OOP
Object Oriented Programming

for: SCII/2019 / SCCI 2019 / SCCJ 2019
July – October 2022

By: Salesio M. Kiura
Department of Computer Science and Informatics
School of Computing and Informatics,
Technical University of Kenya (TU-K)

Session 1:
Introductions, Course Overview, Introduction to OOP as a paradigm

4th July 2022

The Course will be offered in a Blended version: Online and face to face

- Lecturer : Prof. Salesio M, Kiura
- Consultations: D21, timing: TBA
- Moodle link: https://elearning.tukenya.ac.ke/ Enrollment Key: OOP2019
- Delivery:
  - Lectures
  - Exercises / Activities
  - Assignments, Assessment tests
- Timing:
  - D23: Monday 7-9, 9 -11 / 11-1, 2-4 | | **online?**: Thursday 7-9

The course will be delivered in a learner-centered way

- Participants introductions
  - Names, Experiences with programming, prospects or plans in the field
  - Suggestions / Recommendations for the course
- Expect Participative engagements
  - Learner - centered learning
- This is a Practical course – practice, practice, practice!
  - Take your time to learn Programming (through practice)

Course objectives

- **At the end of the course, the student should be able to:**
  - Demonstrate an in-depth understanding of Object Oriented paradigm and concepts
  - Apply object oriented concepts using a selected language (Java)
  - Implement principles of inheritance, exception handling, abstract classes, packages, etc.
  - Analyze application scenarios (for) and design software systems using object oriented analysis and design.

Summary of contents

- Introducing OOP as a Paradigm
- OOP Concepts
- Getting started with Java
- Objects and Classes
- Functions
- Inheritance and Polymorphism (Practical implementations)
- File Operations using Java
- Mini-Project**

Reference material

- Books
  - Object oriented programming (with java)
    - David J. Barnes (or any other)
  - Java-The Complete Reference (11th Edition now)
    - You don't have to get the latest edition, the concepts are what we are focusing on
  - Fundamentals of Programming using Java
- Internet
- Java communities
- Colleagues!
- Software especially the IDE (?)

Reflection

- What are your thoughts / comments on the following questions:
  1. Why do we program?
  2. Why should we learn a new "programming language"
  3. What is/are some ideas I have (always have) for programming?
  4. What are (key) words that summarize your hitherto programming experiences?

Paradigms

- What is a paradigm?
  - In science, **paradigm** describes distinct concepts or thought patterns in any scientific discipline or other epistemological context
  - A **programming paradigm** is a fundamental style of computer programming
- Programming Paradigm
  - A way of conceptualizing what it means to perform computation and how tasks to be carried out on the computer should be structured and organized.
  - Computation refers to Implementation of an I/O relation

OOP as a paradigm

- As a paradigm, object oriented programming represents:
  - A programming "technique" (?)
  - A way of thinking about programming
  - A view of a program
- OOP is not a programming language
  - A programming language consists of words, symbols, and rules for writing a program
- There are four main paradigms : object-oriented, imperative, functional and logic programming

Programming paradigms

- ***Imperative Programming***
  - program as a collection of statements and procedures affecting data (variables)
- ***Object-Oriented Programming***
  - ***program as a collection of classes for interacting objects***
- ***Functional Programming***
  - program as a collection of (math) functions
- ***Logic programming***
  - Program as a collection of logical sentences

Imperative Programming

- What Makes a Language Imperative?
  - Programs written in imperative programming languages consist of
    - A program state
    - Instructions that change the program state
  - Program instructions are "imperative" in the grammatical sense of imperative verbs that express a command
    - Imperative verbs are also known as 'bossy verbs'- they tell people what to do! e.g. **close** the door; **empty** the bin; **eat** your dinner!

Imperative Languages

- Commands in an imperative language are similar to the native machine instructions of traditional computer hardware – the von Neumann-Eckley model.
- Some old Imperative Languages
  - assembly languages
  - 1954-1955: Fortran (FORmula TRANslator)
  - Late 1950's: Algol (ALGOrithmic Language)
  - 1958: Cobol (COmmon Business Oriented Language)

- Defining Characteristics of Imperative Languages:
  - Statements are <u>commands</u>
    - Command order is critical to correct execution;
    - Programmers control all aspects: algorithm specification, memory management, variable declarations, etc
  - They work by modifying program state
  - Statements reflect machine language instructions.

