SULEIMAN DAHIRU

FCP/CSE/22/2002

MOBILE APPLICATIN DEVELOPMENT

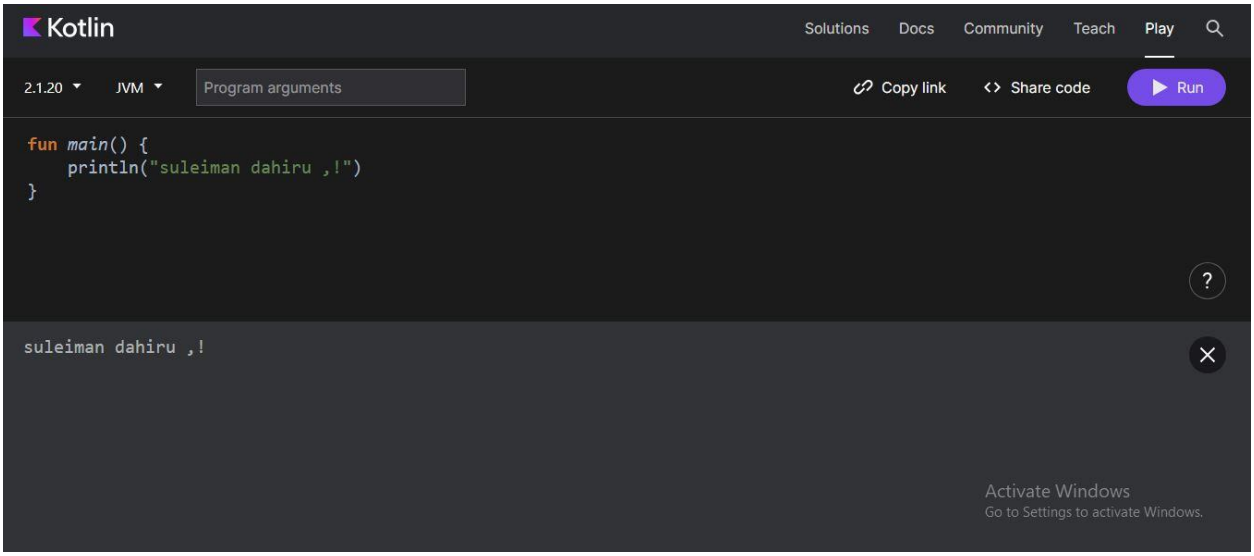
ASSIGMENT

PRACTICAL PART

Task 1: Hello World

Write a Kotlin program that prints Hello, [Your Name]! to the console.

Expected Output: Hello, Suleiman dahiru,



Explanation:

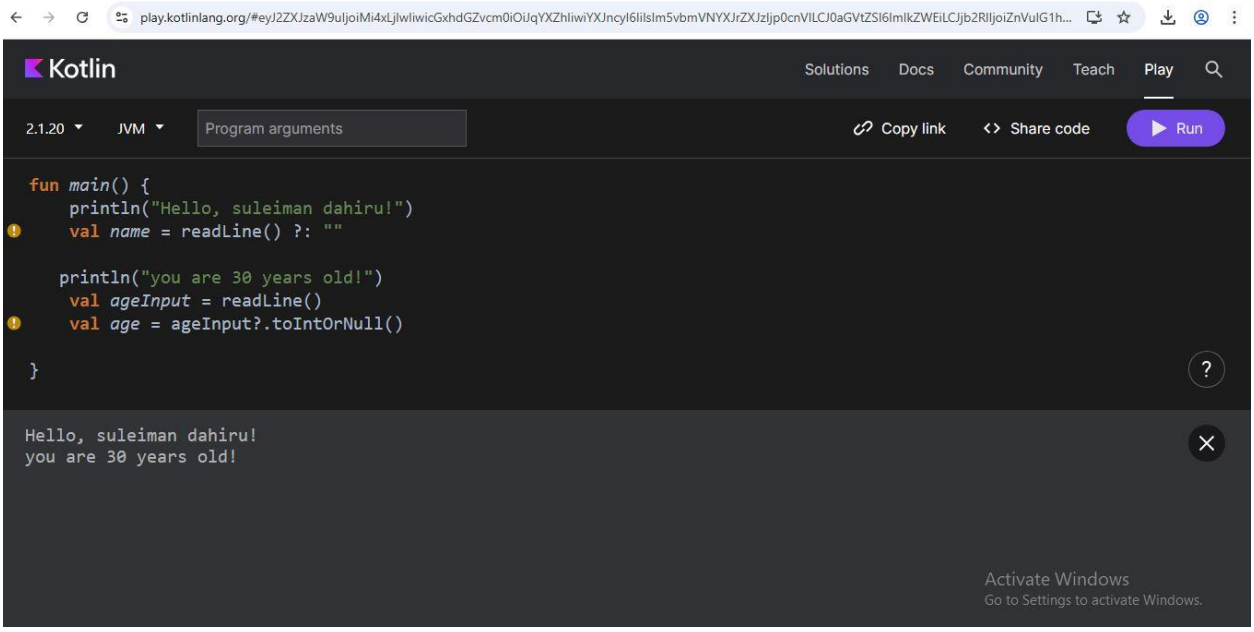
fun main() defines the main function where the program starts executing.

