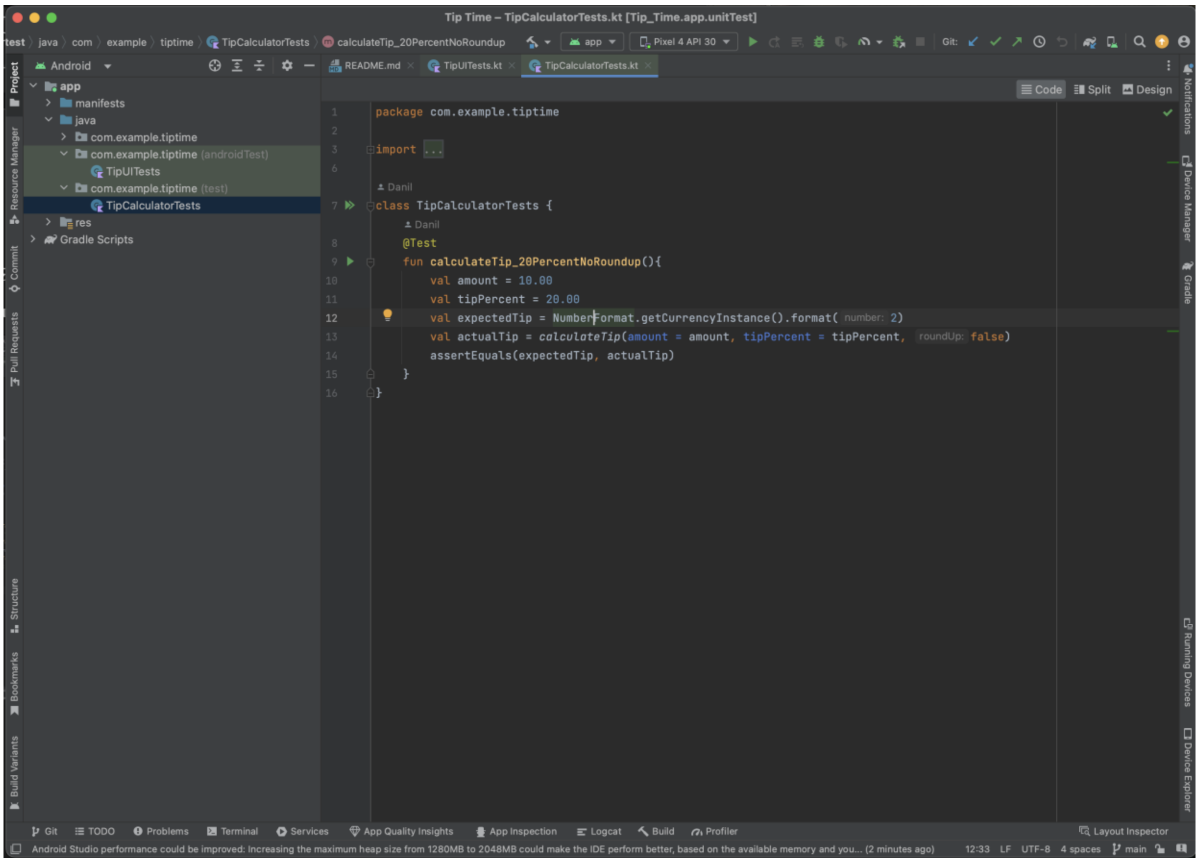
Корначенко Данил

Группа 9ИС-390К

Практическая работа №3

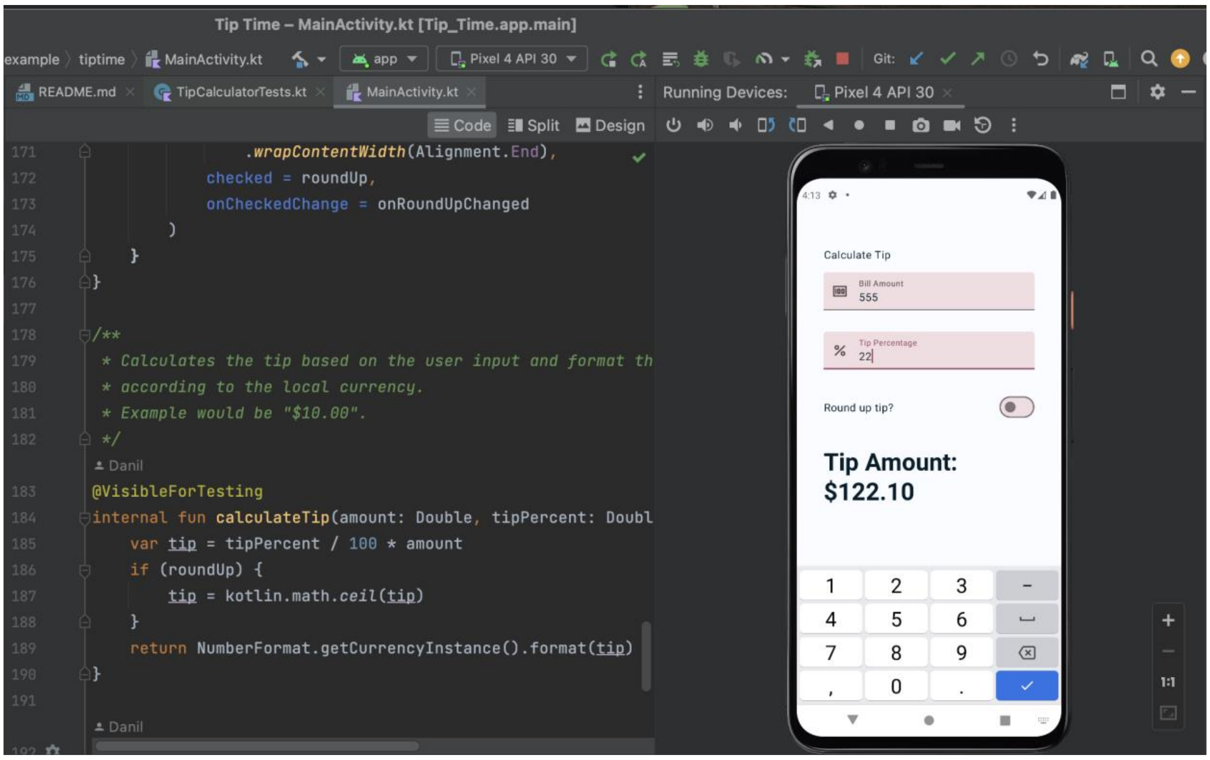
Юнит-тесты



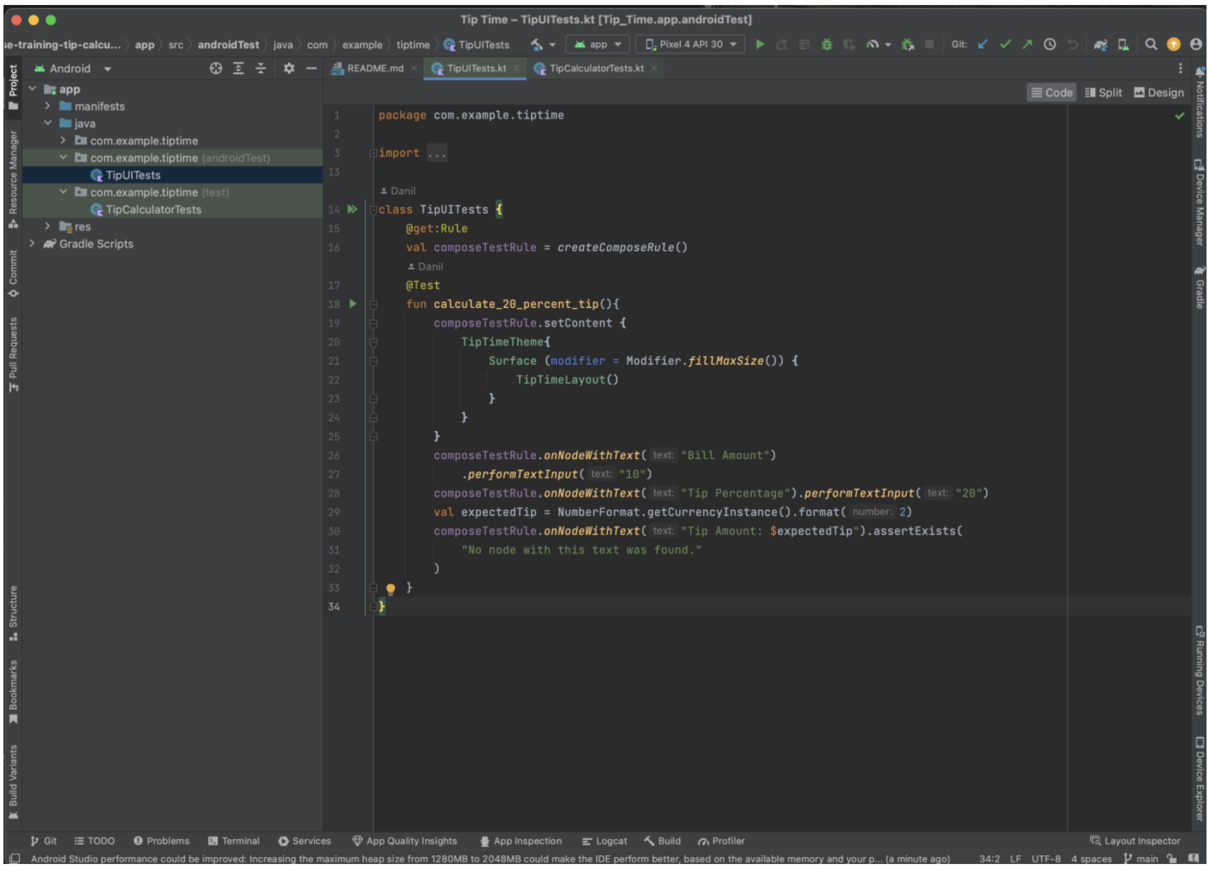
Tip Time – TipCalculatorTests.kt [Tip_Time.app.unitTest]

```
package com.example.tiptime

import ...

class TipCalculatorTests {
    @Test
    fun calculateTip_20PercentNoRoundup(){
        val amount = 10.00
        val tipPercent = 20.00
        val expectedTip = NumberFormat.getCurrencyInstance().format( number: 2)
        val actualTip = calculateTip(amount = amount, tipPercent = tipPercent, roundUp: false)
        assertEquals(expectedTip, actualTip)
    }
}
```

Android Studio performance could be improved: Increasing the maximum heap size from 1280MB to 2048MB could make the IDE perform better, based on the available memory and you... (2 minutes ago)

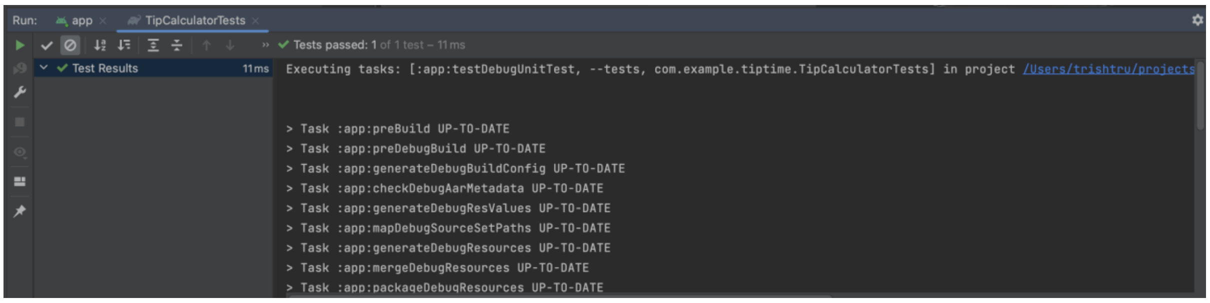

Tip Time – MainActivity.kt [Tip_Time.app.main]

```
                .wrapContentWidth(Alignment.End),
            checked = roundUp,
            onCheckedChange = onRoundUpChanged
        )
    }
}

/**
 * Calculates the tip based on the user input and format th
 * according to the local currency.
 * Example would be "$10.00".
 */
Danil
@VisibleForTesting
internal fun calculateTip(amount: Double, tipPercent: Doubl
    var tip = tipPercent / 100 * amount
    if (roundUp) {
        tip = kotlin.math.ceil(tip)
    }
    return NumberFormat.getCurrencyInstance().format(tip)
}

Danil
```


Tip Time – TipUITests.kt [Tip_Time.app.androidTest]

```
package com.example.tiptime

import ...

Danil
class TipUITests {
    @get:Rule
    val composeTestRule = createComposeRule()
    Danil
    @Test
    fun calculate_20_percent_tip(){
        composeTestRule.setContent {
            TipTimeTheme{
                Surface (modifier = Modifier.fillMaxSize()) {
                    TipTimeLayout()
                }
            }
        }
        composeTestRule.onNodeWithText( text: "Bill Amount")
            .performTextInput( text: "10")
        composeTestRule.onNodeWithText( text: "Tip Percentage").performTextInput( text: "20")
        val expectedTip = NumberFormat.getCurrencyInstance().format( number: 2)
        composeTestRule.onNodeWithText( text: "Tip Amount: $expectedTip").assertExists(
            "No node with this text was found."
        )
    }
}
```

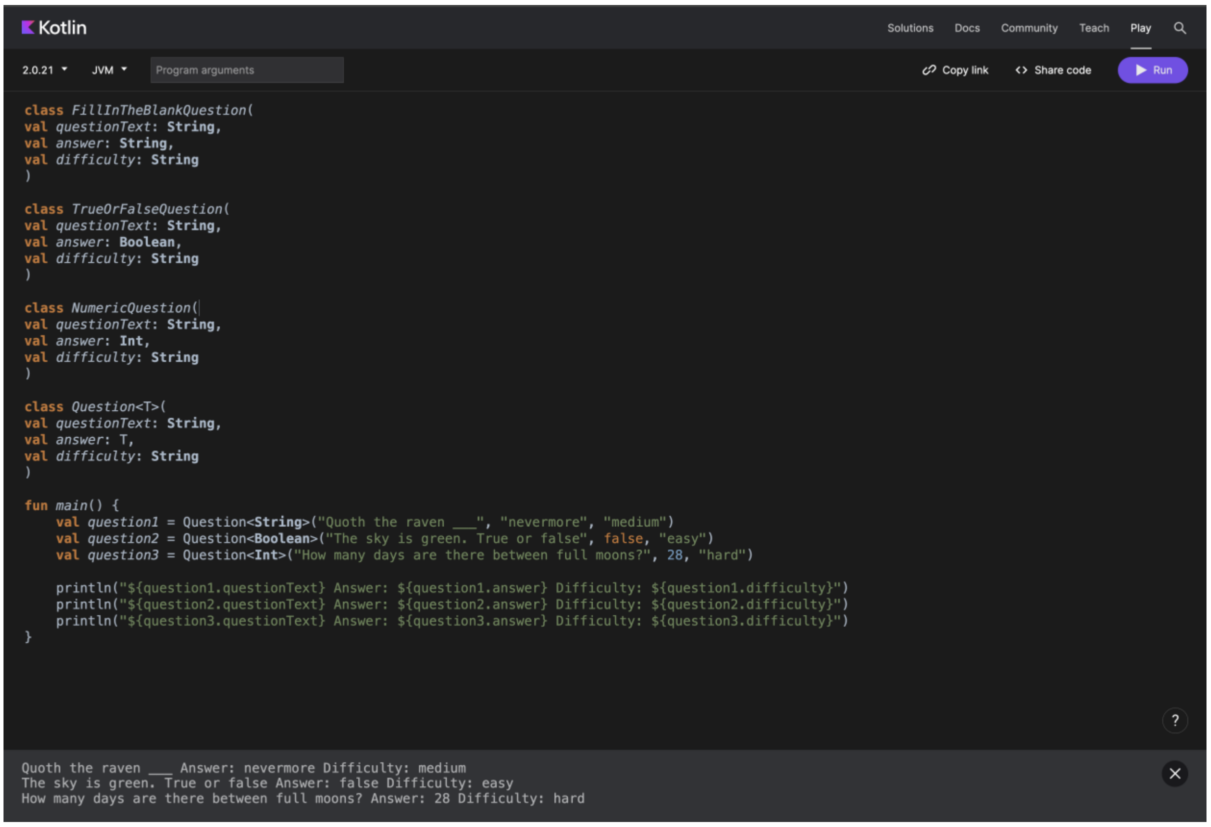
Android Studio performance could be improved: Increasing the maximum heap size from 1280MB to 2048MB could make the IDE perform better, based on the available memory and your p... (a minute ago)


