Department of Computer Engineering
University of Peradeniya
CO523 - Programming Languages

Lab 01: Introduction to Programming Paradigms

---

1. Objectives

- Understand and explore different programming paradigms.
- Implement the same programming task using Procedural and Object-Oriented paradigms.
- Compare and analyze differences in program structure between the paradigms

2. Programming Paradigms

A **Programming Paradigm** is a fundamental approach or style of programming that provides a set of principles, concepts, and techniques for designing and implementing computer programs. It defines the structure, organization, and flow of the code, as well as the methodologies for problem-solving and expressing computations.

Programming paradigms dictate how programmers should think about and structure their code. They influence the way programs are written, the techniques used to solve problems, and the overall design philosophy. Different paradigms have their strengths and weaknesses, and choosing the right paradigm for a given task can greatly impact the efficiency, maintainability, and scalability of a program.

For more background on this topic, refer to the article:
https://en.wikipedia.org/wiki/Programming_paradigm

2.1. Procedural Programming

Procedural programming is a paradigm where the program is structured around procedures or functions that manipulate data. It focuses on step-by-step instructions and emphasizes code reusability through the use of functions.

2.2. Object-Oriented Programming (OOP)

Object-Oriented Programming revolves around the concept of objects, which are instances of classes. It organizes code into objects that encapsulate data and behavior. OOP promotes modularity, reusability, and allows for concepts such as inheritance, polymorphism, and encapsulation.

3. Lab Tasks

3.1. Bank Application Implementation

Implement a Bank Application in paradigms:
1. Procedural programming
2. Object-Oriented (OOP) programming

3.1.1. Procedural programming

Download the provided **Lab01_Skeleton.zip** file and use the provided skeleton code **procedural/ BankApp.c** as a starting point for your implementation.

For the test case provided in the `main()` method, your program should produce an output similar to the following.

```
Bank system initialized.
--- Starting Bank System (Default Package) ---
Created account 101 for Ruwan with balance 1000.00
Created account 102 for Nimal with balance 500.00
--- Transactions ---
[101] Deposited 300.00. New balance = 1300.00
[102] Withdrew 100.00. New balance = 400.00
--- Final Balances ---
[101] Owner: Ruwan, Balance: 1300.00
[102] Owner: Nimal, Balance: 400.00
Account not found.
```

Figure 1: Output of C program

3.1.2. Object-Oriented (OOP) programming

Implement the same bank application using OOP. Use two classes:

- BankAccount
- BankApp — the Driver Class

Use the provided skeleton codes in the **Lab01_Skeleton.zip**:

OOP/
- BankAccount.java
- BankApp.java

For the test case provided in the `main()` method, your program should produce an output similar to the following.

```
--- Starting Bank System (Default Package) ---
Created account 101 for Ruwan with balance 1000.00
Created account 102 for Nimal with balance 500.00
--- Transactions ---
[101] Deposited 300.00. New balance = 1300.00
[102] Withdrew 100.00. New balance = 400.00
--- Final Balances ---
[101] Owner: Ruwan, Balance: 1300.00
[102] Owner: Nimal, Balance: 400.00
Account not found.
```

Figure 2: Output of Java Program

4. Submission

Create a ZIP file named **CO523_Lab01_EXXYYY.zip**, where EXXYYY represents your E-number.

The ZIP file must contain your answer scripts organized using the following folder structure:

```
CO523_Lab01_EXXYYY/
│
├── Procedural/
│   └── BankApp.c
│
└── OOP/
    ├── BankAccount.java
    └── BankApp.java
```

Upload the Zip file to the FEeLS by the given deadline.