println() prints the text inside the parentheses to the console, followed by a newline.

Task 2: User Input

Create a Kotlin program that asks the user to input their name and age, then prints a greeting like:

Hello Suleiman dahiru!, you are 30 years old!



Explanation:

- **The program prompts the user to enter their name.**
- **It then asks for the age and tries to convert it to an integer.**
- **If the age is valid, it prints: "Hello, Mustapha Lawal Daura, you are 21 years old!**
- **If the age is invalid, it notifies the user.**
- **readLine() reads user input from the console.**
- **toIntOrNull() safely converts the string to an integer (returns null if it fails).**

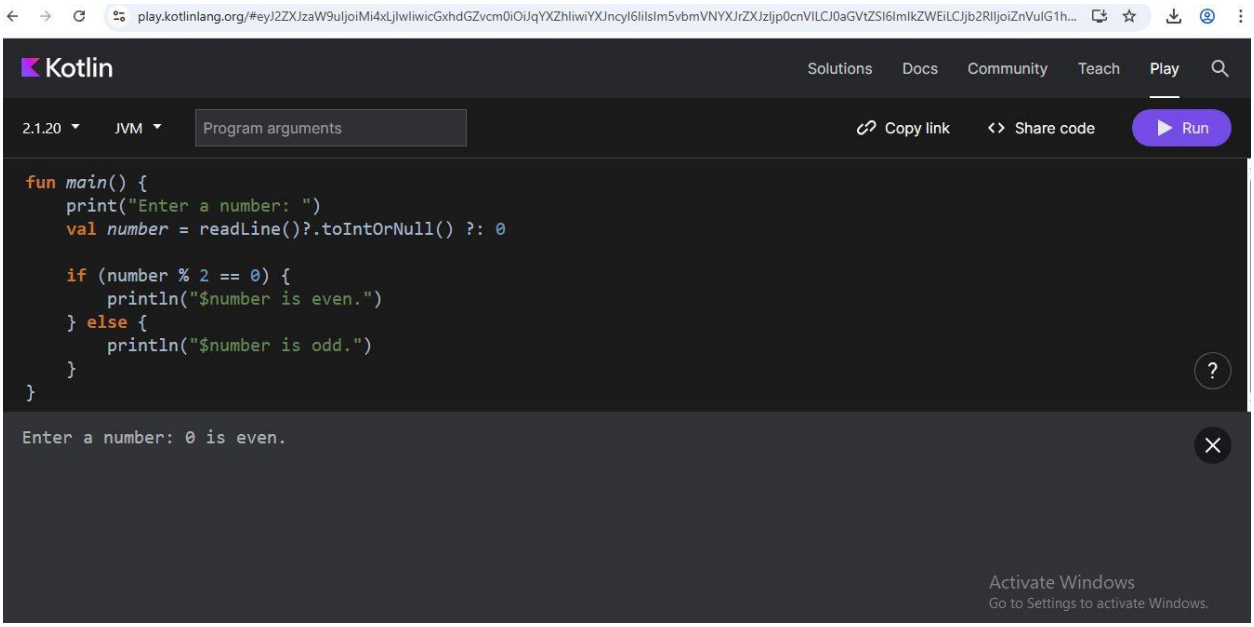
Task 3: Conditional Statements

Write a program that checks if a given number is even or odd.