Run: app TipCalculatorTests
Tests passed: 1 of 1 test – 11 ms
Test Results 11ms

```
Executing tasks: [:app:testDebugUnitTest, --tests, com.example.tiptime.TipCalculatorTests] in project /Users/trishtru/projects


> Task :app:preBuild UP-TO-DATE
> Task :app:preDebugBuild UP-TO-DATE
> Task :app:generateDebugBuildConfig UP-TO-DATE
> Task :app:checkDebugAarMetadata UP-TO-DATE
> Task :app:generateDebugResValues UP-TO-DATE
> Task :app:mapDebugSourceSetPaths UP-TO-DATE
> Task :app:generateDebugResources UP-TO-DATE
> Task :app:mergeDebugResources UP-TO-DATE
> Task :app:packageDebugResources UP-TO-DATE
```

Практическая работа №15

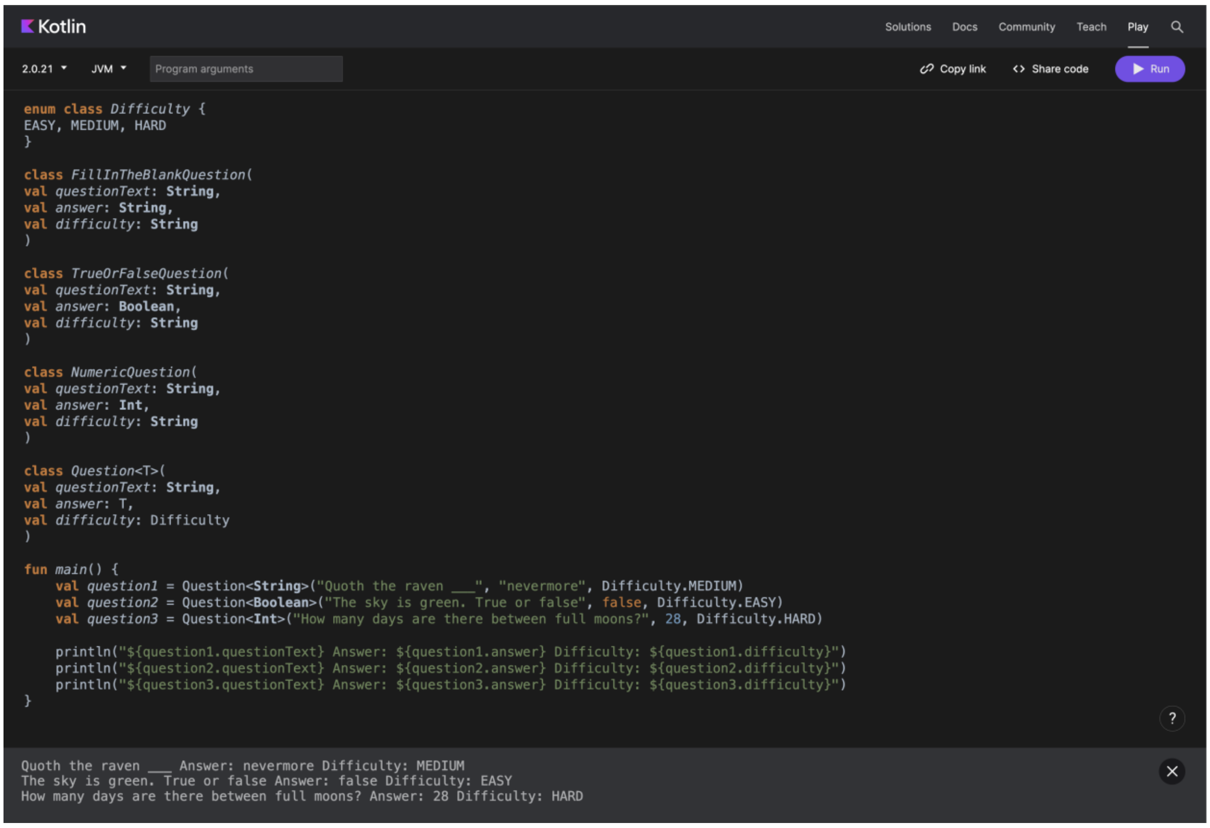
Задание 1.



```kotlin
class FillInTheBlankQuestion(
val questionText: String,
val answer: String,
val difficulty: String
)

class TrueOrFalseQuestion(
val questionText: String,
val answer: Boolean,
val difficulty: String
)

class NumericQuestion(
val questionText: String,
val answer: Int,
val difficulty: String
)

class Question<T>(
val questionText: String,
val answer: T,
val difficulty: String
)

fun main() {
    val question1 = Question<String>("Quoth the raven ___", "nevermore", "medium")
    val question2 = Question<Boolean>("The sky is green. True or false", false, "easy")
    val question3 = Question<Int>("How many days are there between full moons?", 28, "hard")

    println("${question1.questionText} Answer: ${question1.answer} Difficulty: ${question1.difficulty}")
    println("${question2.questionText} Answer: ${question2.answer} Difficulty: ${question2.difficulty}")
    println("${question3.questionText} Answer: ${question3.answer} Difficulty: ${question3.difficulty}")
}
```

```
Quoth the raven ___ Answer: nevermore Difficulty: medium
The sky is green. True or false Answer: false Difficulty: easy
How many days are there between full moons? Answer: 28 Difficulty: hard
```

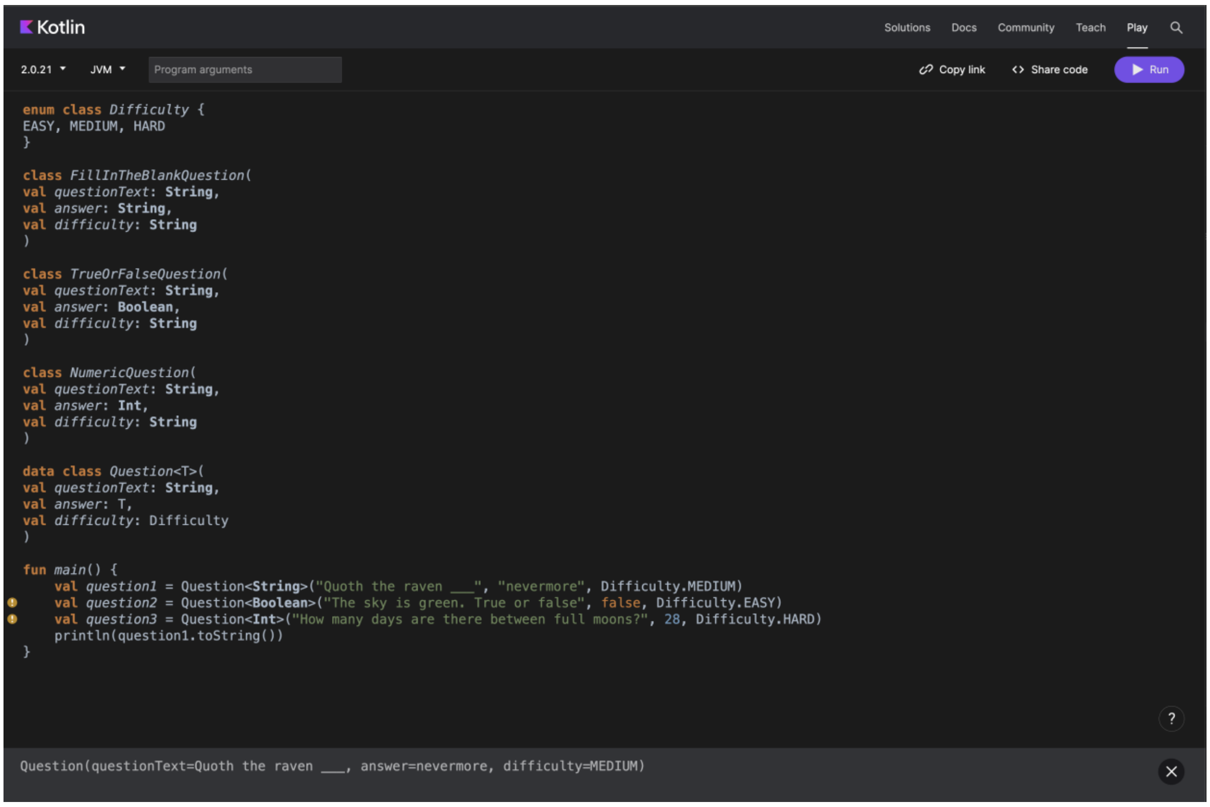
Задание 2.



```kotlin
enum class Difficulty {
EASY, MEDIUM, HARD
}

class FillInTheBlankQuestion(
val questionText: String,
val answer: String,
val difficulty: String
)

class TrueOrFalseQuestion(
val questionText: String,
val answer: Boolean,
val difficulty: String
)

class NumericQuestion(
val questionText: String,
val answer: Int,
val difficulty: String
)

class Question<T>(
val questionText: String,
val answer: T,
val difficulty: Difficulty
)

fun main() {
    val question1 = Question<String>("Quoth the raven ___", "nevermore", Difficulty.MEDIUM)
    val question2 = Question<Boolean>("The sky is green. True or false", false, Difficulty.EASY)
    val question3 = Question<Int>("How many days are there between full moons?", 28, Difficulty.HARD)

    println("${question1.questionText} Answer: ${question1.answer} Difficulty: ${question1.difficulty}")
    println("${question2.questionText} Answer: ${question2.answer} Difficulty: ${question2.difficulty}")
    println("${question3.questionText} Answer: ${question3.answer} Difficulty: ${question3.difficulty}")
}
```

```
Quoth the raven ___ Answer: nevermore Difficulty: MEDIUM
The sky is green. True or false Answer: false Difficulty: EASY
How many days are there between full moons? Answer: 28 Difficulty: HARD
```

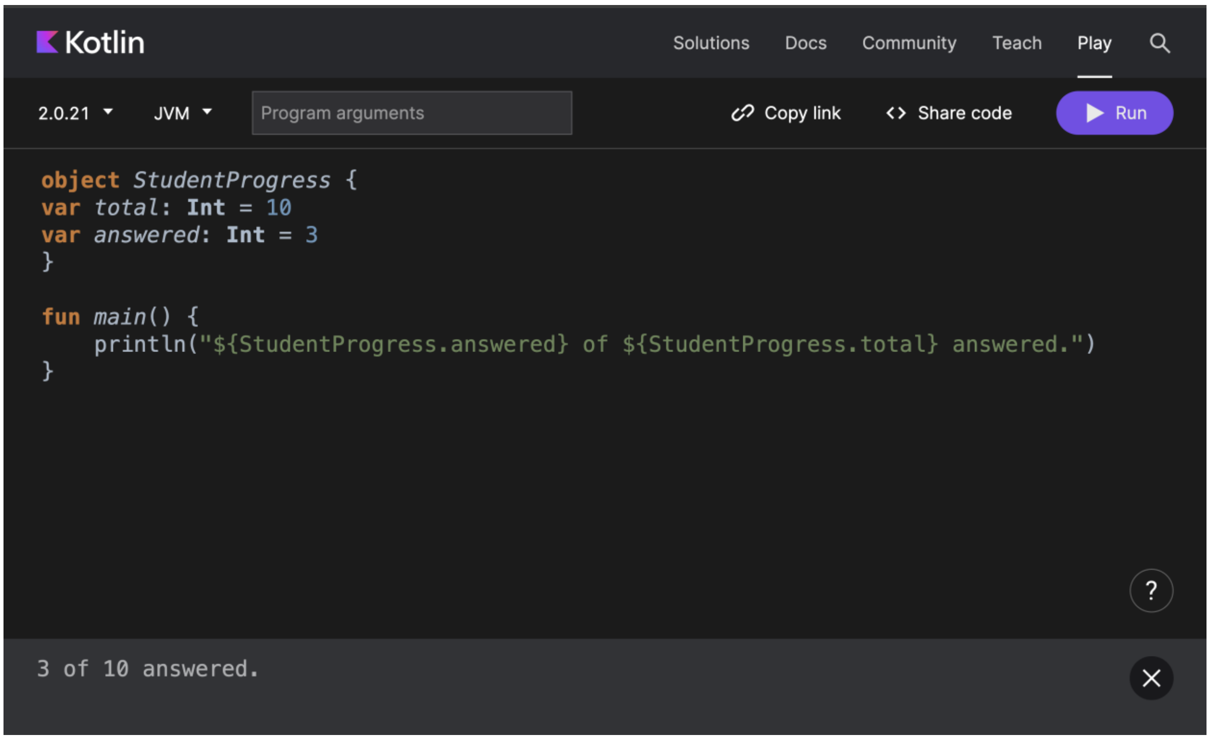
Задание 3.



```kotlin
enum class Difficulty {
EASY, MEDIUM, HARD
}

class FillInTheBlankQuestion(
val questionText: String,
val answer: String,
val difficulty: String
)

class TrueOrFalseQuestion(
val questionText: String,
val answer: Boolean,
val difficulty: String
)

class NumericQuestion(
val questionText: String,
val answer: Int,
val difficulty: String
)

data class Question<T>(
val questionText: String,
val answer: T,
val difficulty: Difficulty
)

fun main() {
    val question1 = Question<String>("Quoth the raven ___", "nevermore", Difficulty.MEDIUM)
    val question2 = Question<Boolean>("The sky is green. True or false", false, Difficulty.EASY)
    val question3 = Question<Int>("How many days are there between full moons?", 28, Difficulty.HARD)
    println(question1.toString())
}
```

```
Question(questionText=Quoth the raven ___, answer=nevermore, difficulty=MEDIUM)
```

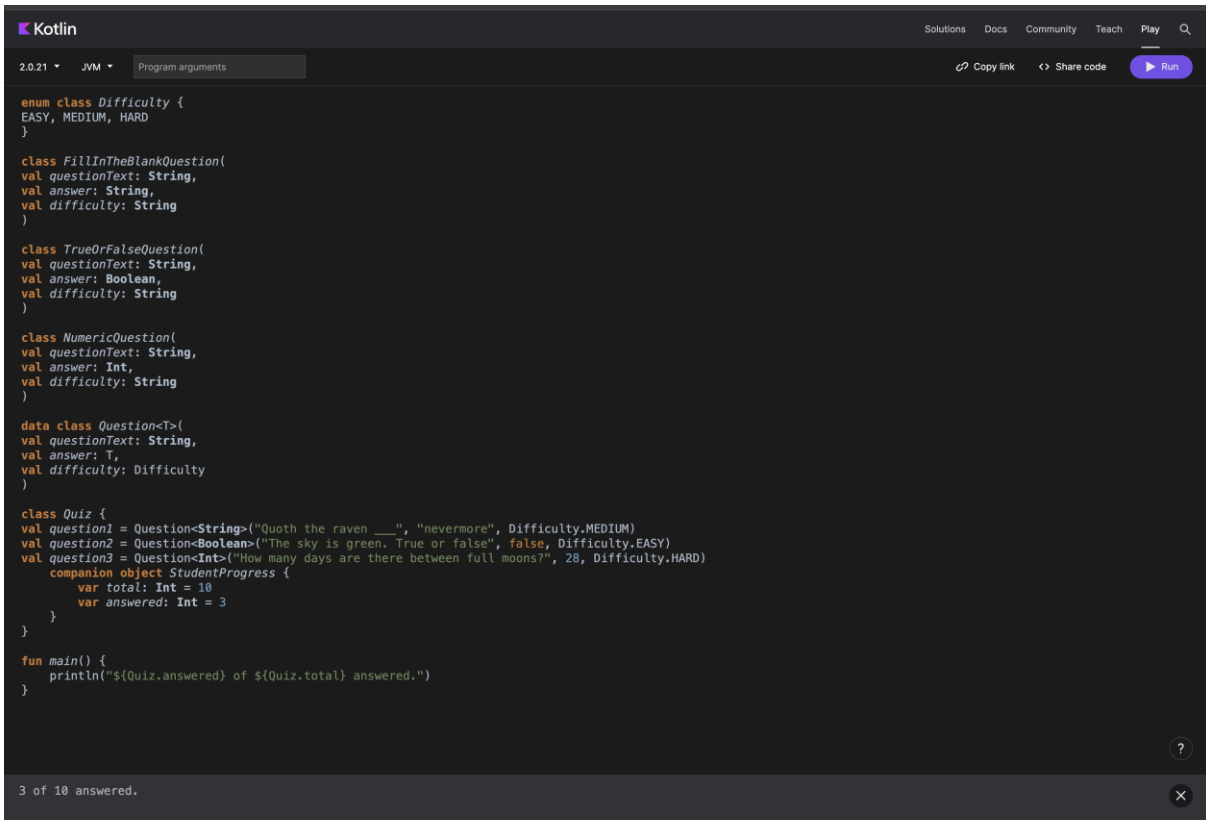
Задание 4.



```kotlin
object StudentProgress {
var total: Int = 10
var answered: Int = 3
}

fun main() {
    println("${StudentProgress.answered} of ${StudentProgress.total} answered.")
}
```

```
3 of 10 answered.
```

