Step Fulfilment Document

Tuesday, 11 March 2025 14:59

By: Grace Shuuya 220109702

- Downloaded Repo from provided link as a ZIP file
- Extracted "Homework 1" into **C:\Users\p.shuuya\AndroidStudioProjects** , opened Android Studio and opened the homework folder from there.
- created my GitHub Repository at https://github.com/Kahewa/Alias-Game-API
- Followed the steps from the "README.md" from the homework folder as follows:
- After Successfully completing the steps, I cloned my repository to Github Desktop and used vscode to push my work onto my repository.

1. Create a type alias `Identifier`, an `IdentifierFactory` class, and a `uniqueIdentifier` function in the `jetbrains.kotlin.course.alias.util` package.

- The type alias `Identifier` needs to be an alias for the `Int` type. If you change the type in the future, e.g. create a new class, it will be changed automatically in all places.
- The `IdentifierFactory` class is a class for generating unique identifiers, e.g. identifiers for different game cards or teams. It should have a special `counter` – an `Int` property for storing the last unique number. By default, `counter` should be zero.
It should have a special `counter` – an `Int` property to store the last unique number. By default, `counter` should be zero.
- The `uniqueIdentifier` function returns a new unique identifier by incrementing the `counter` and returning it.

Step Fulfilment:

M↓ README.md IdentifierGr23.kt ×

```kotlin
package jetbrains.kotlin.course.alias.util

typealias Identifier = Int
//creates a shortcut for the type "Int" and call it "identifier."
//makes it easier to understand that we're working with IDs.

class IdentifierFactory { //his class is like a machine that creates unique ID numbers.
    var counter: Int = 0 //a counter keeps track of the last number used. It starts at 0.
        private set //this means only the class itself can change the counter.
    fun generateUniqueIdentifier(): Identifier {  //this function creates a new unique ID.
        return ++counter //increases the counter by 1 and returns the new number.
    }
}

fun uniqueIdentifier(factory: IdentifierFactory): Identifier { // uses the IdentifierFactory to get a new unique ID.
    return factory.generateUniqueIdentifier() //asks the factory to generate and return a new ID.
}
```

2. Create a data class `Team` in the `jetbrains.kotlin.course.alias.team` package to store the information about teams.

- It must have two properties in the primary constructor: `id` of `Identifier` type to identify each team and `points` of `Int` type to store the number of points in the game. For points, set the default value `0`.
- It must have an additional property `name`, which initializes automatically as `"Team#${id + 1}"` and will be shown in the leaderboard.

Step Fulfilment:

M↓ README.md TeamGr23.kt ×

```kotlin
package jetbrains.kotlin.course.alias.team

import jetbrains.kotlin.course.alias.util.Identifier
//imports the Identifier type from the jetbrains.kotlin.course.alias.util package


data class Team( //this is a data class to store information about a team.
    val id: Identifier, //unique ID for the team
    var points: Int = 0, //the team's points, starting at 0 by default
) {
    //this property creates a team name like "Team#1", "Team#2", etc.
    val name: String = "Team#${id + 1}"
}
```

3. The package `jetbrains.kotlin.course.alias.team` already has a regular class `TeamService`. It is responsible for the game logic for the teams. In this task, you need to implement several things to bring the game to life.
- Add a property `identifierFactory` with the type `IdentifierFactory` to generate identifiers for each team. Don't forget to add the default value for it by creating a new instance of the `IdentifierFactory` class.
- Add a companion object to the `TeamService` class and declare the `teamsStorage` variable to store all previous teams. The storage type should be `MutableMap`, which maps `Identifier` to `Team`. Don't forget to initialize it via an empty map.
- Implement the `generateTeamsForOneRound` method. The method must generate a list of teams and also store all of them in the `teamsStorage` map. To create new teams you need to use `identifierFactory` from the `TeamService` class to generate a new ID. We need to create this method to save game results for the leaderboard.

Step Fulfilment:

README.md × TeamService.kt ×

```kotlin
import jetbrains.kotlin.course.alias.util.Identifier
import jetbrains.kotlin.course.alias.util.IdentifierFactory
import org.springframework.stereotype.Service

@Service
class TeamService {
    //a factory to generate unique IDs for teams
    val identifierFactory: IdentifierFactory = IdentifierFactory()

    //a companion object to store all teams in one place
    companion object {
        //a map to store teams, using their IDs as keys
        val teamsStorage: MutableMap<Identifier, Team> = mutableMapOf()
    }

    //this method creates a list of teams for one round of the game.
    fun generateTeamsForOneRound(teamsNumber: Int): List<Team> { //the line that was here before implementation
        val teams = mutableListOf<Team>()  //creates an empty list to store the new teams

        //this is a loop to create the specified number of teams
        for (i in 1 ≤ .. ≤ teamsNumber) {
            //generates a unique ID for the team
            val teamId = identifierFactory.generateUniqueIdentifier()
            //creates a new team with the generated ID
            val team = Team(id = teamId)
            //adds the team to the list
            teams.add(team)
            //stores the team in the teamsStorage map for future reference
            teamsStorage[teamId] = team
        }

        return teams //returns the list of created teams
```

4. Create two classes to work with the cards in the `jetbrains.kotlin.course.alias.card` package.

- A value class `Word` with one `String` `word` property to store a word.
- A data class `Card` to store information for each card. Each card must store an `id` with the `Identifier` type and a list of `words` (`List<Word>`). These properties don't have default values and must also be defined in the primary constructor.

