

# CMPS 312 Mobile Application Development

## Lab 3-Kotlin OOP and Lambdas

---

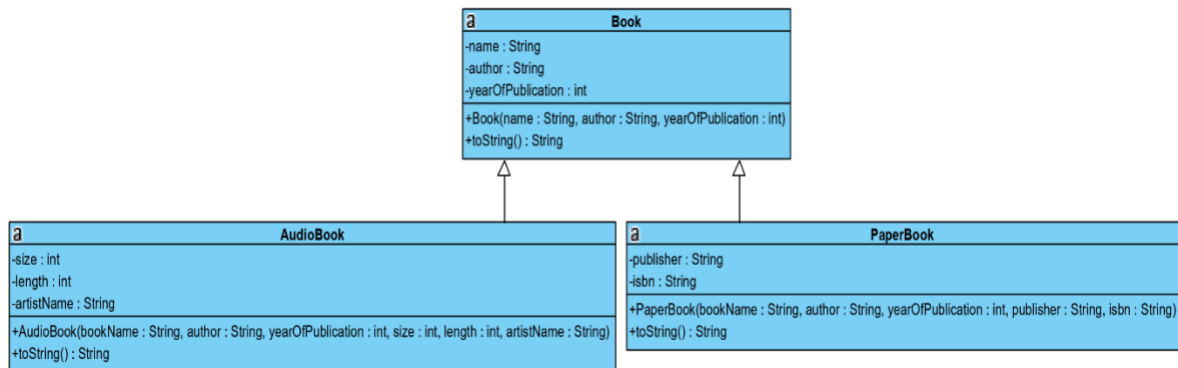
### OBJECTIVE

1. Practice Object Oriented Programming (OOP) using Kotlin
2. Read and parse JSON data
3. Practice processing collections using lambdas

### PART A – OOP

#### EXERCISE 1

1. Create an application named **Books** with no Activity and package name `cmps312.lab3.books`
2. Create a package called **model**.
3. Implement the following class hierarchy inside the model package.



- The `toString()` of **Book** should return **Name, Author, Year of Publication**.
  - The `toString()` of **PaperBook** should return **Name, Author, Year of Publication, Publisher, ISBN**.
  - The `toString()` of **AudioBook** should return **Name, Author, Year of Publication, Size, Length, ArtistName** (e.g., *Name: Ali Baba and the Forty Thieves, Author: Hanna Diyab*).
4. Create a main function to test your implementation.
  5. In the main function create a List having 2 audio books and 2 paper books.
  6. Display the details of each book using the list's `forEach` method.

## Sample Output

Book Name : C++  
Author Name : John  
Year Of Publication : 1/2/2019  
Publisher : Oriely  
Isbn : 100-11-11

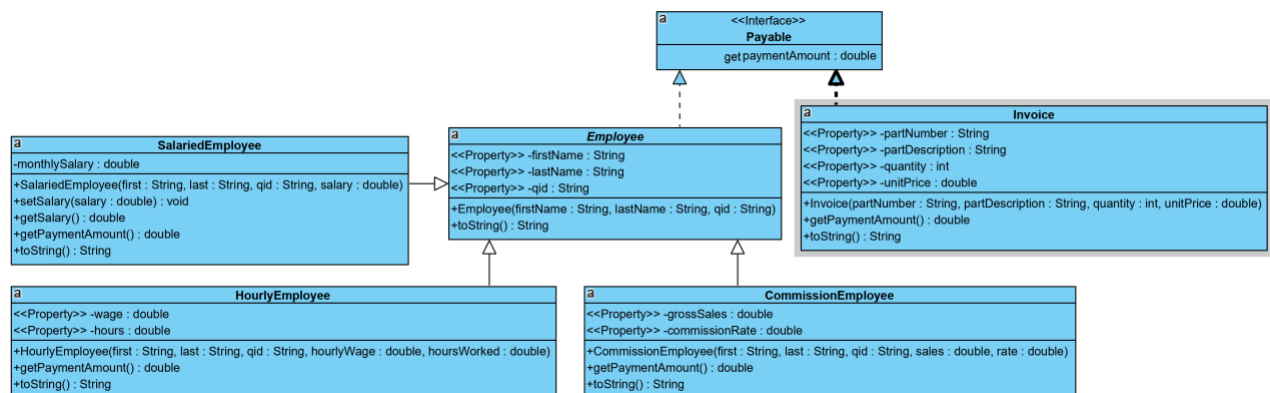
Book Name : Java  
Author Name : Mark  
Year Of Publication : 1/2/2019  
Publisher : NewTimes  
Isbn : 100-11-12

Book Name : Android  
Author Name : Baaji  
Year Of Publication : 1/2/2019  
Publisher : Sanford  
Isbn : 100-11-13

Book Name : How to get Rich  
Author Name : Ali  
Year Of Publication : 1/2/2019  
Size : 100  
Length : 25  
Artist Name : Black Panter

## EXERCISE 2

1. Create an application named **Payroll** with no Activity and package name `cmps312.lab3.payroll`
2. Create a package named **model**
3. Implement the following class hierarchy inside the **model** package