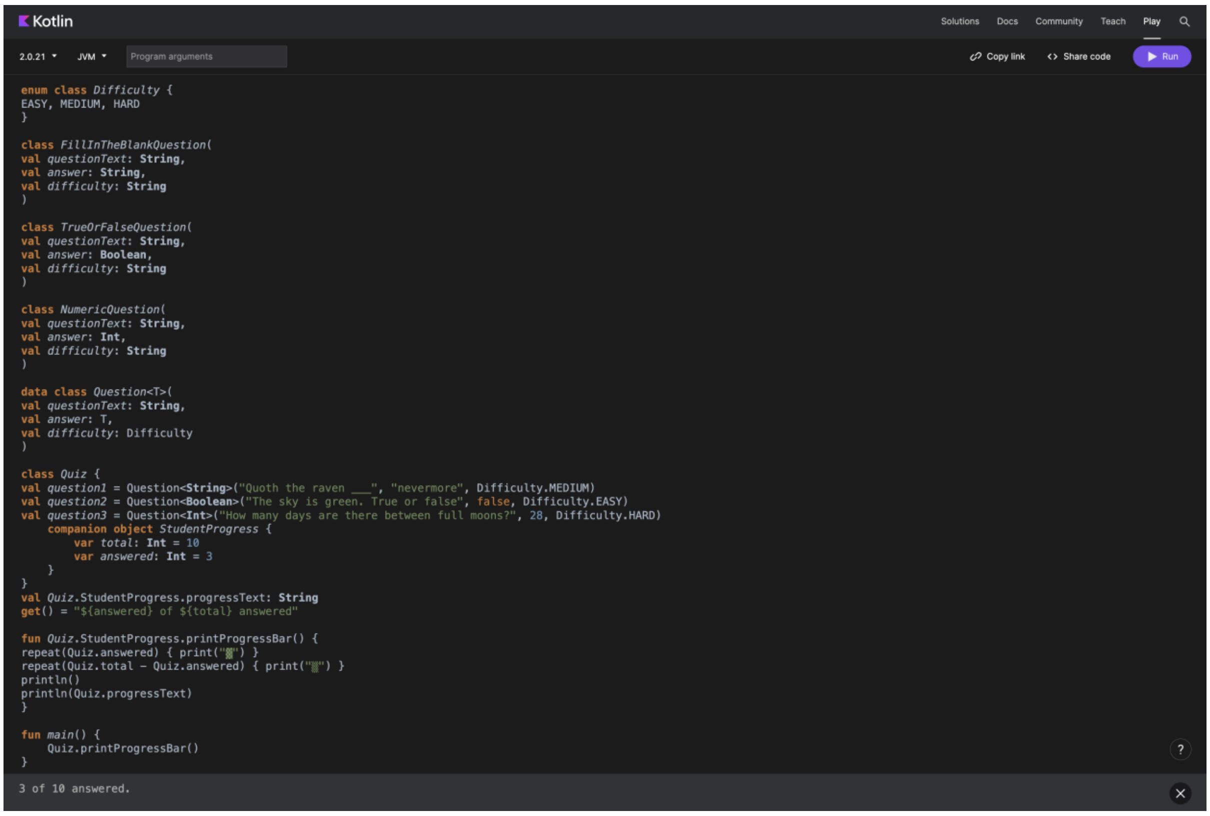
Задание 5.



```kotlin
enum class Difficulty {
EASY, MEDIUM, HARD
}

class FillInTheBlankQuestion(
val questionText: String,
val answer: String,
val difficulty: String
)

class TrueOrFalseQuestion(
val questionText: String,
val answer: Boolean,
val difficulty: String
)

class NumericQuestion(
val questionText: String,
val answer: Int,
val difficulty: String
)

data class Question<T>(
val questionText: String,
val answer: T,
val difficulty: Difficulty
)

class Quiz {
val question1 = Question<String>("Quoth the raven ___", "nevermore", Difficulty.MEDIUM)
val question2 = Question<Boolean>("The sky is green. True or false", false, Difficulty.EASY)
val question3 = Question<Int>("How many days are there between full moons?", 28, Difficulty.HARD)
    companion object StudentProgress {
        var total: Int = 10
        var answered: Int = 3
    }
}

fun main() {
    println("${Quiz.answered} of ${Quiz.total} answered.")
}
```

```
3 of 10 answered.
```

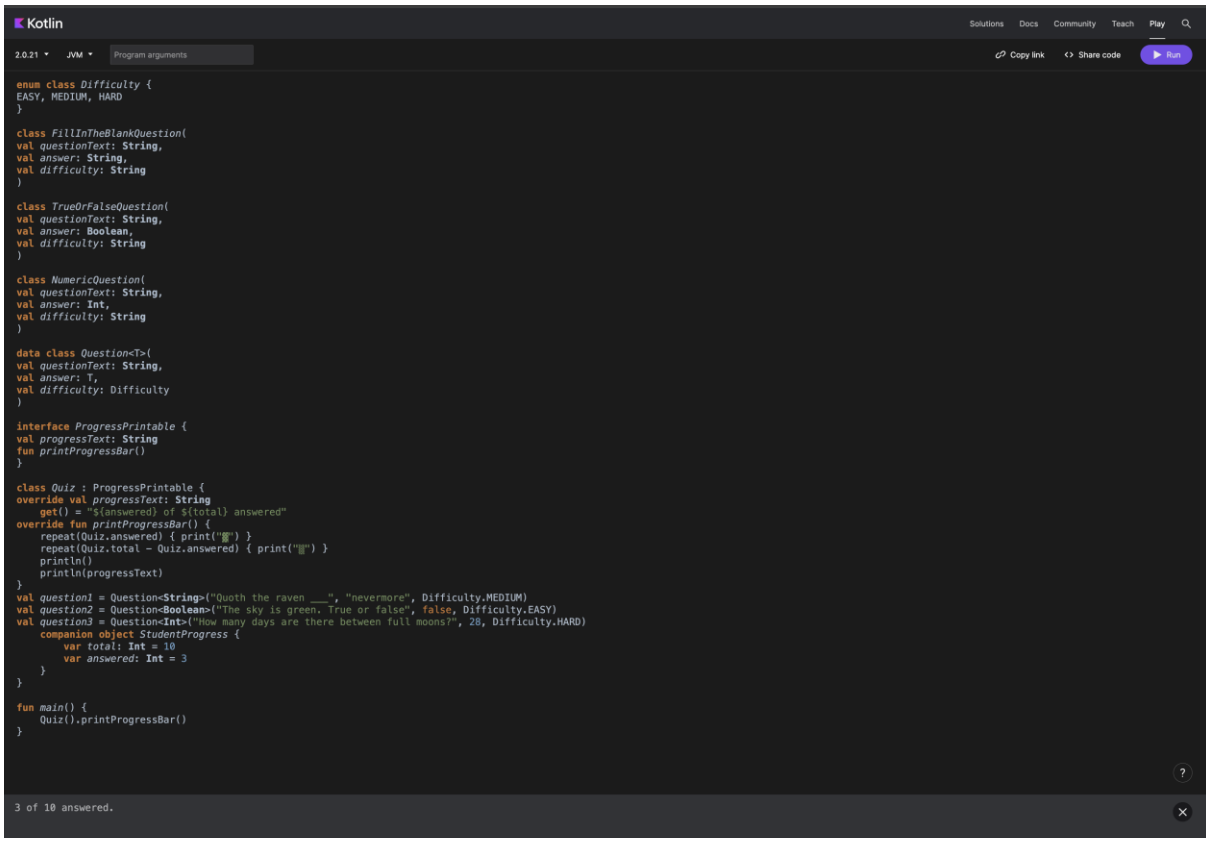
Задание 6.



```kotlin
enum class Difficulty {
EASY, MEDIUM, HARD
}

class FillInTheBlankQuestion(
val questionText: String,
val answer: String,
val difficulty: String
)

class TrueOrFalseQuestion(
val questionText: String,
val answer: Boolean,
val difficulty: String
)

class NumericQuestion(
val questionText: String,
val answer: Int,
val difficulty: String
)

data class Question<T>(
val questionText: String,
val answer: T,
val difficulty: Difficulty
)

class Quiz {
val question1 = Question<String>("Quoth the raven ___", "nevermore", Difficulty.MEDIUM)
val question2 = Question<Boolean>("The sky is green. True or false", false, Difficulty.EASY)
val question3 = Question<Int>("How many days are there between full moons?", 28, Difficulty.HARD)
    companion object StudentProgress {
        var total: Int = 10
        var answered: Int = 3
    }
}
val Quiz.StudentProgress.progressText: String
get() = "${answered} of ${total} answered"

fun Quiz.StudentProgress.printProgressBar() {
repeat(Quiz.answered) { print("▓") }
repeat(Quiz.total - Quiz.answered) { print("▒") }
println()
println(Quiz.progressText)
}

fun main() {
    Quiz.printProgressBar()
}
```

```
3 of 10 answered.
```

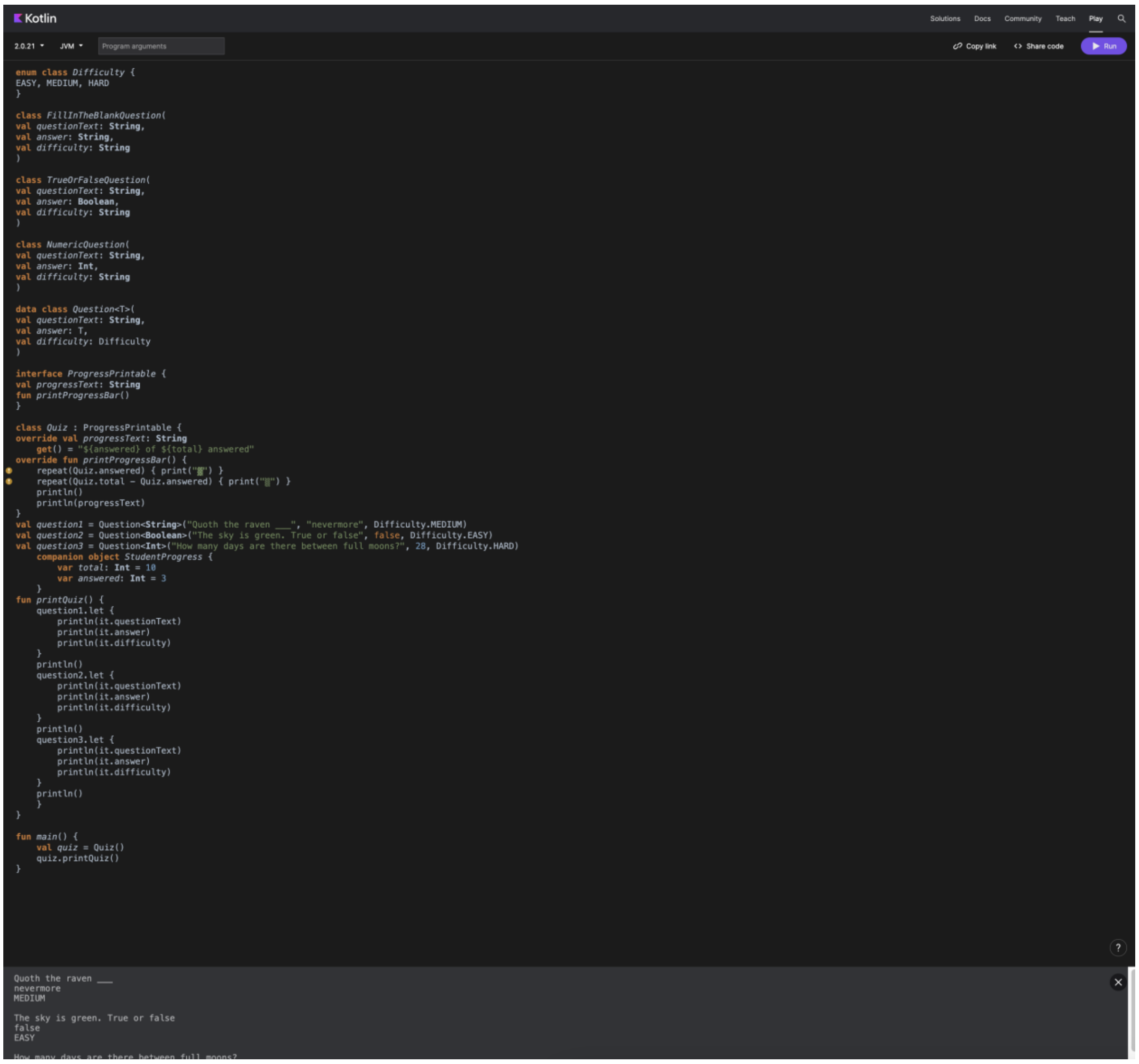
Задание 7.



```kotlin
enum class Difficulty {
EASY, MEDIUM, HARD
}

class FillInTheBlankQuestion(
val questionText: String,
val answer: String,
val difficulty: String
)

class TrueOrFalseQuestion(
val questionText: String,
val answer: Boolean,
val difficulty: String
)

class NumericQuestion(
val questionText: String,
val answer: Int,
val difficulty: String
)

data class Question<T>(
val questionText: String,
val answer: T,
val difficulty: Difficulty
)

interface ProgressPrintable {
val progressText: String
fun printProgressBar()
}

class Quiz : ProgressPrintable {
override val progressText: String
    get() = "${answered} of ${total} answered"
override fun printProgressBar() {
    repeat(Quiz.answered) { print("▓") }
    repeat(Quiz.total - Quiz.answered) { print("▒") }
    println()
    println(progressText)
}
val question1 = Question<String>("Quoth the raven ___", "nevermore", Difficulty.MEDIUM)
val question2 = Question<Boolean>("The sky is green. True or false", false, Difficulty.EASY)
val question3 = Question<Int>("How many days are there between full moons?", 28, Difficulty.HARD)
    companion object StudentProgress {
        var total: Int = 10
        var answered: Int = 3
    }
}

fun main() {
    Quiz().printProgressBar()
}
```

```
3 of 10 answered.
```

Задание 8.