Sample Output:

Enter a number: 10

10 is an even number

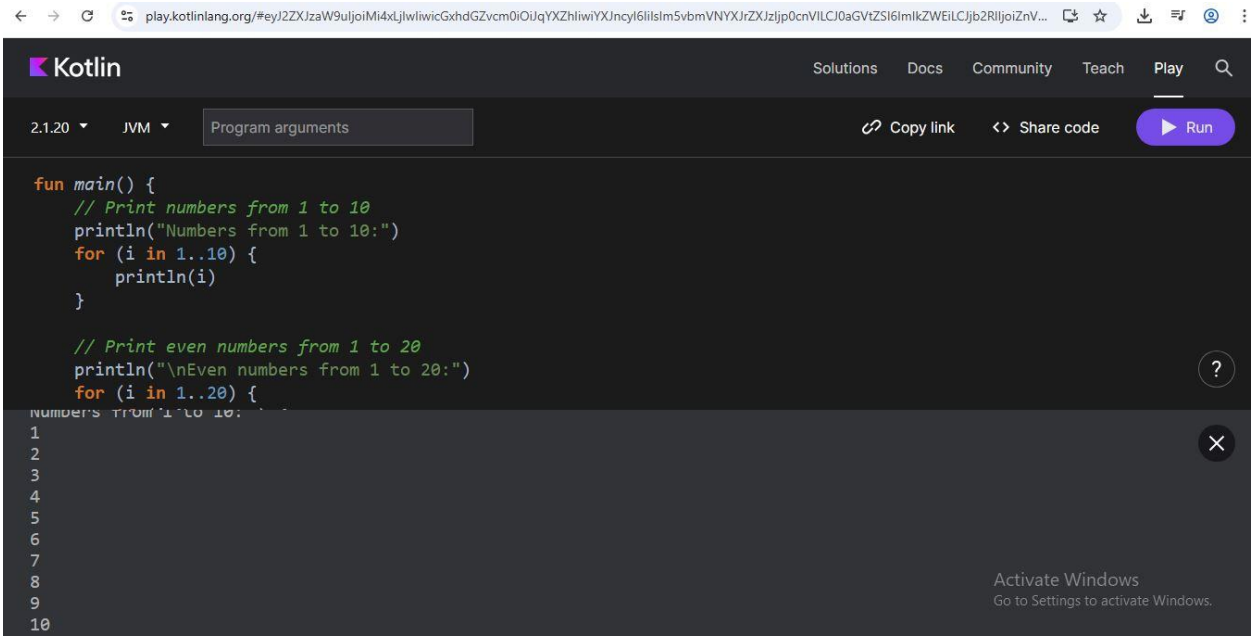


Explanation:

- **The program asks the user to input a number using readLine().**
- **It attempts to convert the input to an integer using toIntOrNull(). If the input is not a valid integer, it defaults to 0.**
- **It checks if the number is even by using the modulo operator (%). If the remainder of the division by 2 is 0, the number is even.**
- **It prints whether the number is even or odd.**

Task 4: Loops and Ranges

Print numbers from 1 to 10 using a for loop. Then, print only even numbers from 1 to 20