<u>Step Fulfilment:</u>

```
package jetbrains.kotlin.course.alias.card

import jetbrains.kotlin.course.alias.util.Identifier

// this is a value class to store a single word. It is optimized
// for performance and avoids unnecessary object creation.

@JvmInline
value class Word(val word: String)

//data class to store information about a card.
data class Card(
    val id: Identifier, //unique ID for the card
    val words: List<Word> //list of words on the card
)
```

5. The package `jetbrains.kotlin.course.alias.card` already has the regular class `CardService`. You need to add several properties and implement several methods.

- Add a property `identifierFactory` with the type `IdentifierFactory` to generate identifiers for each card. Don't forget to add the default value for it by creating a new instance of the `IdentifierFactory` class.
- Add a property `cards` that stores a list of cards (`List<Card>`). Initialize it by calling the `generateCards` method.
- Add a companion object to the `CardService` class and declare the `WORDS_IN_CARD` const variable to store the number of words for the cards. You need to assign the value `4` to it. Also, declare `cardsAmount` here, which stores the possible number of cards: `words.size / WORDS_IN_CARD`. The project contains a predefined list of words called `words`.
- Add the `toWords` function to the `CardService` class, which is an extension function for `List<String>` and converts each element from this list into `Word`.
- Implement the `generateCards` function, which shuffles the `words` list, splits it into chunks with `WORDS_IN_CARD` words, takes `cardsAmount` chunks for `cardsAmount` cards, and finally creates a new `Card` for each chunk.
- Implement the `getCardByIndex` method, which accepts `index` (an integer number) and the `Card` at this index. If the card does not exist, throw an error.

Step Fulfilment:

README.md CardService.kt

```kotlin
package jetbrains.kotlin.course.alias.card
import jetbrains.kotlin.course.alias.util.IdentifierFactory
import org.springframework.stereotype.Service


@Service
class CardService {
    //a factory to generate unique IDs for cards
    private val identifierFactory: IdentifierFactory = IdentifierFactory()

    //a list of cards, initialized by calling generateCards()
    private val cards: List<Card> = generateCards()

    // Companion object to store constants and shared properties
    companion object {
        //number of words per card
        const val WORDS_IN_CARD: Int = 4

        //predefined list of words (made up to fit 5 cards)
        val words: List<String> = listOf(
            "Apple", "Banana", "Orange", "Grape", "Salad", "Boerewors", "Braai", "Burger", "Matthew",
            "Mark", "Luke", "John", "Android", "Apple", "Linux", "Windows", "Pink",
            "Blue", "Red", "Purple"
        )

        //number of cards based on the number of words and WORDS_IN_CARD
        val cardsAmount: Int = words.size / WORDS_IN_CARD
    }

    //extension function to convert List<String> to List<Word>
    private fun List<String>.toWords(): List<Word> = this.map { Word(it) }

```

6. Implement several things in the already defined class `GameResultsService` in the `jetbrains.kotlin.course.alias.results` package.