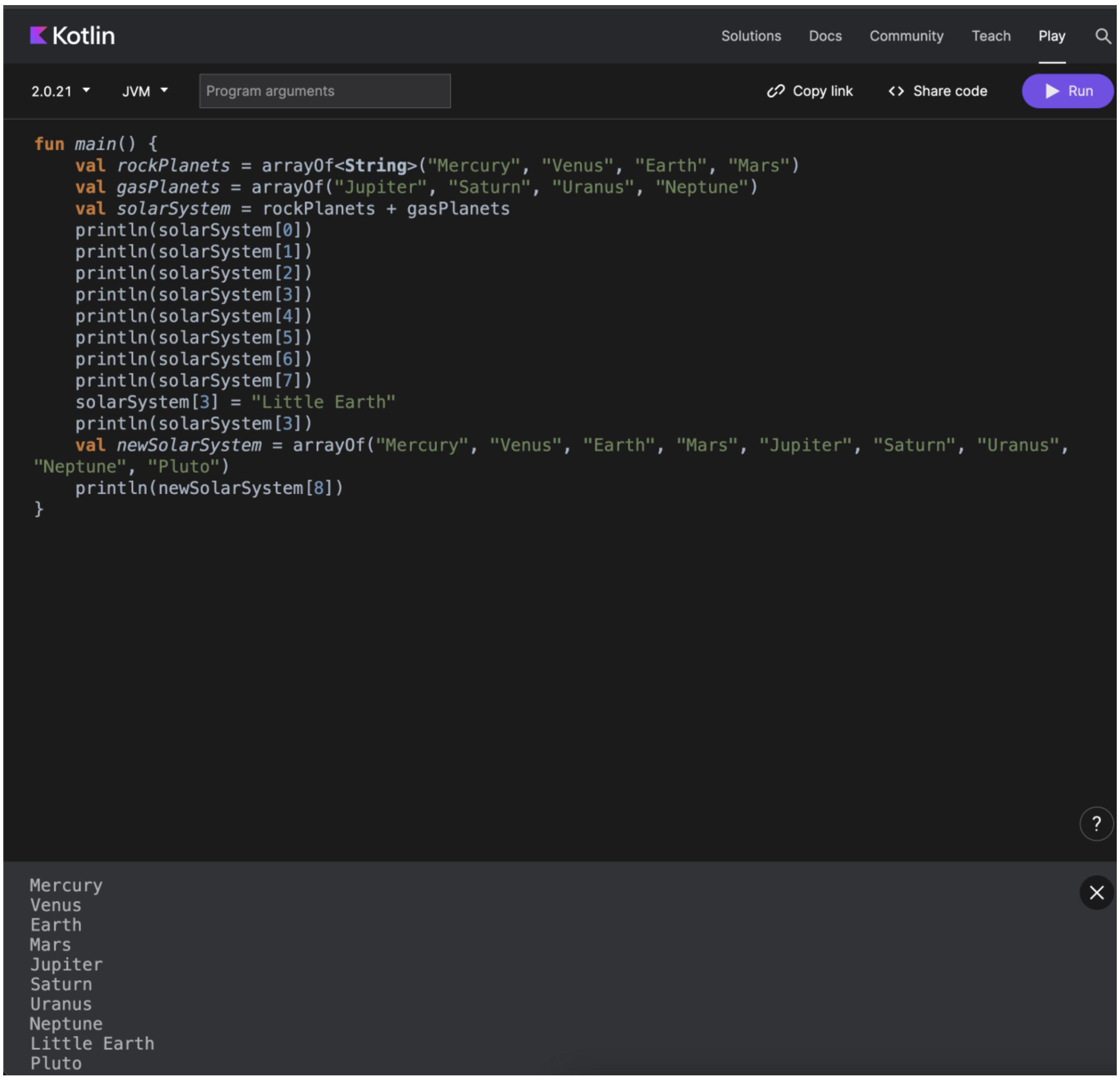


```kotlin
enum class Difficulty {
EASY, MEDIUM, HARD
}

class FillInTheBlankQuestion(
val questionText: String,
val answer: String,
val difficulty: String
)

class TrueOrFalseQuestion(
val questionText: String,
val answer: Boolean,
val difficulty: String
)

class NumericQuestion(
val questionText: String,
val answer: Int,
val difficulty: String
)

data class Question<T>(
val questionText: String,
val answer: T,
val difficulty: Difficulty
)

interface ProgressPrintable {
val progressText: String
fun printProgressBar()
}

class Quiz : ProgressPrintable {
override val progressText: String
    get() = "${answered} of ${total} answered"
override fun printProgressBar() {
    repeat(Quiz.answered) { print("▓") }
    repeat(Quiz.total - Quiz.answered) { print("▒") }
    println()
    println(progressText)
}
val question1 = Question<String>("Quoth the raven ___", "nevermore", Difficulty.MEDIUM)
val question2 = Question<Boolean>("The sky is green. True or false", false, Difficulty.EASY)
val question3 = Question<Int>("How many days are there between full moons?", 28, Difficulty.HARD)
    companion object StudentProgress {
        var total: Int = 10
        var answered: Int = 3
    }
fun printQuiz() {
    question1.let {
        println(it.questionText)
        println(it.answer)
        println(it.difficulty)
    }
    println()
    question2.let {
        println(it.questionText)
        println(it.answer)
        println(it.difficulty)
    }
    println()
    question3.let {
        println(it.questionText)
        println(it.answer)
        println(it.difficulty)
    }
    println()
    }
}

fun main() {
    val quiz = Quiz()
    quiz.printQuiz()
}
```

```
Quoth the raven ___
nevermore
MEDIUM

The sky is green. True or false
false
EASY

How many days are there between full moons?
```

Практическая работа №16

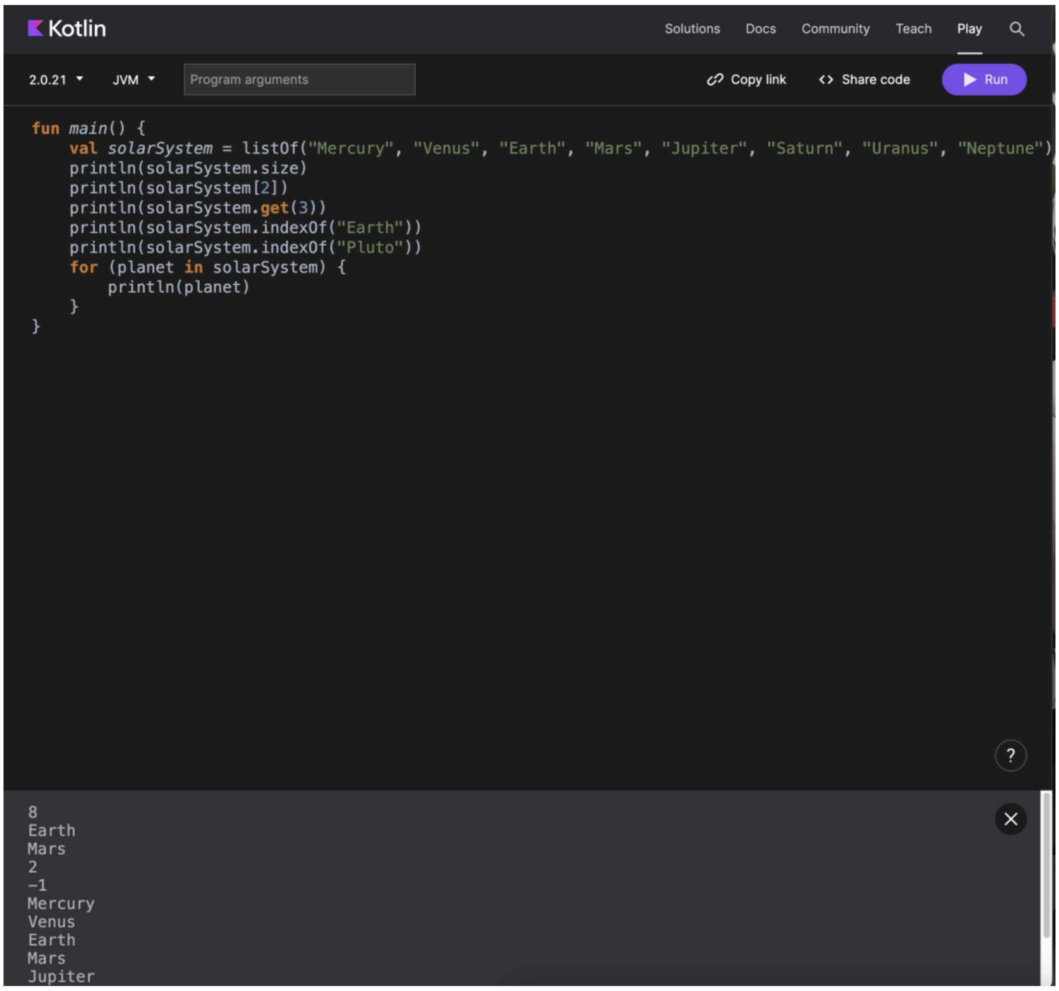
Задание 1-2.



```kotlin
fun main() {
    val rockPlanets = arrayOf<String>("Mercury", "Venus", "Earth", "Mars")
    val gasPlanets = arrayOf("Jupiter", "Saturn", "Uranus", "Neptune")
    val solarSystem = rockPlanets + gasPlanets
    println(solarSystem[0])
    println(solarSystem[1])
    println(solarSystem[2])
    println(solarSystem[3])
    println(solarSystem[4])
    println(solarSystem[5])
    println(solarSystem[6])
    println(solarSystem[7])
    solarSystem[3] = "Little Earth"
    println(solarSystem[3])
    val newSolarSystem = arrayOf("Mercury", "Venus", "Earth", "Mars", "Jupiter", "Saturn", "Uranus",
"Neptune", "Pluto")
    println(newSolarSystem[8])
}
```

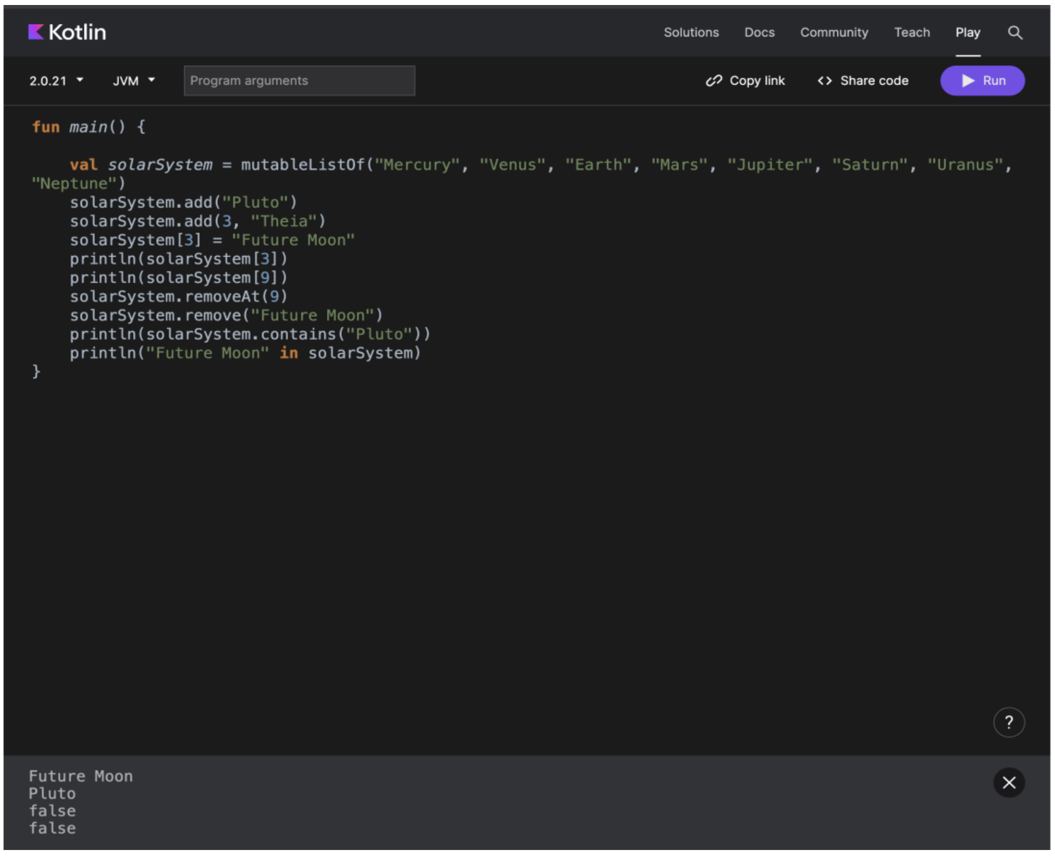
```
Mercury
Venus
Earth
Mars
Jupiter
Saturn
Uranus
Neptune
Little Earth
Pluto
```

Задание 3.



```
fun main() {
    val solarSystem = listOf("Mercury", "Venus", "Earth", "Mars", "Jupiter", "Saturn", "Uranus", "Neptune")
    println(solarSystem.size)
    println(solarSystem[2])
    println(solarSystem.get(3))
    println(solarSystem.indexOf("Earth"))
    println(solarSystem.indexOf("Pluto"))
    for (planet in solarSystem) {
        println(planet)
    }
}
```

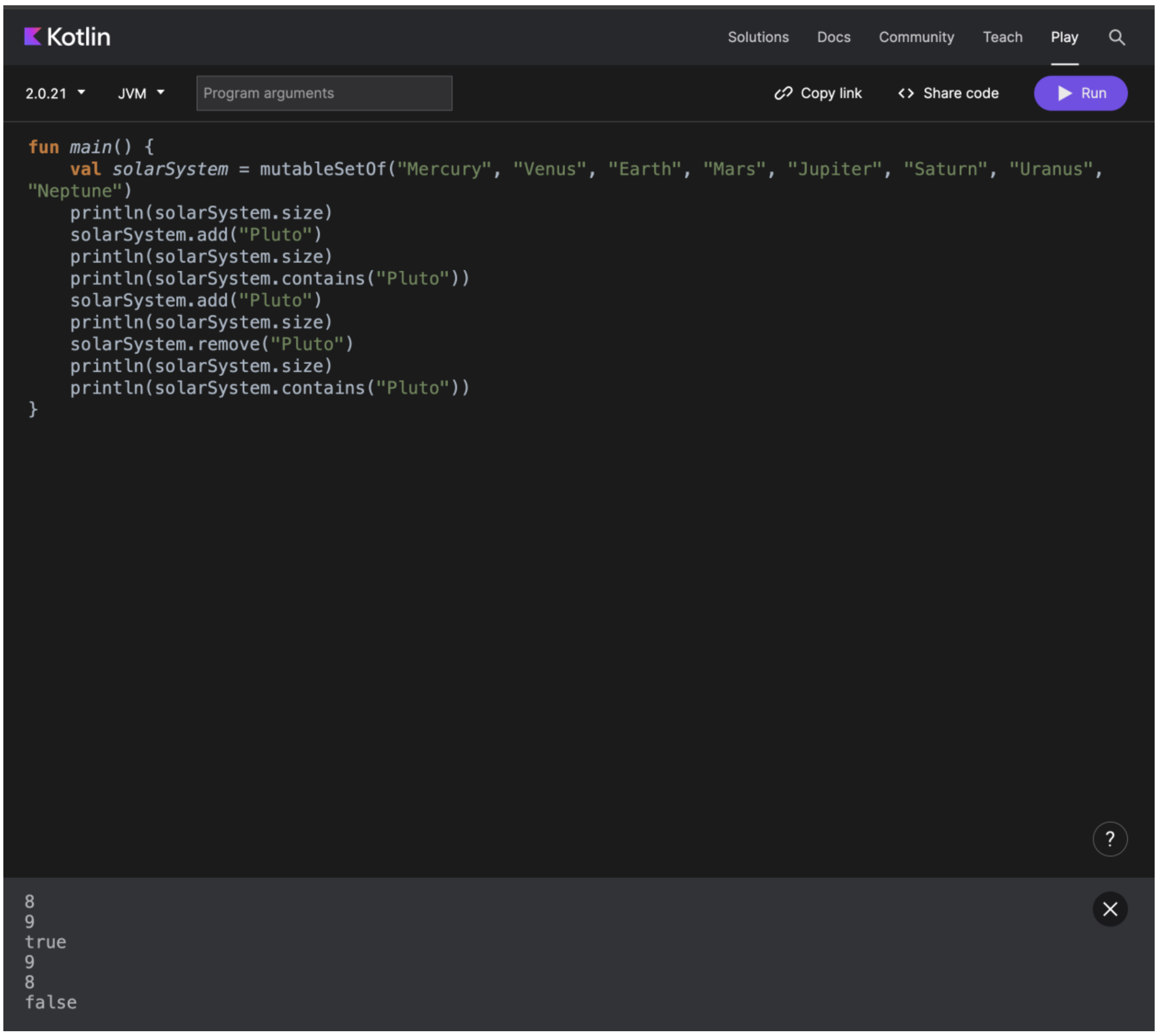
```
8
Earth
Mars
2
-1
Mercury
Venus
Earth
Mars
Jupiter
```



```
fun main() {

    val solarSystem = mutableListOf("Mercury", "Venus", "Earth", "Mars", "Jupiter", "Saturn", "Uranus",
"Neptune")
    solarSystem.add("Pluto")
    solarSystem.add(3, "Theia")
    solarSystem[3] = "Future Moon"
    println(solarSystem[3])
    println(solarSystem[9])
    solarSystem.removeAt(9)
    solarSystem.remove("Future Moon")
    println(solarSystem.contains("Pluto"))
    println("Future Moon" in solarSystem)
}
```

```
Future Moon
Pluto
false
false
```