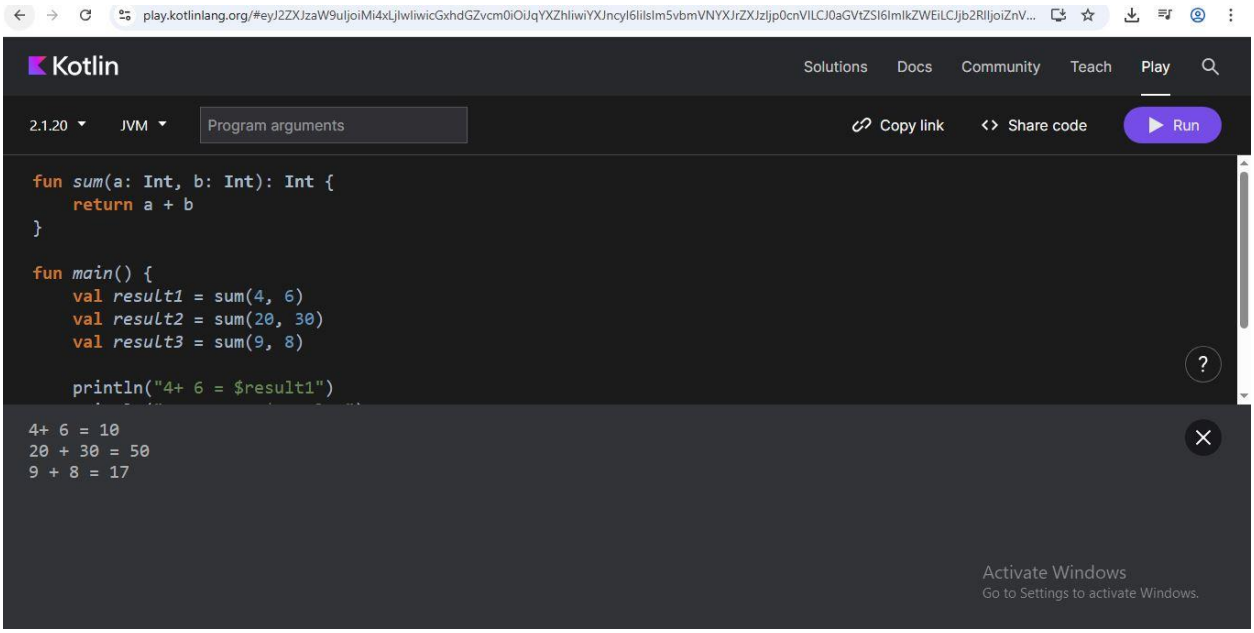


Explanation:

- **The first loop iterates from 1 to 10 using the .. operator and prints each number.**
- **The second loop iterates from 1 to 20 and checks if each number is even using the modulo operator (%). If the number is even, it prints the number.**

Task 5: Functions

Write a function sum(a: Int, b: Int): Int that returns the sum of two numbers. Call it with different values and display the result.



Explanation:

- **The sum function takes two Int parameters, a and b, and returns their sum.**
- **In the main function, we call sum with different values and store the results variables.**
- **We then print the results using string templates.**

Task 6: Arrays

Create an array of 5 names. Loop through the array and print each name with a greeting.

Sample Output:

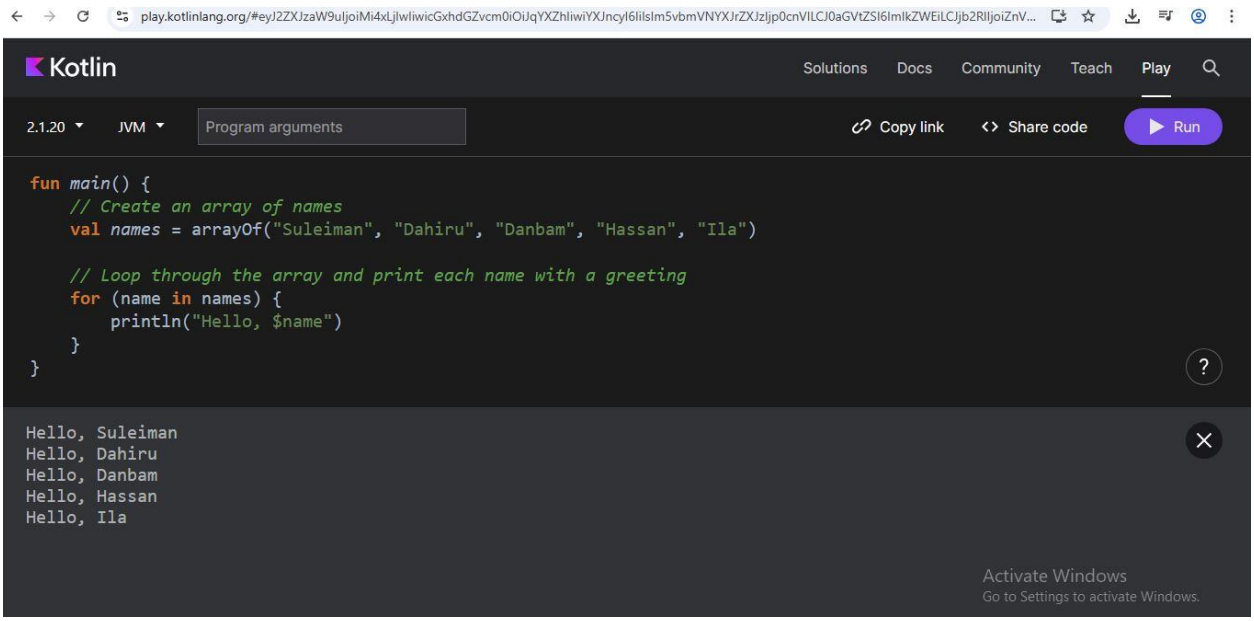
Hello, suleiman

Hello, dahiru

Hello, dambam

Hello, hassan

Hello, ila

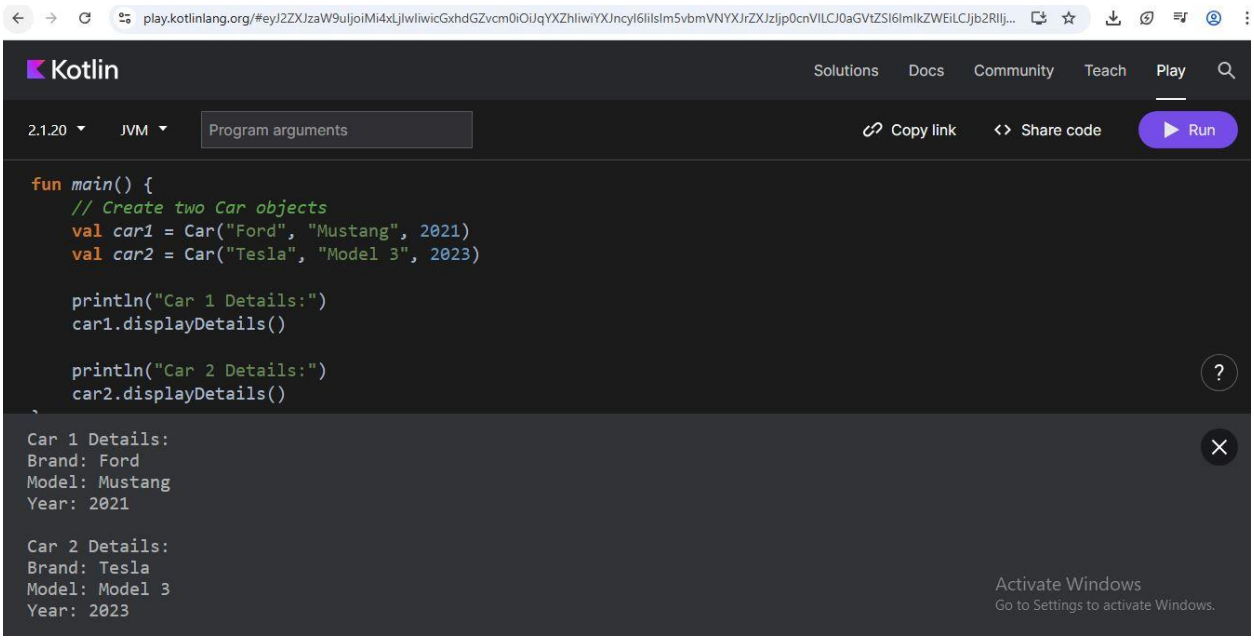


Explanation:

- **Create an array of names**
- **Loop through the array and print each name with a greeting**

Task 7: Classes and Objects

Define a class Car with properties brand, model, and year. Add a function displayDetails() that prints the car details. Create at least two objects of this class

