **}** *composable*(  
 route = Screen.Find.route  
 ) **{** *FindScreen*(navController)  
 **}  
 }**}

**Модель**

**Класс Task**

package com.n7art.rmiet.Model  
  
import android.os.Parcel  
import android.os.Parcelable  
import java.util.Calendar  
  
class Task(\_name: String, \_text: String, \_subjects: String, \_date: Calendar) : Parcelable {  
 var name: String = ""  
 var text: String = ""  
 var subject: String = ""  
 var date: Calendar = Calendar.getInstance()  
  
 init {  
 name = \_name  
 text = \_text  
 subject = \_subjects  
 date = \_date  
 }  
  
 constructor() : this("", "", "", Calendar.getInstance())  
 constructor(parcel: Parcel) : this(  
 parcel.readString()!!,  
 parcel.readString()!!,  
 parcel.readString()!!,  
 parcel.readSerializable() as Calendar  
 )  
  
 override fun writeToParcel(parcel: Parcel, flags: Int) {  
 parcel.writeString(name)  
 parcel.writeString(text)  
 parcel.writeString(subject)  
 parcel.writeSerializable(date)  
 }  
  
 override fun describeContents(): Int {  
 return 0  
 }  
  
 companion object CREATOR : Parcelable.Creator<Task> {  
 override fun createFromParcel(parcel: Parcel): Task {  
 return Task(parcel)  
 }  
  
 override fun newArray(size: Int): Array<Task?> {  
 return *arrayOfNulls*(size)  
 }  
 }  
}

**Класс JSONFile**

package com.n7art.rmiet.Model  
  
import androidx.compose.runtime.Composable  
import androidx.compose.ui.platform.*LocalContext*import com.google.gson.GsonBuilder  
import com.n7art.rmiet.R  
import java.io.\*  
  
@Composable  
fun createFile(): List<Group> {  
 val inputStream: InputStream = *LocalContext*.current.*resources*.openRawResource(R.raw.*schedule*)  
  
 val reader = BufferedReader(InputStreamReader(inputStream))  
 val builder = GsonBuilder()  
 val gson = builder.create()  
 val groups: List<Group> = gson.fromJson(reader, Array<Group>::class.*java*).*toList*()  
  
 try {  
 inputStream.close()  
 } catch (e: IOException) {  
 e.printStackTrace()  
 }  
 return groups  
}

**Класс Group**

package com.n7art.rmiet.Model  
  
class Group(\_name: String, \_schedule: Schedule) {  
 var name: String  
 var schedule: Schedule  
  
 init {  
 name = \_name;  
 schedule = \_schedule;  
 }  
}

**Класс Schedule**

package com.n7art.rmiet.Model  
  
class Schedule(\_ch1: Week, \_z1: Week, \_ch2: Week, \_z2: Week) {  
 var ch1: Week  
 var z1: Week  
 var ch2: Week  
 var z2: Week  
  
 init {  
 this.ch1 = \_ch1  
 this.z1 = \_z1  
 this.ch2 = \_ch2  
 this.z2 = \_z2  
 }  
  
 fun WeekToList(): List<Week> {  
 return *listOf*(ch1, z1, ch2, z2)  
 }  
}

**Класс Week**

package com.n7art.rmiet.Model  
  
class Week(\_mon: Day, \_tue: Day, \_wed: Day, \_thu: Day, \_fri: Day, \_sat: Day) {  
 var mon: Day  
 var tue: Day  
 var wed: Day  
 var thu: Day  
 var fri: Day  
 var sat: Day  
  
 init {  
 mon = \_mon  
 tue = \_tue  
 wed = \_wed  
 thu = \_thu  
 fri = \_fri  
 sat = \_sat  
 }  
  
 fun DayToList(): List<Day> {  
 return *listOf*(mon, tue, wed, thu, fri, sat)  
 }  
}

**Класс Day**

package com.n7art.rmiet.Model  
  
class Day(\_lesson1: Lesson, \_lesson2: Lesson, \_lesson3: Lesson, \_lesson4: Lesson, \_lesson5: Lesson, \_lesson6: Lesson, \_lesson7: Lesson) {  
 var lesson1: Lesson  
 var lesson2: Lesson  
 var lesson3: Lesson  
 var lesson4: Lesson  
 var lesson5: Lesson  
 var lesson6: Lesson  
 var lesson7: Lesson  
  
 init {  
 lesson1 = \_lesson1  
 lesson2 = \_lesson2  
 lesson3 = \_lesson3  
 lesson4 = \_lesson4  
 lesson5 = \_lesson5  
 lesson6 = \_lesson6  
 lesson7 = \_lesson7  
 }  
  
 fun LessonToList(): List<Lesson> {  
 return *listOf*(lesson1, lesson2, lesson3, lesson4, lesson5, lesson6, lesson7)  
 }  
}

**Класс Lesson**

package com.n7art.rmiet.Model  
  
import android.util.Log  
import androidx.compose.foundation.layout.\*  
import androidx.compose.material.Text  
import androidx.compose.runtime.Composable  
import androidx.compose.ui.Modifier  
  
class Lesson(\_name: String, \_teacher: String, \_classroom: String) {  
 var name: String  
 var teacher: String  
 var classroom: String  
  
 init {  
 name = \_name  
 teacher = \_teacher  
 classroom = \_classroom  
 }  
  
 @Composable  
 fun Output() {  
 Column(  
 modifier = Modifier  
 .*fillMaxSize*(),  
 verticalArrangement = Arrangement.Top,  
 ) **{** Text(name)  
 *Text*(teacher)  
 *Text*(classroom)  
  
 **}**}  
}