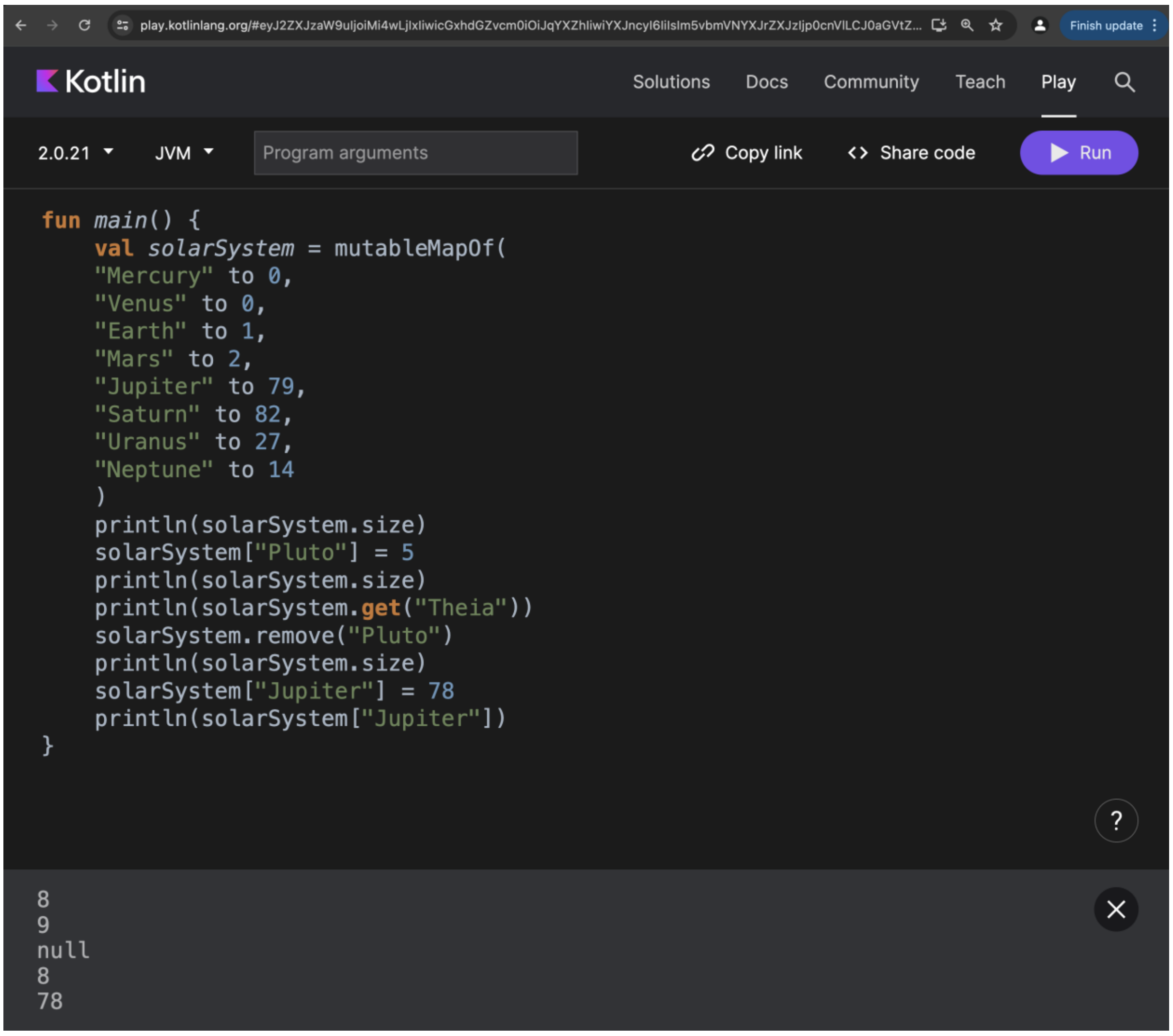
Задание 4.



```
fun main() {
    val solarSystem = mutableSetOf("Mercury", "Venus", "Earth", "Mars", "Jupiter", "Saturn", "Uranus",
"Neptune")
    println(solarSystem.size)
    solarSystem.add("Pluto")
    println(solarSystem.size)
    println(solarSystem.contains("Pluto"))
    solarSystem.add("Pluto")
    println(solarSystem.size)
    solarSystem.remove("Pluto")
    println(solarSystem.size)
    println(solarSystem.contains("Pluto"))
}
```

```
8
9
true
9
8
false
```

Задание 5.



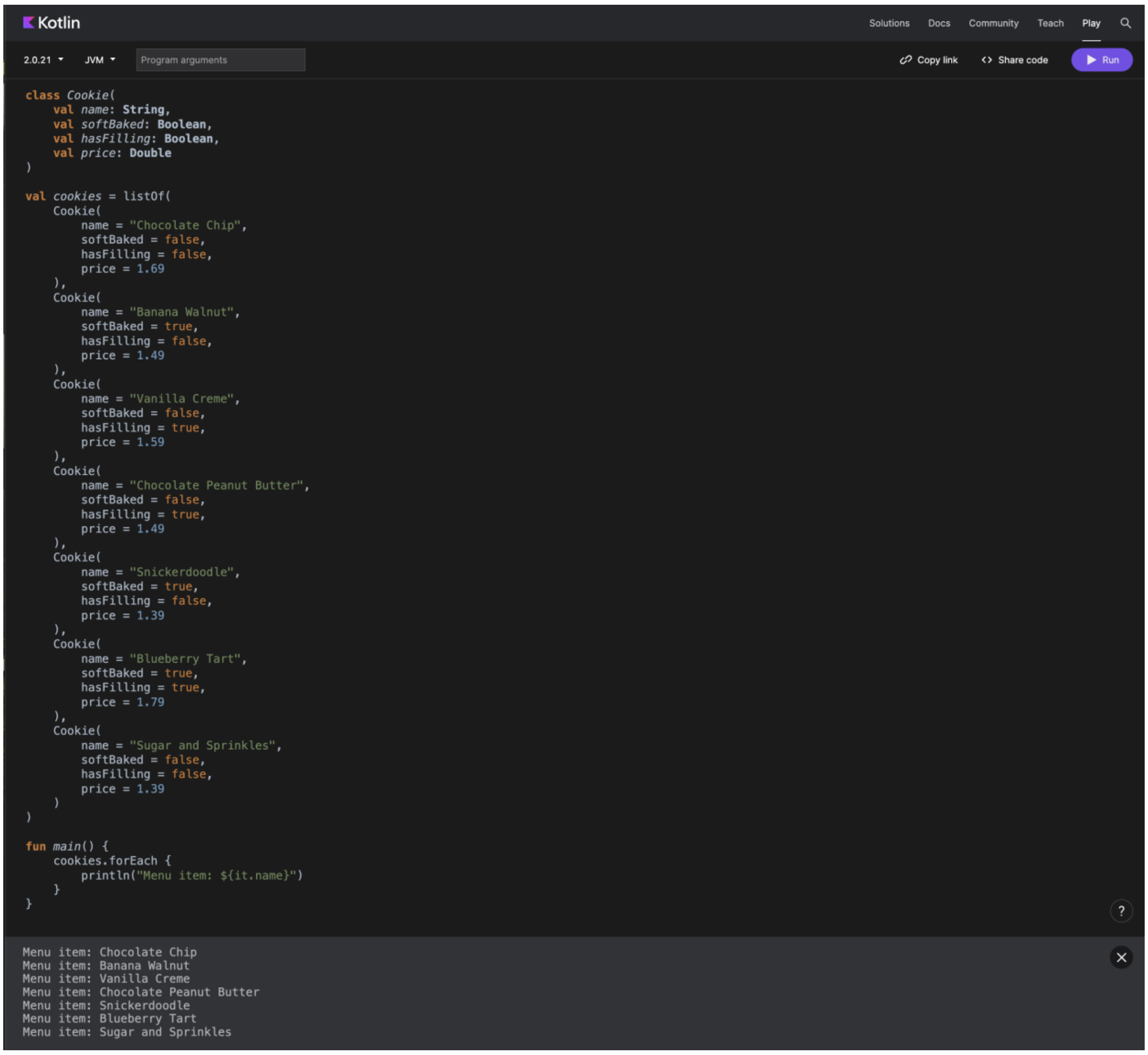
play.kotlinlang.org/#eyJ2ZXJzaW9uIjoiMi4wLjIxIiwicGxhdGZvcm0iOiJqYXZhIiwiYXJncyI6IiIsIm5vbmVNYXJrZXJzIjp0cnVlLCJ0aGVtZ...
Finish update
Kotlin
Solutions
Docs
Community
Teach
Play
2.0.21
JVM
Program arguments
Copy link
Share code
Run

```kotlin
fun main() {
    val solarSystem = mutableMapOf(
    "Mercury" to 0,
    "Venus" to 0,
    "Earth" to 1,
    "Mars" to 2,
    "Jupiter" to 79,
    "Saturn" to 82,
    "Uranus" to 27,
    "Neptune" to 14
    )
    println(solarSystem.size)
    solarSystem["Pluto"] = 5
    println(solarSystem.size)
    println(solarSystem.get("Theia"))
    solarSystem.remove("Pluto")
    println(solarSystem.size)
    solarSystem["Jupiter"] = 78
    println(solarSystem["Jupiter"])
}
```

```
8
9
null
8
78
```

Практическая работа №17

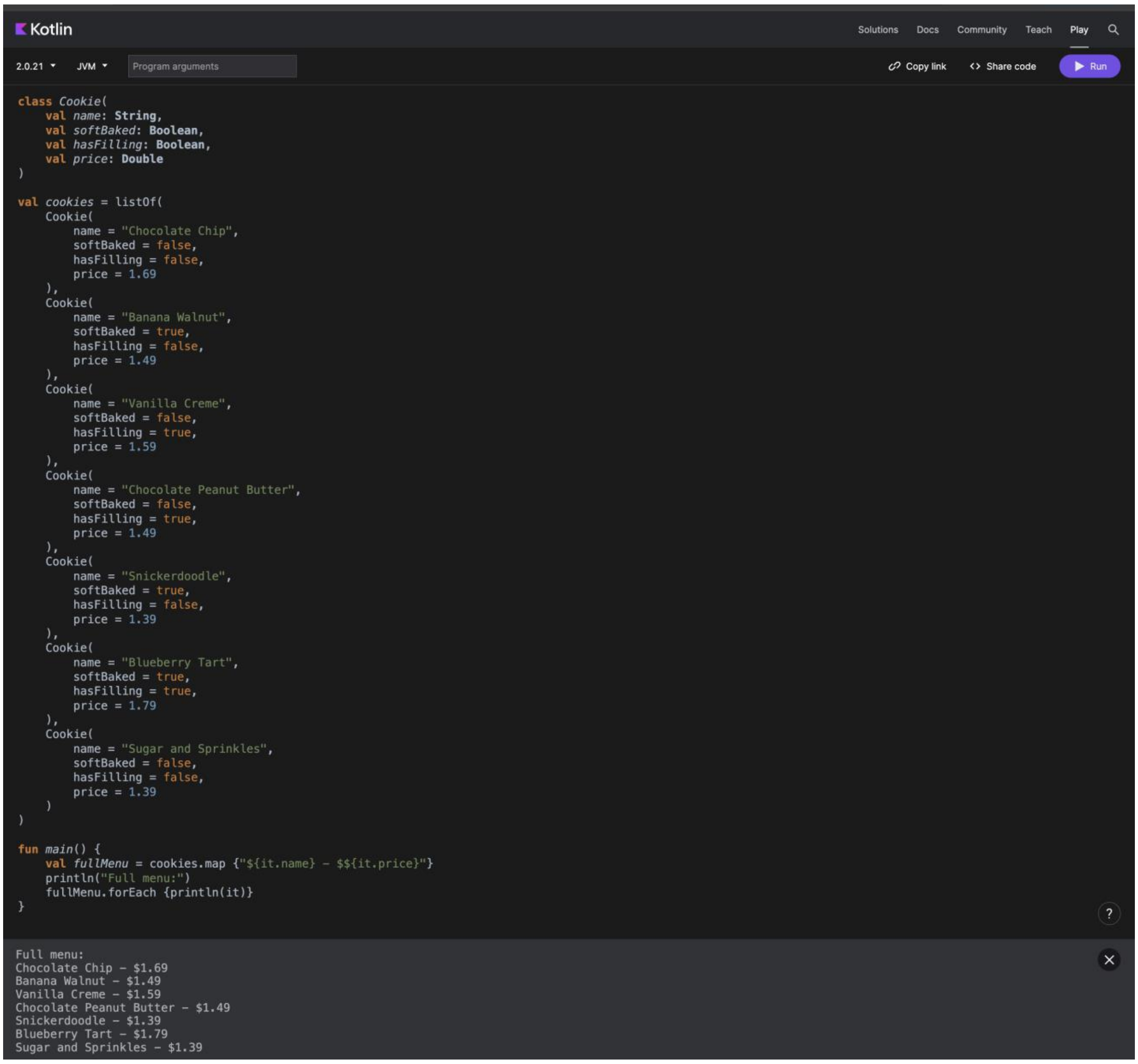
Задание 1-2.



```kotlin
class Cookie(
    val name: String,
    val softBaked: Boolean,
    val hasFilling: Boolean,
    val price: Double
)

val cookies = listOf(
    Cookie(
        name = "Chocolate Chip",
        softBaked = false,
        hasFilling = false,
        price = 1.69
    ),
    Cookie(
        name = "Banana Walnut",
        softBaked = true,
        hasFilling = false,
        price = 1.49
    ),
    Cookie(
        name = "Vanilla Creme",
        softBaked = false,
        hasFilling = true,
        price = 1.59
    ),
    Cookie(
        name = "Chocolate Peanut Butter",
        softBaked = false,
        hasFilling = true,
        price = 1.49
    ),
    Cookie(
        name = "Snickerdoodle",
        softBaked = true,
        hasFilling = false,
        price = 1.39
    ),
    Cookie(
        name = "Blueberry Tart",
        softBaked = true,
        hasFilling = true,
        price = 1.79
    ),
    Cookie(
        name = "Sugar and Sprinkles",
        softBaked = false,
        hasFilling = false,
        price = 1.39
    )
)

fun main() {
    cookies.forEach {
        println("Menu item: ${it.name}")
    }
}
```

```
Menu item: Chocolate Chip
Menu item: Banana Walnut
Menu item: Vanilla Creme
Menu item: Chocolate Peanut Butter
Menu item: Snickerdoodle
Menu item: Blueberry Tart
Menu item: Sugar and Sprinkles
```

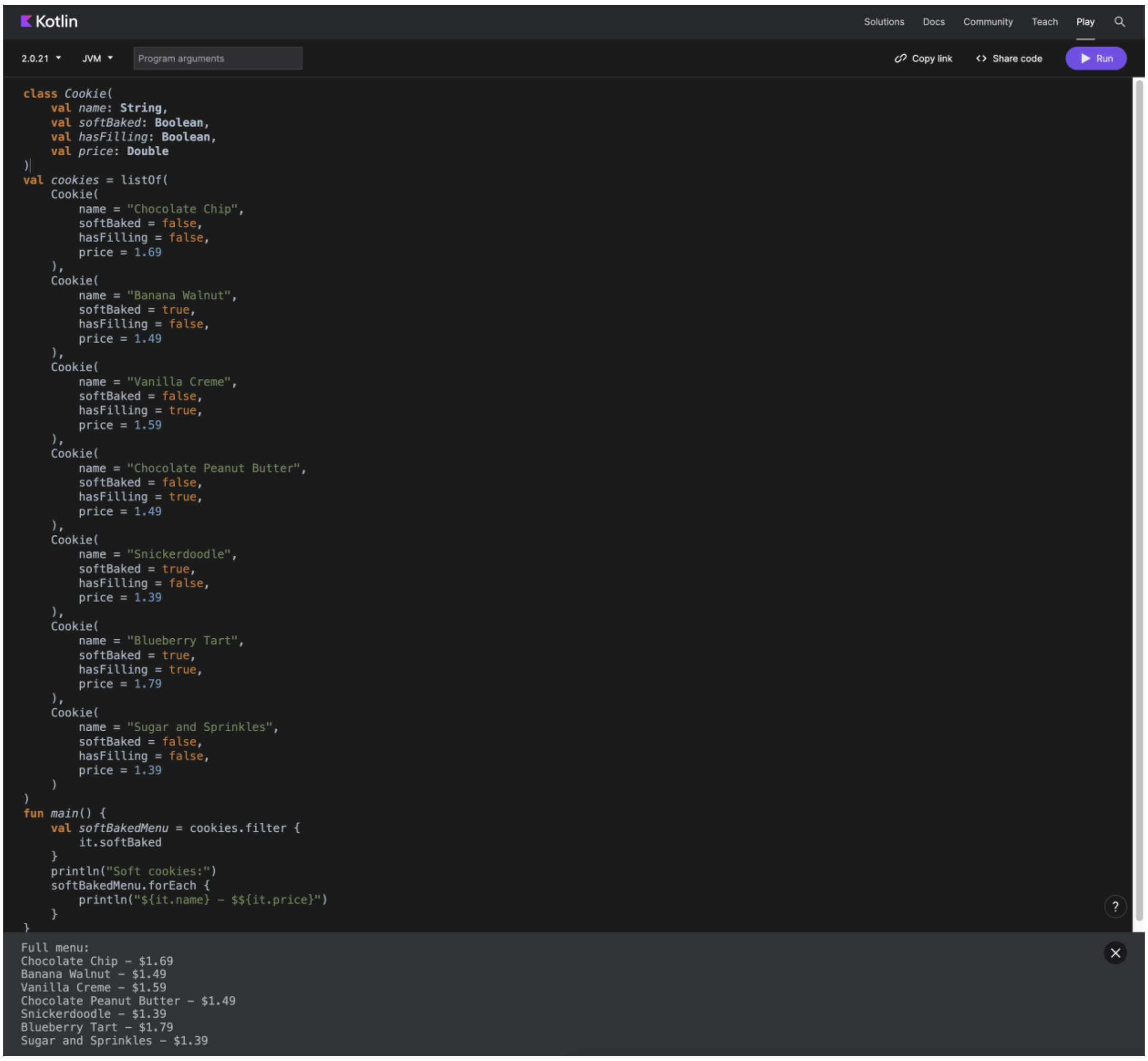
Задание 3.



```kotlin
class Cookie(
    val name: String,
    val softBaked: Boolean,
    val hasFilling: Boolean,
    val price: Double
)

val cookies = listOf(
    Cookie(
        name = "Chocolate Chip",
        softBaked = false,
        hasFilling = false,
        price = 1.69
    ),
    Cookie(
        name = "Banana Walnut",
        softBaked = true,
        hasFilling = false,
        price = 1.49
    ),
    Cookie(
        name = "Vanilla Creme",
        softBaked = false,
        hasFilling = true,
        price = 1.59
    ),
    Cookie(
        name = "Chocolate Peanut Butter",
        softBaked = false,
        hasFilling = true,
        price = 1.49
    ),
    Cookie(
        name = "Snickerdoodle",
        softBaked = true,
        hasFilling = false,
        price = 1.39
    ),
    Cookie(
        name = "Blueberry Tart",
        softBaked = true,
        hasFilling = true,
        price = 1.79
    ),
    Cookie(
        name = "Sugar and Sprinkles",
        softBaked = false,
        hasFilling = false,
        price = 1.39
    )
)

fun main() {
    val fullMenu = cookies.map {"${it.name} - $${it.price}"}
    println("Full menu:")
    fullMenu.forEach {println(it)}
}
```

```
Full menu:
Chocolate Chip - $1.69
Banana Walnut - $1.49
Vanilla Creme - $1.59
Chocolate Peanut Butter - $1.49
Snickerdoodle - $1.39
Blueberry Tart - $1.79
Sugar and Sprinkles - $1.39
```