- Note that the amount to pay for `HourlyEmployee` is  $wage * hours$ . For `CommissionEmployee`, it is  $grossSales * commissionRate$ . For `Invoice`, it is  $quantity * unitPrice$ .
- Make sure the **salary**, **rate** and **sales** are all non-negative numbers otherwise display a warning message. [hint: for data validation using `init` or `set` methods]

## Test your implementation using the main method

```
fun main() {  
  
    // create payable array List  
    val payables = arrayListOf<Payable>()  
  
    // populate array with objects that implement Payable  
    payables.add(Invoice("01234", "Textbook", 2, 375.00))  
    payables.add(Invoice("56789", "USB Disk", 3, 179.95))  
    payables.add(SalariedEmployee("Ahmed", "Ali", "111-11-1111", 15000.00))  
    payables.add(HourlyEmployee("Fatima", "Saleh", "222-22-2222", 160.75, 40.0))  
    payables.add(CommissionEmployee("Samir", "Sami", "333-33-3333", 100000.0, .06))  
  
    println("Invoices and Employees processed polymorphically:\n");  
  
    // generically process each element in array payableObjects using foreach  
    payables.forEach { payable ->  
        // output currentPayable and its appropriate payment amount  
        println("$payable\n")  
  
        //If SalariedEmployee then increase the salary by 10%  
        if (payable is SalariedEmployee) {  
            val oldBaseSalary = payable.monthlySalary;  
            payable.monthlySalary = oldBaseSalary * 1.1;  
            println("New salary with 10%% increase is: QR ${payable.getPaymentAmount()}\n");  
        }  
    }  
}
```

Invoices and Employees processed polymorphically:

Part Number : 01234  
Part Description : Textbook  
Payment Amount : 750.0

Part Number : 56789  
Part Description : USB Disk  
Payment Amount : 539.8499999999999

First Name :Ahmed  
Last Name :Ali  
QID :111-11-1111  
Payment Amount : 15000.0

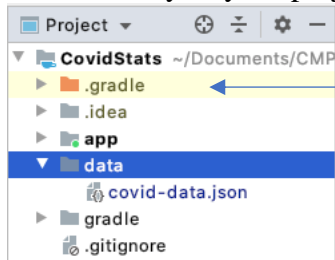
New salary with 10%% increase is: QR 16500.0

First Name :Fatima  
Last Name :Saleh  
QID :222-22-2222  
Payment Amount : 6430.0

First Name :Samir  
Last Name :Sami  
QID :333-33-3333  
Payment Amount : 6000.0

## **PART B - LAMBDA**

1. Create an application named **CovidTracker** with no Activity and package name `cmps312.lab3.covidtracker`
2. Copy the **covid-data.json** from **Lab 3** folder into a subdirectory named **data** under the root directory of your project (create the **data** subfolder yourself)



3. To be able use `@Serializable` and `Json` class you need to add these dependencies then sync:

a) Add to dependencies of the 2<sup>nd</sup> build.gradle  
`//Added for Kotlin Serialization`  
`implementation 'org.jetbrains.kotlinx:kotlinx-serialization-json:1.2.2'`

b) Add this apply plugin to the 2nd build.gradle under the `plugins{ }`  
`//Added for Kotlin Serialization`  
`id 'org.jetbrains.kotlin.plugin.serialization' version '1.5.30'`

4. Create a data class called **CovidStat** (in a Kotlin file named `CovidStat`). Derive `CovidStat` properties from the JSON object shown below. Note that some of the statistics could be null for some countries. Make `CovidStat` class serializable.

```
{
  "id": 1,
  "country": "United States",
  "continent": "Americas",
  "region": "Northern America",
  "totalCases": 6215592,
  "newCases": 38571,
  "totalDeaths": 187736,
  "newDeaths": 512,
  "totalRecovered": 3456263,
  "newRecovered": 30540,
  "activeCases": 2571593,
  "criticalCases": 15864,
  "casesPer1M": 18759,
  "deathsPer1M": 567,
  "totalTests": 82624841,
  "testsPer1M": 249373,
  "population": 331330464
},
```

5. Add **CovidStatRepository** object (in a Kotlin file named `CovidStatRepository`).

The `init` function of this object should **load** the json data in `data/covid-data.json` file into `covidStats` list.

The `CovidStatRepository` should implement the following functions that return:

- The **total COVID deaths around the world**.
  - The total **active cases** for a specific **continent**.
  - The **top five countries** with the **highest number of COVID cases**.
  - The **top five countries** with the **lowest number of COVID cases**.
  - The total **critical cases** of the neighboring countries of a given country. For example, if the country is `Qatar` then you should return all countries having the same region as `Qatar` and their respective critical cases. Finally sort those countries by population.
  - The top three regions in a **continent** with the highest recovery
  - The country with the **lowest death** in a **continent**.
6. Add a Kotlin file named **main** to test the functions in `CovidStatRepository` object. Test as you implement the functions of `CovidStatRepository` object. Do NOT leave the testing to the end.