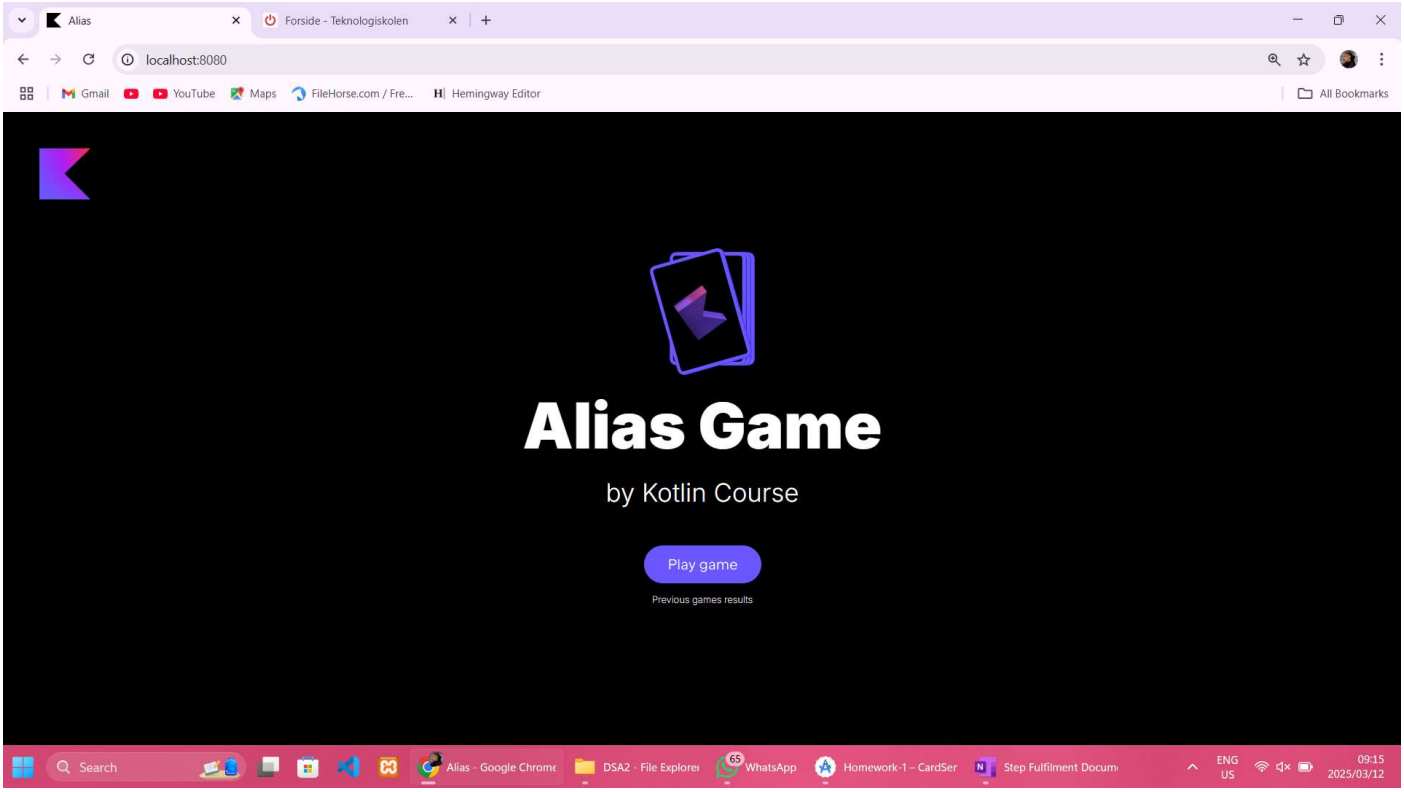
- Add a type alias `GameResult` referring to `List<Team>` to the `jetbrains.kotlin.course.alias.results` package.
- Add a companion object to the `GameResultsService` and declare the `gameHistory` variable for storing the list of game results (`MutableList<GameResult>`). By default, it must be initialized via an empty list.
- Implement the `saveGameResults` method that adds the `result` to the `gameHistory`. Before adding the `result` you need to check two requirements and throw an error if they are broken: `result` must be not empty and all team IDs from the `result` must be in the `TeamService.teamsStorage`.
- Implement the `getAllGameResults` method that returns the reversed `gameHistory` list.

<u>Step Fulfilment:</u>

```kotlin
package jetbrains.kotlin.course.alias.results
import jetbrains.kotlin.course.alias.team.Team
import jetbrains.kotlin.course.alias.team.TeamService
import org.springframework.stereotype.Service

// Type alias for GameResult (a list of teams)
typealias GameResult = List<Team>

@Service
class GameResultsService {
    // Companion object to store game history
    companion object {
        // Mutable list to store game results
        val gameHistory: MutableList<GameResult> = mutableListOf()
    }

    // Function to save game results
    fun saveGameResults(result: GameResult) {
        // Check if the result is not empty
        require(result.isNotEmpty()) { "Game result cannot be empty." }

        // Check if all team IDs in the result are valid (exist in TeamService.teamsStorage)
        val invalidTeams = result.filter { team ->
            team.id !in TeamService.teamsStorage.keys
        }
        require(invalidTeams.isEmpty()) {
            "Invalid teams found in the result: ${invalidTeams.joinToString { it.name }}"
        }

        // Add the result to the game history
        gameHistory.add(result)
    }
```

At the end of this part you will have the following application:
![**The Alias game**](*./utils/src/main/resources/images/states/alias/state2.gif*)

Step Fulfilment:


Alias
Forside - Teknologiskolen
localhost:8080
Gmail
YouTube
Maps
FileHorse.com / Fre...
Hemingway Editor
All Bookmarks
Alias Game
by Kotlin Course
Play game
Previous games results
Search
Alias - Google Chrome
DSA2 - File Explorer
WhatsApp
Homework-1 – CardSer
Step Fulfilment Docum
ENG
US
09:15
2025/03/12