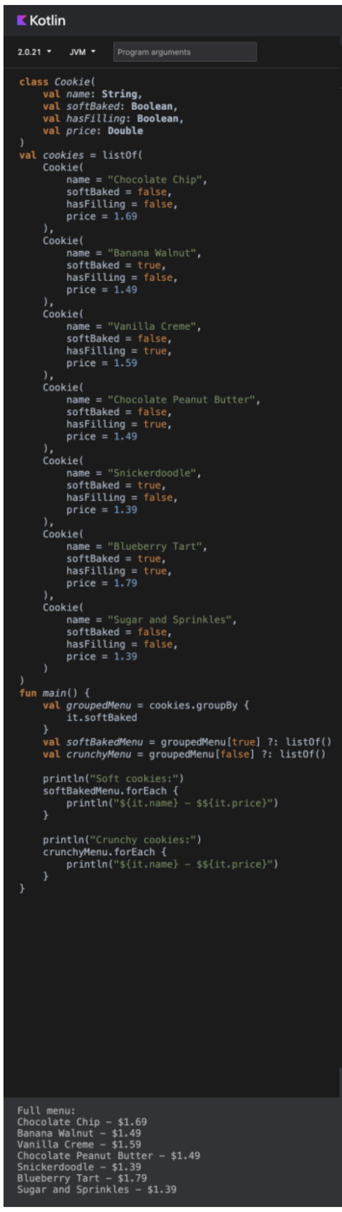
Задание 4.



```kotlin
class Cookie(
    val name: String,
    val softBaked: Boolean,
    val hasFilling: Boolean,
    val price: Double
)
val cookies = listOf(
    Cookie(
        name = "Chocolate Chip",
        softBaked = false,
        hasFilling = false,
        price = 1.69
    ),
    Cookie(
        name = "Banana Walnut",
        softBaked = true,
        hasFilling = false,
        price = 1.49
    ),
    Cookie(
        name = "Vanilla Creme",
        softBaked = false,
        hasFilling = true,
        price = 1.59
    ),
    Cookie(
        name = "Chocolate Peanut Butter",
        softBaked = false,
        hasFilling = true,
        price = 1.49
    ),
    Cookie(
        name = "Snickerdoodle",
        softBaked = true,
        hasFilling = false,
        price = 1.39
    ),
    Cookie(
        name = "Blueberry Tart",
        softBaked = true,
        hasFilling = true,
        price = 1.79
    ),
    Cookie(
        name = "Sugar and Sprinkles",
        softBaked = false,
        hasFilling = false,
        price = 1.39
    )
)
fun main() {
    val softBakedMenu = cookies.filter {
        it.softBaked
    }
    println("Soft cookies:")
    softBakedMenu.forEach {
        println("${it.name} - $${it.price}")
    }
}
```

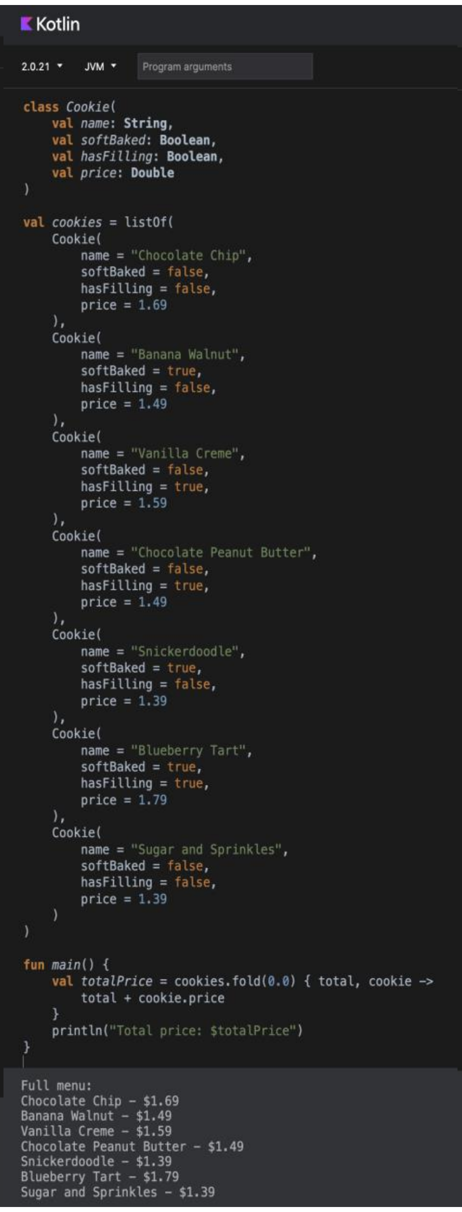
```
Full menu:
Chocolate Chip - $1.69
Banana Walnut - $1.49
Vanilla Creme - $1.59
Chocolate Peanut Butter - $1.49
Snickerdoodle - $1.39
Blueberry Tart - $1.79
Sugar and Sprinkles - $1.39
```

Задание 5,6,7.



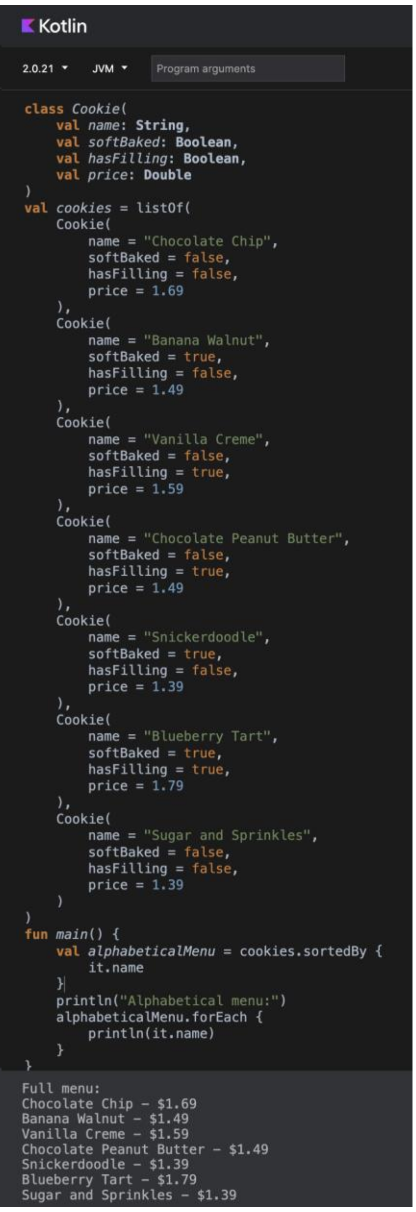
```kotlin
class Cookie(
    val name: String,
    val softBaked: Boolean,
    val hasFilling: Boolean,
    val price: Double
)
val cookies = listOf(
    Cookie(
        name = "Chocolate Chip",
        softBaked = false,
        hasFilling = false,
        price = 1.69
    ),
    Cookie(
        name = "Banana Walnut",
        softBaked = true,
        hasFilling = false,
        price = 1.49
    ),
    Cookie(
        name = "Vanilla Creme",
        softBaked = false,
        hasFilling = true,
        price = 1.59
    ),
    Cookie(
        name = "Chocolate Peanut Butter",
        softBaked = false,
        hasFilling = true,
        price = 1.49
    ),
    Cookie(
        name = "Snickerdoodle",
        softBaked = true,
        hasFilling = false,
        price = 1.39
    ),
    Cookie(
        name = "Blueberry Tart",
        softBaked = true,
        hasFilling = true,
        price = 1.79
    ),
    Cookie(
        name = "Sugar and Sprinkles",
        softBaked = false,
        hasFilling = false,
        price = 1.39
    )
)
fun main() {
    val groupedMenu = cookies.groupBy {
        it.softBaked
    }
    val softBakedMenu = groupedMenu[true] ?: listOf()
    val crunchyMenu = groupedMenu[false] ?: listOf()

    println("Soft cookies:")
    softBakedMenu.forEach {
        println("${it.name} - $${it.price}")
    }

    println("Crunchy cookies:")
    crunchyMenu.forEach {
        println("${it.name} - $${it.price}")
    }
}
```

```
Full menu:
Chocolate Chip - $1.69
Banana Walnut - $1.49
Vanilla Creme - $1.59
Chocolate Peanut Butter - $1.49
Snickerdoodle - $1.39
Blueberry Tart - $1.79
Sugar and Sprinkles - $1.39
```



```kotlin
class Cookie(
    val name: String,
    val softBaked: Boolean,
    val hasFilling: Boolean,
    val price: Double
)

val cookies = listOf(
    Cookie(
        name = "Chocolate Chip",
        softBaked = false,
        hasFilling = false,
        price = 1.69
    ),
    Cookie(
        name = "Banana Walnut",
        softBaked = true,
        hasFilling = false,
        price = 1.49
    ),
    Cookie(
        name = "Vanilla Creme",
        softBaked = false,
        hasFilling = true,
        price = 1.59
    ),
    Cookie(
        name = "Chocolate Peanut Butter",
        softBaked = false,
        hasFilling = true,
        price = 1.49
    ),
    Cookie(
        name = "Snickerdoodle",
        softBaked = true,
        hasFilling = false,
        price = 1.39
    ),
    Cookie(
        name = "Blueberry Tart",
        softBaked = true,
        hasFilling = true,
        price = 1.79
    ),
    Cookie(
        name = "Sugar and Sprinkles",
        softBaked = false,
        hasFilling = false,
        price = 1.39
    )
)

fun main() {
    val totalPrice = cookies.fold(0.0) { total, cookie ->
        total + cookie.price
    }
    println("Total price: $totalPrice")
}
```

```
Full menu:
Chocolate Chip - $1.69
Banana Walnut - $1.49
Vanilla Creme - $1.59
Chocolate Peanut Butter - $1.49
Snickerdoodle - $1.39
Blueberry Tart - $1.79
Sugar and Sprinkles - $1.39
```

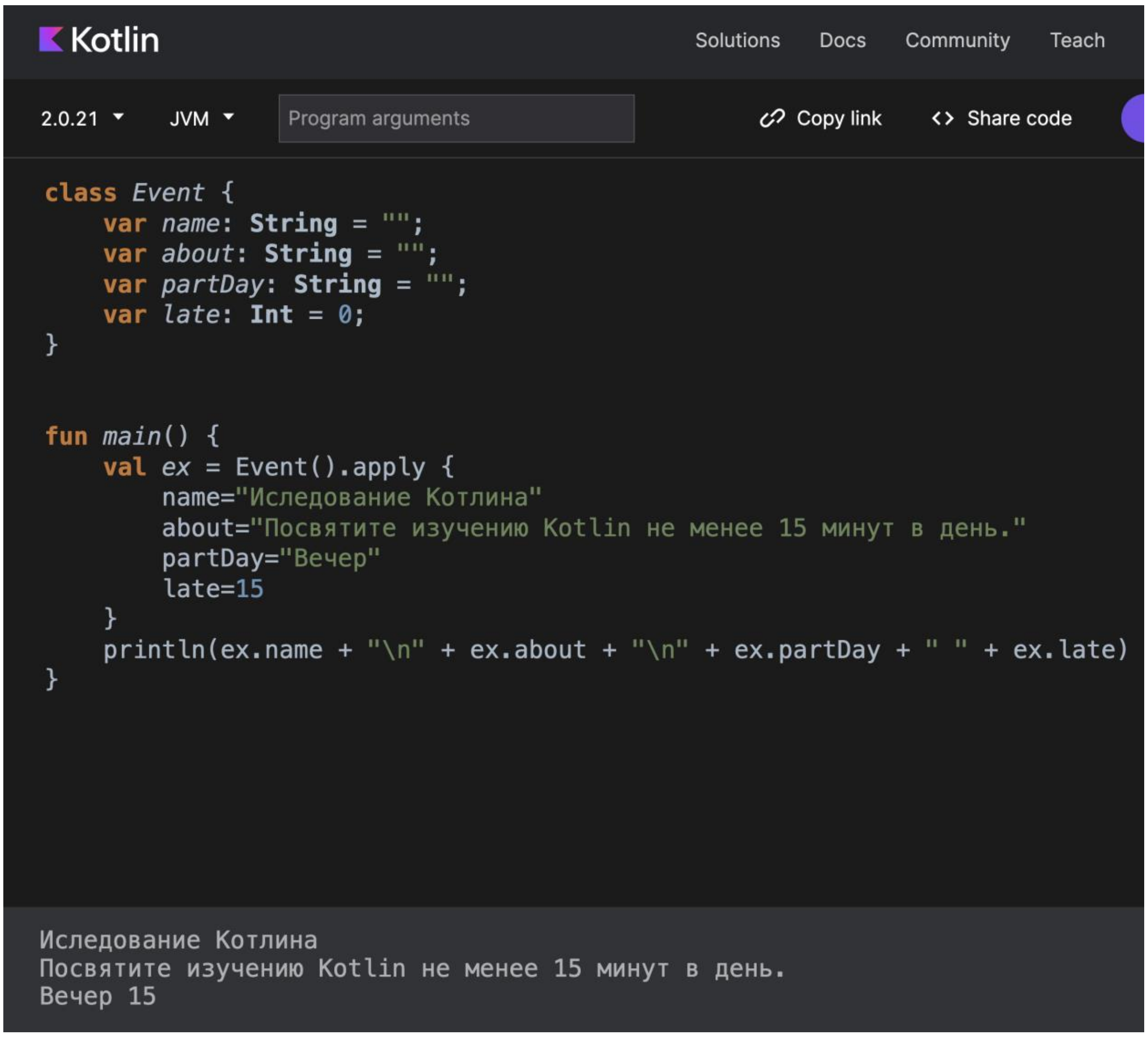


```kotlin
class Cookie(
    val name: String,
    val softBaked: Boolean,
    val hasFilling: Boolean,
    val price: Double
)
val cookies = listOf(
    Cookie(
        name = "Chocolate Chip",
        softBaked = false,
        hasFilling = false,
        price = 1.69
    ),
    Cookie(
        name = "Banana Walnut",
        softBaked = true,
        hasFilling = false,
        price = 1.49
    ),
    Cookie(
        name = "Vanilla Creme",
        softBaked = false,
        hasFilling = true,
        price = 1.59
    ),
    Cookie(
        name = "Chocolate Peanut Butter",
        softBaked = false,
        hasFilling = true,
        price = 1.49
    ),
    Cookie(
        name = "Snickerdoodle",
        softBaked = true,
        hasFilling = false,
        price = 1.39
    ),
    Cookie(
        name = "Blueberry Tart",
        softBaked = true,
        hasFilling = true,
        price = 1.79
    ),
    Cookie(
        name = "Sugar and Sprinkles",
        softBaked = false,
        hasFilling = false,
        price = 1.39
    )
)
fun main() {
    val alphabeticalMenu = cookies.sortedBy {
        it.name
    }
    println("Alphabetical menu:")
    alphabeticalMenu.forEach {
        println(it.name)
    }
}
```

```
Full menu:
Chocolate Chip - $1.69
Banana Walnut - $1.49
Vanilla Creme - $1.59
Chocolate Peanut Butter - $1.49
Snickerdoodle - $1.39
Blueberry Tart - $1.79
Sugar and Sprinkles - $1.39
```