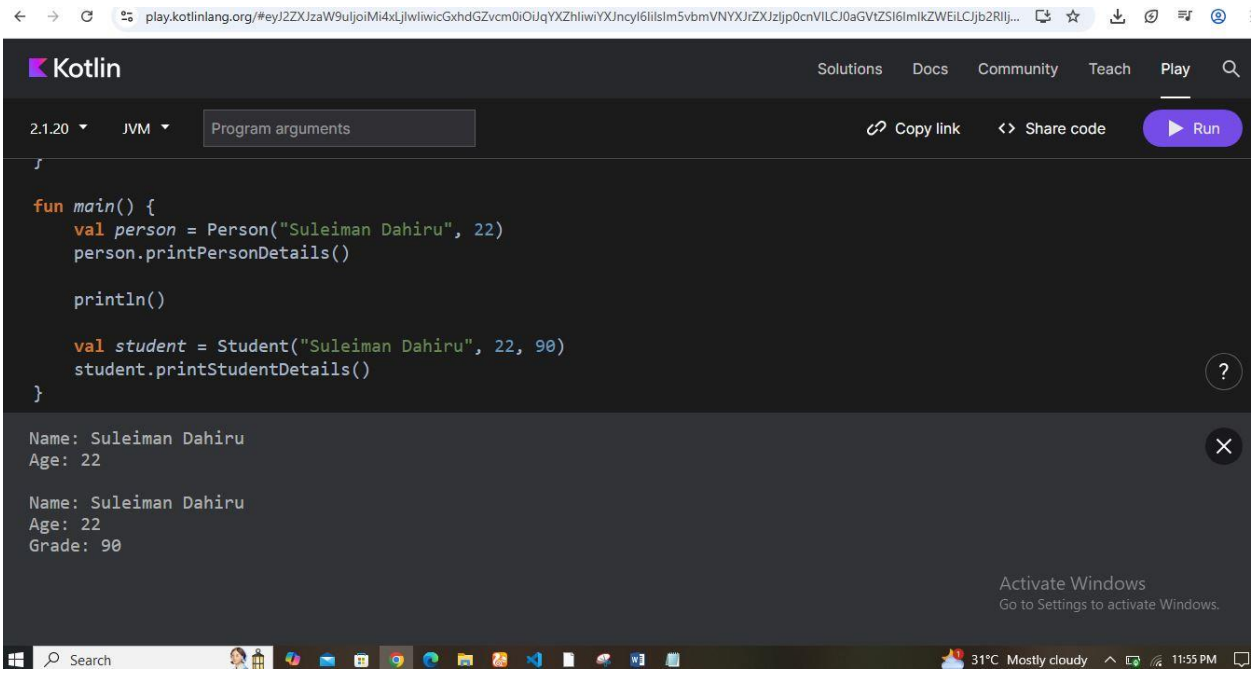


Explanation:

- **We define a Car class with properties brand, model, and year.**
- **The displayDetails function prints the car's details.**
- **In the main function, we create two Car objects, car1 and car2.**
- **We call the displayDetails function on each object to print their details.**

Task 8: Inheritance

Create a base class Person with properties name and age. Create a subclass Student that adds a property grade. Add methods to print each detail.

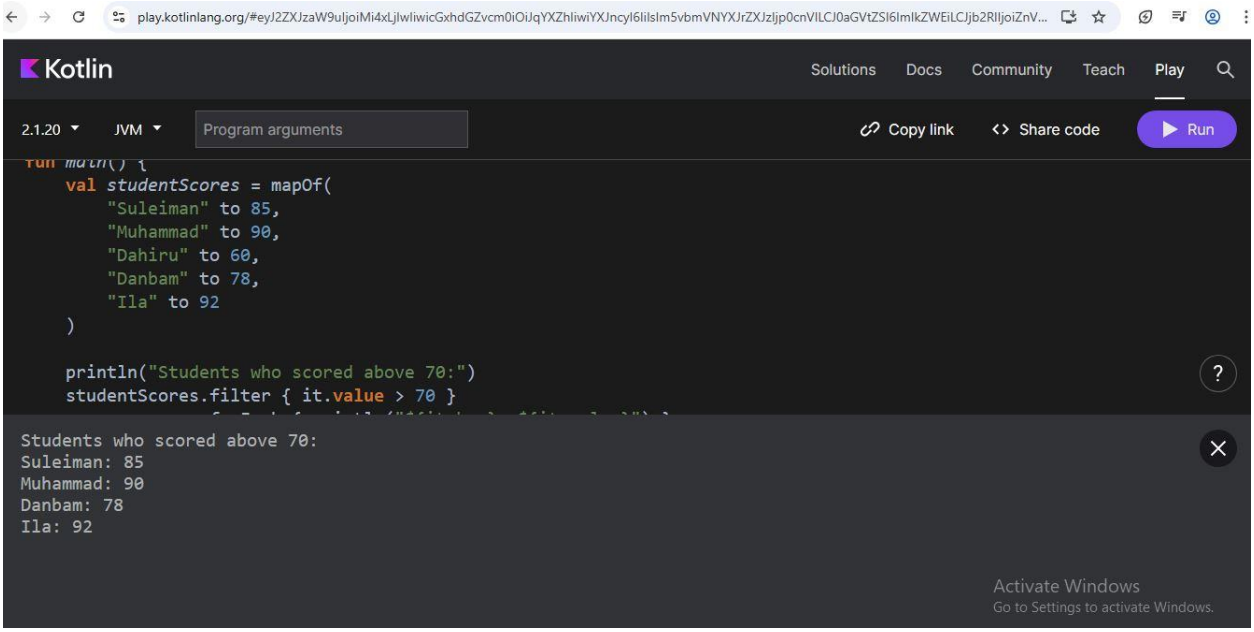


Explanation:

- **We define a base class Person with properties name and age, and a method printDetails to print these properties. The open keyword is used to allow inheritance.**
- **The Student class is a subclass of Person and adds a grade property. It also defines a printStudentDetails method to print all properties, including grade.**
- **In the Student class, we use super to access the properties and methods of the Person class.**
- **We override the toString method to provide a string representation of the Student object, including the grade property.**
- **In the example usage section, we create instances of Person and Student, and demonstrate how to print their details using the printDetails and printStudentDetails methods.**

Task 9: Collections and Map

Create a map with student names as keys and their scores as values. Print students who scored above 70.

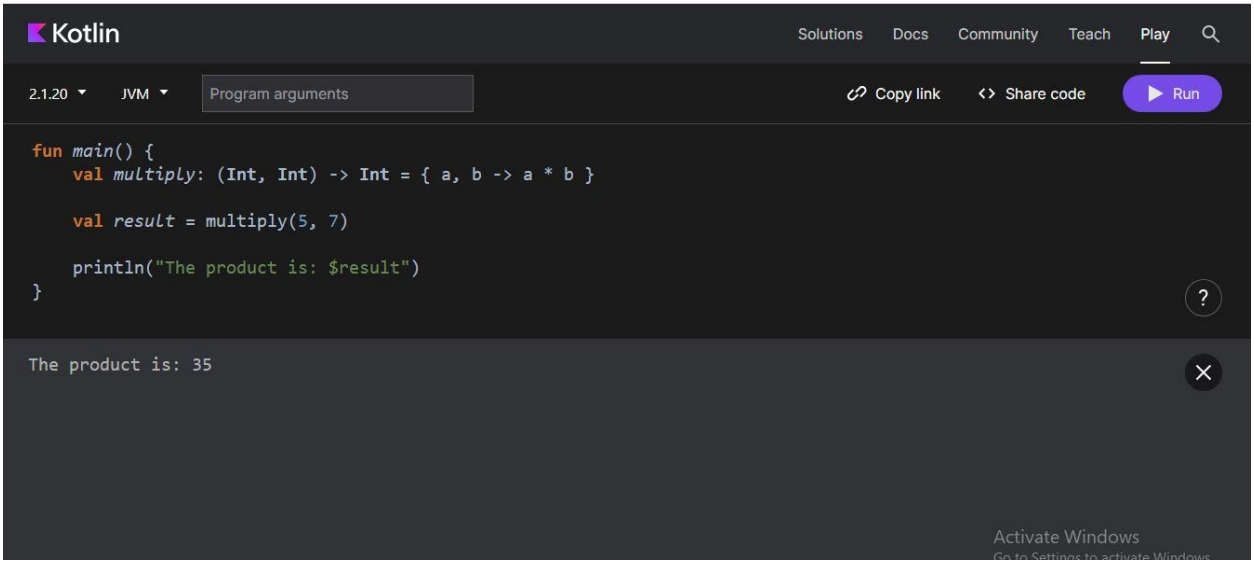


Explanation:

- **This creates an immutable map (mapOf) named studentScores.**
- **The map contains key-value pairs where the key is the student's name (a string) and the value is their score (an integer).**
- **The to keyword is used to create these key-value pairs.**
- **This uses the filter function to create a new map that only includes the students who scored above 70.**
- **The lambda expression { it.value > 70 } is the condition for filtering. it refers to each key-value pair in the map, and it.value refers to the score.**
- **The resulting map will only include the pairs where the score is greater than 70.**
- **This uses the forEach function to iterate over the filtered map and print each student's name and score.**

- **The lambda expression { println("${it.key}: ${it.value}") } is executed for each pair in the filtered map.**
- **${it.key} refers to the student's name, and ${it.value} refers to their score.**

Task 10: Lambda Expression Write a lambda expression that takes two integers and returns their product. Call it and print the result.



Explanation:

- **(Int, Int) -> Int defines the type of the lambda expression, which takes two integers as input and returns an integer.**
- **{ a, b -> a * b } is the lambda expression itself, where a and b are the input parameters, and a * b is the expression that gets evaluated and returned.**
- **multiply(5, 7) calls the lambda expression with the arguments 5 and 7, and the result is stored in the result variable.**
- **Finally, the result is printed to the console.**