Exercise Kotlin
Instructions for Students:

1. Open Kotlin Playground.
2. Clear the editor.
3. Complete the tasks below in order.

Exercise 1: The Nullable Bio (Variables)

Goal: Understand `val`, `var`, and Null Safety.

- **Task:**
    1. Create a `val` for your `firstName` (String).
    2. Create a nullable `var` for `middleName` (String?). Set it to `null`.
    3. Create a `val` for `age` (Int).
    4. Print a single string using **String Templates** (`$`) that says: "name: [First] [Middle] Age: [Age]".
    5. *Bonus:* Use the Elvis operator (`?:`) so that if `middleName` is null, it prints "N/A" instead of "null".

Exercise 2: Notification Priority (Control Flow)

Goal: Master the `when` expression.

- **Task:**
    1. Create a variable `priority` (Int). Set it to a number between 1 and 4.
    2. Write a `when` expression that prints a message based on the number:
        - 1 -> "Red: Critical Error"
        - 2 -> "Orange: Warning"
        - 3 -> "Yellow: Info"
        - Anything else -> "Green: System Normal"
    3. Change the `priority` value to test different outputs.

Exercise 3: The Discount Calculator (Functions)

Goal: Use functions and default arguments.

- **Task:**
    1. Write a function called `calculatePrice`.
    2. It takes two arguments: `price` (Double) and `discountPercent` (Int).
    3. Give `discountPercent` a **default value** of `0`.
    4. The function should return the final price.
    5. Call the function twice in `main()`:
        - Once with a discount (e.g., `100.0, 20`).

- Once without a discount (e.g., `50.0`).

Exercise 4: The Song Library (Data Classes)

Goal: Model data and use class methods.

- **Task:**
    1. Create a `class` named `Song` with properties: `title` (String), `artist` (String), and `playCount` (Int).
    2. Inside the class, add a function `isPopular()` that returns `true` if `playCount` is greater than 1000.
    3. In `main()`, create a specific song instance.
    4. Print the song info and whether it is popular or not.

Exercise 5: The Shopping Cart (Collections & Lambdas)

Goal: Use Lists, `filter`, and `map`.

- **Task:**
    1. Create a list of prices: `val prices = listOf(10, 200, 5, 45, 12, 100)`.
    2. **Step A:** Create a new list called `expensiveItems` containing only prices greater than `40` (use `.filter`).
    3. **Step B:** Create a new list called `withTax` that takes the `expensiveItems` and adds 10% to each price (use `.map`).
    4. Print the final `withTax` list.

Delivery as a .txt file at: https://www.dropbox.com/request/OXlEuzDjNOyDTDdPzuc7