Практическая работа №18

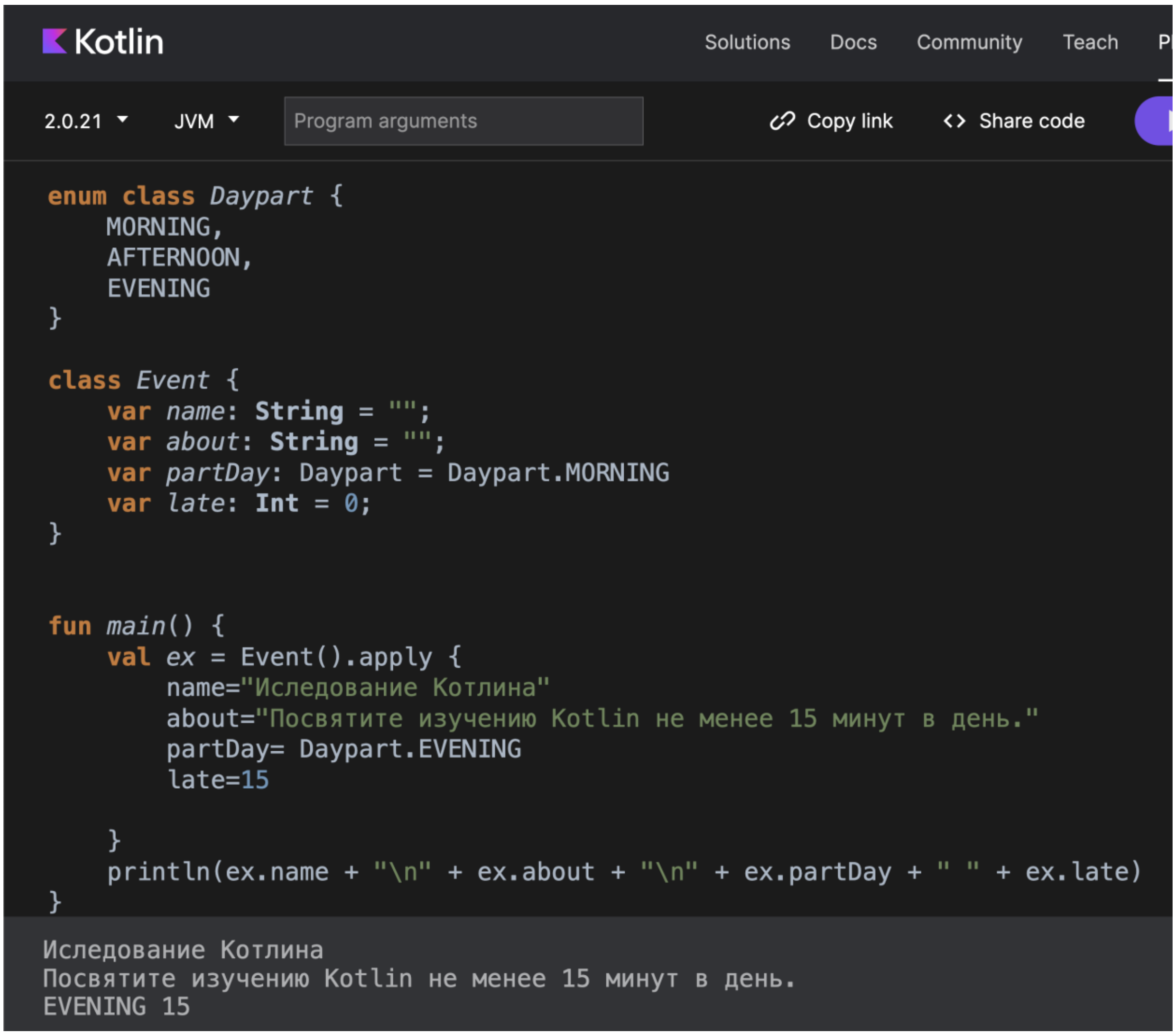
Задание 1.



```
class Event {
    var name: String = "";
    var about: String = "";
    var partDay: String = "";
    var late: Int = 0;
}


fun main() {
    val ex = Event().apply {
        name="Иследование Котлина"
        about="Посвятите изучению Kotlin не менее 15 минут в день."
        partDay="Вечер"
        late=15
    }
    println(ex.name + "\n" + ex.about + "\n" + ex.partDay + " " + ex.late)
}
```

```
Иследование Котлина
Посвятите изучению Kotlin не менее 15 минут в день.
Вечер 15
```

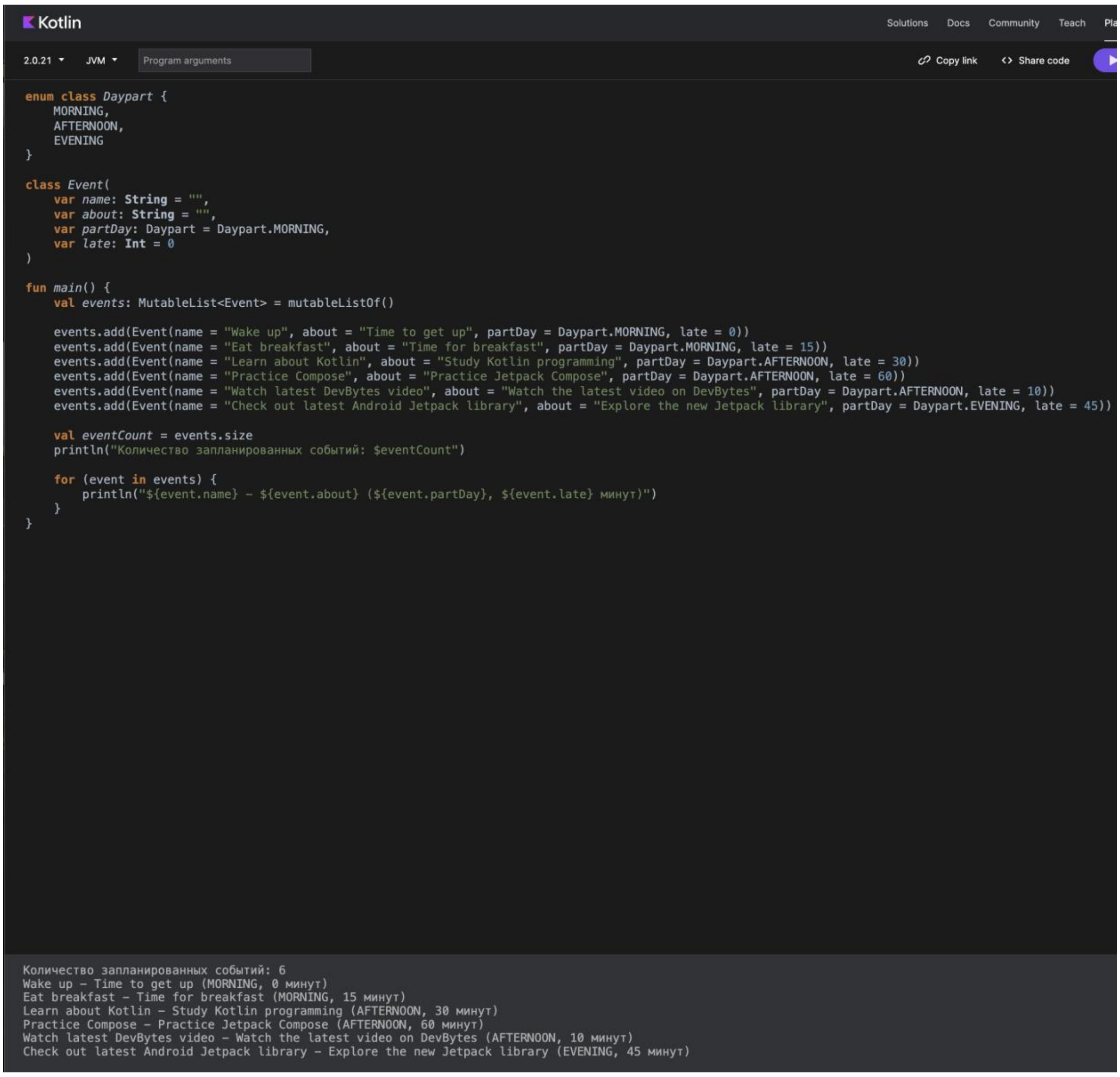
Задание 2.



```kotlin
enum class Daypart {
    MORNING,
    AFTERNOON,
    EVENING
}

class Event {
    var name: String = "";
    var about: String = "";
    var partDay: Daypart = Daypart.MORNING
    var late: Int = 0;
}


fun main() {
    val ex = Event().apply {
        name="Иследование Котлина"
        about="Посвятите изучению Kotlin не менее 15 минут в день."
        partDay= Daypart.EVENING
        late=15

    }
    println(ex.name + "\n" + ex.about + "\n" + ex.partDay + " " + ex.late)
}
```

```
Иследование Котлина
Посвятите изучению Kotlin не менее 15 минут в день.
EVENING 15
```

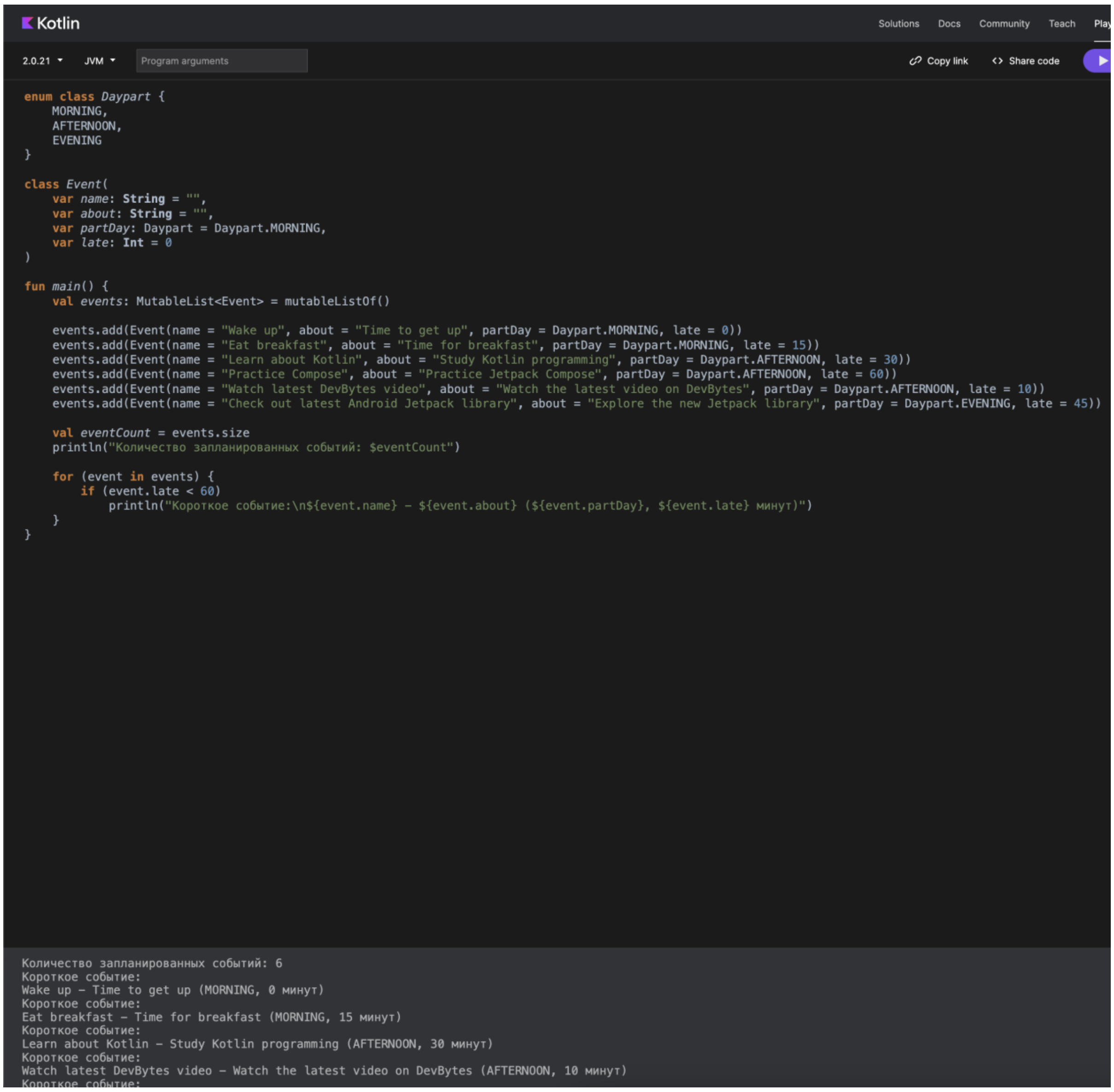
Задание 3.



```kotlin
enum class Daypart {
    MORNING,
    AFTERNOON,
    EVENING
}

class Event(
    var name: String = "",
    var about: String = "",
    var partDay: Daypart = Daypart.MORNING,
    var late: Int = 0
)

fun main() {
    val events: MutableList<Event> = mutableListOf()

    events.add(Event(name = "Wake up", about = "Time to get up", partDay = Daypart.MORNING, late = 0))
    events.add(Event(name = "Eat breakfast", about = "Time for breakfast", partDay = Daypart.MORNING, late = 15))
    events.add(Event(name = "Learn about Kotlin", about = "Study Kotlin programming", partDay = Daypart.AFTERNOON, late = 30))
    events.add(Event(name = "Practice Compose", about = "Practice Jetpack Compose", partDay = Daypart.AFTERNOON, late = 60))
    events.add(Event(name = "Watch latest DevBytes video", about = "Watch the latest video on DevBytes", partDay = Daypart.AFTERNOON, late = 10))
    events.add(Event(name = "Check out latest Android Jetpack library", about = "Explore the new Jetpack library", partDay = Daypart.EVENING, late = 45))

    val eventCount = events.size
    println("Количество запланированных событий: $eventCount")

    for (event in events) {
        println("${event.name} - ${event.about} (${event.partDay}, ${event.late} минут)")
    }
}
```

```
Количество запланированных событий: 6
Wake up - Time to get up (MORNING, 0 минут)
Eat breakfast - Time for breakfast (MORNING, 15 минут)
Learn about Kotlin - Study Kotlin programming (AFTERNOON, 30 минут)
Practice Compose - Practice Jetpack Compose (AFTERNOON, 60 минут)
Watch latest DevBytes video - Watch the latest video on DevBytes (AFTERNOON, 10 минут)
Check out latest Android Jetpack library - Explore the new Jetpack library (EVENING, 45 минут)
```

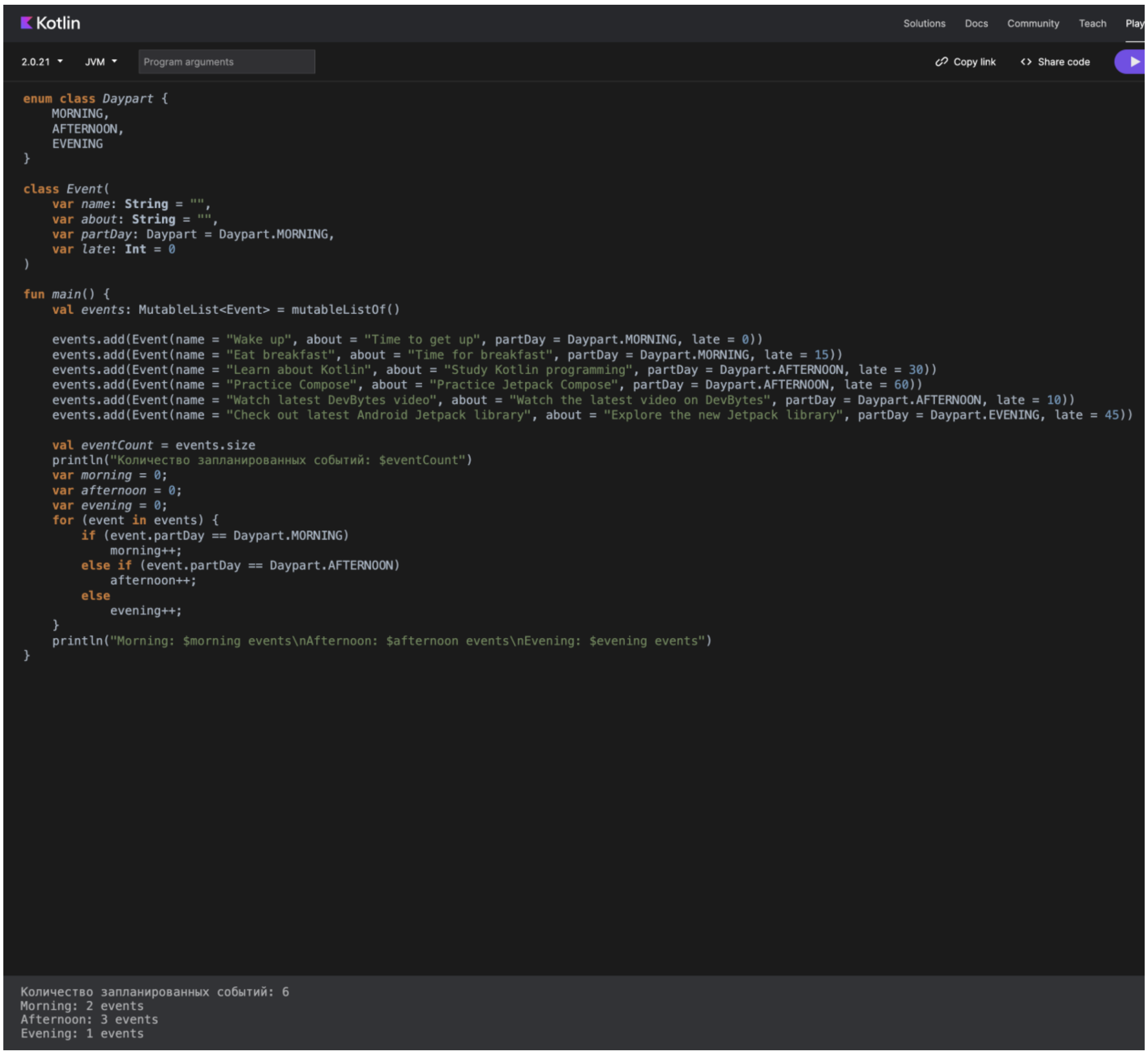
Задание 4.



```kotlin
enum class Daypart {
    MORNING,
    AFTERNOON,
    EVENING
}

class Event(
    var name: String = "",
    var about: String = "",
    var partDay: Daypart = Daypart.MORNING,
    var late: Int = 0
)

fun main() {
    val events: MutableList<Event> = mutableListOf()

    events.add(Event(name = "Wake up", about = "Time to get up", partDay = Daypart.MORNING, late = 0))
    events.add(Event(name = "Eat breakfast", about = "Time for breakfast", partDay = Daypart.MORNING, late = 15))
    events.add(Event(name = "Learn about Kotlin", about = "Study Kotlin programming", partDay = Daypart.AFTERNOON, late = 30))
    events.add(Event(name = "Practice Compose", about = "Practice Jetpack Compose", partDay = Daypart.AFTERNOON, late = 60))
    events.add(Event(name = "Watch latest DevBytes video", about = "Watch the latest video on DevBytes", partDay = Daypart.AFTERNOON, late = 10))
    events.add(Event(name = "Check out latest Android Jetpack library", about = "Explore the new Jetpack library", partDay = Daypart.EVENING, late = 45))

    val eventCount = events.size
    println("Количество запланированных событий: $eventCount")

    for (event in events) {
        if (event.late < 60)
            println("Короткое событие:\n${event.name} – ${event.about} (${event.partDay}, ${event.late} минут)")
    }
}
```

```
Количество запланированных событий: 6
Короткое событие:
Wake up – Time to get up (MORNING, 0 минут)
Короткое событие:
Eat breakfast – Time for breakfast (MORNING, 15 минут)
Короткое событие:
Learn about Kotlin – Study Kotlin programming (AFTERNOON, 30 минут)
Короткое событие:
Watch latest DevBytes video – Watch the latest video on DevBytes (AFTERNOON, 10 минут)
Короткое событие:
```

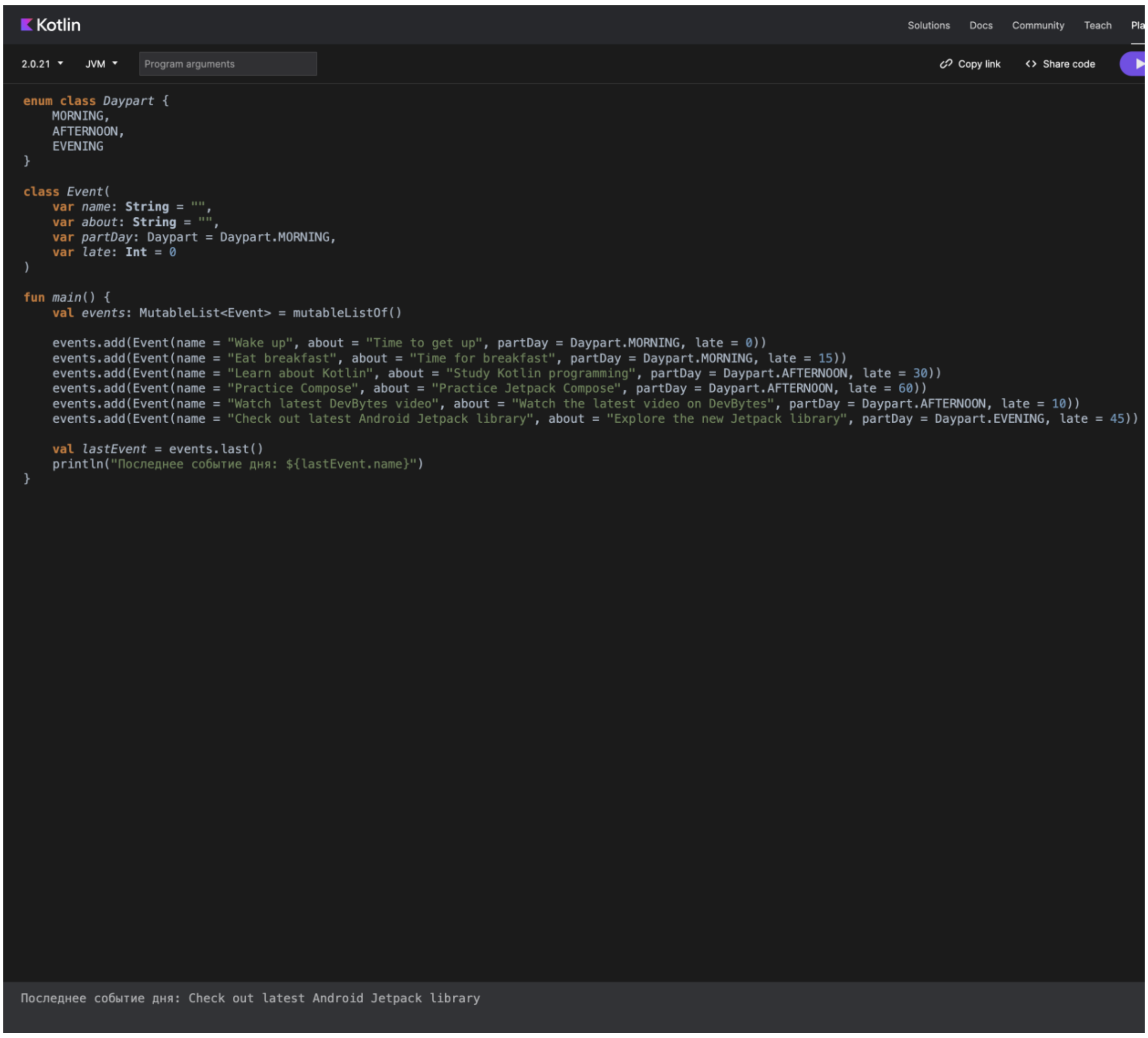
Задание 5.



```kotlin
enum class Daypart {
    MORNING,
    AFTERNOON,
    EVENING
}

class Event(
    var name: String = "",
    var about: String = "",
    var partDay: Daypart = Daypart.MORNING,
    var late: Int = 0
)

fun main() {
    val events: MutableList<Event> = mutableListOf()

    events.add(Event(name = "Wake up", about = "Time to get up", partDay = Daypart.MORNING, late = 0))
    events.add(Event(name = "Eat breakfast", about = "Time for breakfast", partDay = Daypart.MORNING, late = 15))
    events.add(Event(name = "Learn about Kotlin", about = "Study Kotlin programming", partDay = Daypart.AFTERNOON, late = 30))
    events.add(Event(name = "Practice Compose", about = "Practice Jetpack Compose", partDay = Daypart.AFTERNOON, late = 60))
    events.add(Event(name = "Watch latest DevBytes video", about = "Watch the latest video on DevBytes", partDay = Daypart.AFTERNOON, late = 10))
    events.add(Event(name = "Check out latest Android Jetpack library", about = "Explore the new Jetpack library", partDay = Daypart.EVENING, late = 45))

    val eventCount = events.size
    println("Количество запланированных событий: $eventCount")
    var morning = 0;
    var afternoon = 0;
    var evening = 0;
    for (event in events) {
        if (event.partDay == Daypart.MORNING)
            morning++;
        else if (event.partDay == Daypart.AFTERNOON)
            afternoon++;
        else
            evening++;
    }
    println("Morning: $morning events\nAfternoon: $afternoon events\nEvening: $evening events")
}
```

```
Количество запланированных событий: 6
Morning: 2 events
Afternoon: 3 events
Evening: 1 events
```

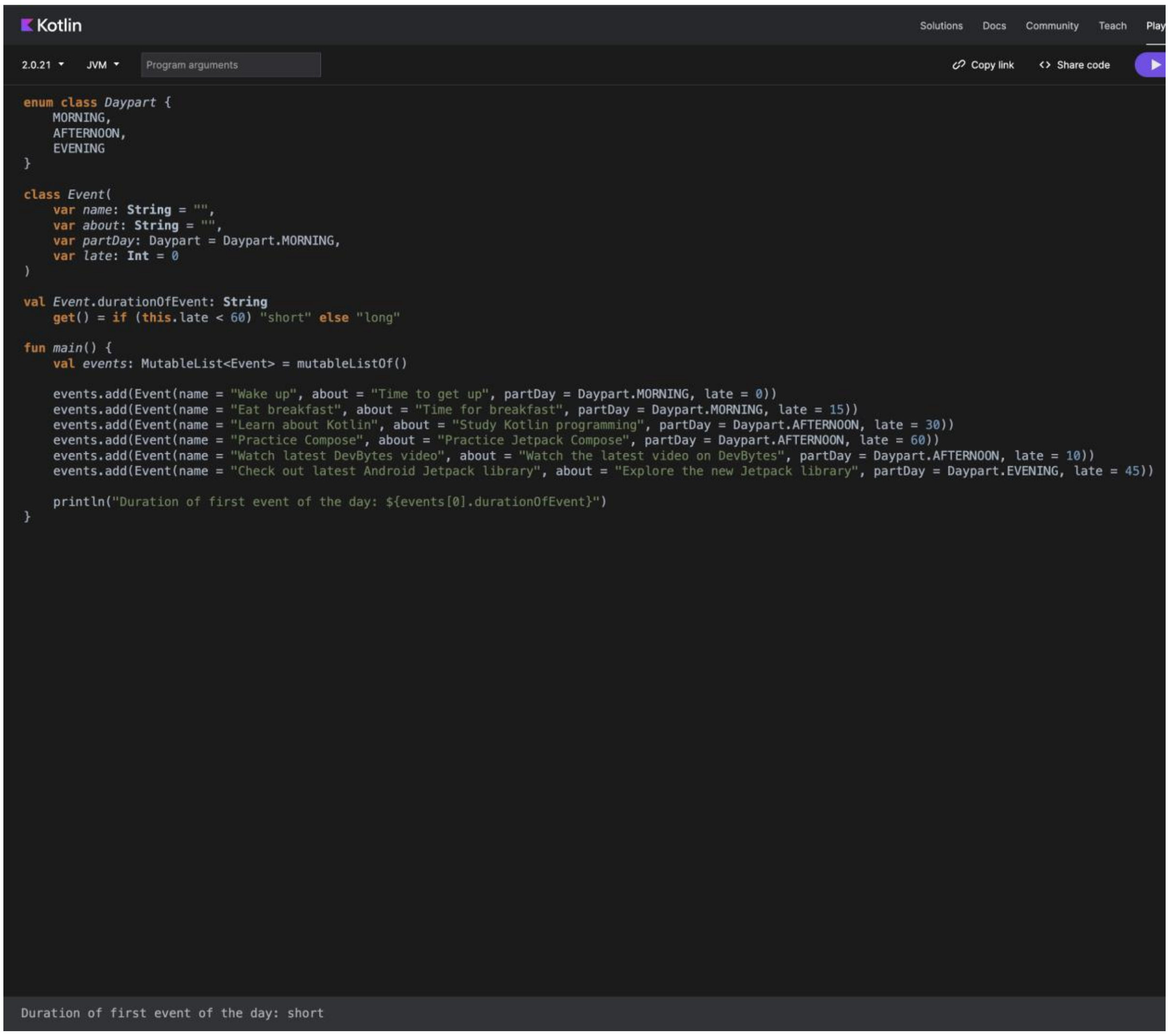
Задание 6.



```kotlin
enum class Daypart {
    MORNING,
    AFTERNOON,
    EVENING
}

class Event(
    var name: String = "",
    var about: String = "",
    var partDay: Daypart = Daypart.MORNING,
    var late: Int = 0
)

fun main() {
    val events: MutableList<Event> = mutableListOf()

    events.add(Event(name = "Wake up", about = "Time to get up", partDay = Daypart.MORNING, late = 0))
    events.add(Event(name = "Eat breakfast", about = "Time for breakfast", partDay = Daypart.MORNING, late = 15))
    events.add(Event(name = "Learn about Kotlin", about = "Study Kotlin programming", partDay = Daypart.AFTERNOON, late = 30))
    events.add(Event(name = "Practice Compose", about = "Practice Jetpack Compose", partDay = Daypart.AFTERNOON, late = 60))
    events.add(Event(name = "Watch latest DevBytes video", about = "Watch the latest video on DevBytes", partDay = Daypart.AFTERNOON, late = 10))
    events.add(Event(name = "Check out latest Android Jetpack library", about = "Explore the new Jetpack library", partDay = Daypart.EVENING, late = 45))

    val lastEvent = events.last()
    println("Последнее событие дня: ${lastEvent.name}")
}
```

```
Последнее событие дня: Check out latest Android Jetpack library
```

Задание 7.



```kotlin
enum class Daypart {
    MORNING,
    AFTERNOON,
    EVENING
}

class Event(
    var name: String = "",
    var about: String = "",
    var partDay: Daypart = Daypart.MORNING,
    var late: Int = 0
)

val Event.durationOfEvent: String
    get() = if (this.late < 60) "short" else "long"

fun main() {
    val events: MutableList<Event> = mutableListOf()

    events.add(Event(name = "Wake up", about = "Time to get up", partDay = Daypart.MORNING, late = 0))
    events.add(Event(name = "Eat breakfast", about = "Time for breakfast", partDay = Daypart.MORNING, late = 15))
    events.add(Event(name = "Learn about Kotlin", about = "Study Kotlin programming", partDay = Daypart.AFTERNOON, late = 30))
    events.add(Event(name = "Practice Compose", about = "Practice Jetpack Compose", partDay = Daypart.AFTERNOON, late = 60))
    events.add(Event(name = "Watch latest DevBytes video", about = "Watch the latest video on DevBytes", partDay = Daypart.AFTERNOON, late = 10))
    events.add(Event(name = "Check out latest Android Jetpack library", about = "Explore the new Jetpack library", partDay = Daypart.EVENING, late = 45))

    println("Duration of first event of the day: ${events[0].durationOfEvent}")
}
```

```
Duration of first event of the day: short
```