- Imperative: Summary
  - Imperative programming is the oldest programming paradigm
  - It is based on the von Neumann-Eckley model of a computer
  - It works by changing the program state through assignment statements
  - Procedural abstraction & structured programming are its design techniques.

Programming paradigms

- Programming paradigms
  - Imperative
  - **Functional**
  - Logic
  - OO

Functional Programming

- Functional programming is style of programming in which the basic method of computation is the application of functions to arguments;
- A functional language is one that supports and encourages the functional style.

Example

- **Consider:**

```
sum=  0;
for (int i = 1; i ≤ 10; ++i)
   total = total+i;
```

- *Imperative approach*
  - The computation method is variable assignment

- In a functional language this would be different:
  - `sum [1..10]`

  Other examples:
  - `Average [1..5]`
  - `Prod [2 5]`

- *Functional approach*
  - The computation method is function application

Haskell

A Purely Functional Language

- ➔ **functional programming** is a programming paradigm that treats computation as the evaluation of mathematical functions and avoids state and mutable data

Programming paradigms

- Programming paradigms
  - Imperative
  - Functional
  - **Logic**
  - OO

Logic Programming

- Based on Logic ! – what does this mean?
- **Logic programming** is based on the idea of using logical sentences to represent programs and to perform computation
- E.g.
  - M if true (C_1, C_2…C_n) AND False(C_{11},C_{22} ….C_{nn})

- Example languages
  - Prolog
  - LISP
  - Etc
- Application areas:
  - Knowledge representation
  - Problem solving
  - etc

Summary: Imperative, Functional, Logic Programming

- *Imperative programs* describe the details of ***HOW*** the results are to be obtained, in terms of the underlying machine model.
- *Functional/Logic programs* specify ***WHAT*** is to be computed abstractly, leaving the details of data organization and instruction sequencing to the interpreter.

Summary: Imperative, Functional, Logic Programming

Illustration Exercise

- Given the expression (to be computed) as: **a+b+c**
- **Which is imperative, which is functional?**
  1.
     - T := a + b; T := T + c;
  2.
     - Load a; Add b; Add c
     - Push a; Push b; Add; Push c; Add

Answer is on next Slide!

Illustration Exercise (Solution)

- **Which is imperative, which is functional? –answer:**

1. Functional
   - T := a + b; T := T + c; (Intermediate Code)
2. Imperative
   - Load a; Add b; Add c (Accumulator Machine)
   - Push a; Push b; Add; Push c; Add (Stack Machine)

- In life:
  - Are you imperative or functional?
  - Other classifications?

- Programming paradigms
  - √ **Imperative**
  - √ **Functional**
  - √ **Logic**
  - ❑ Object Oriented

OOP Definition

- Object-oriented programming (OOP) is a programming model organized around *objects rather than "actions"* and *data rather than logic*.
- Historically, a program has been viewed as a logical procedure that takes input data, processes it, and produces output data.

OOP approach

- In object oriented programming (software design) programmers define not only the data type of a ***data structure***, but also the types of operations (***functions***) that can be applied to the data structure. The main data structure becomes an object that includes both data and functions. Programmers then create ***relationships between one object and another***.

Evolution of OOP

- By the 1960s, programmers realized that programming systems needed to be broken up into small, manageable pieces.
- **SIMULA**-67 was the first object language. As its name suggests it was used to create simulations. SIMULA was designed by Dahl, Myhrhaug, and Nygaard at the Norwegian Computing Center at Oslo, Norway
- At Xerox Parc (in a project headed by Alan Kay from Utah university), a personal computer called the Dynabook was developed in early 1970s. **Smalltalk** was the object-oriented language developed for programming the Dynabook. It was a simulation and graphics-oriented programming language.

Evolution of OOP (2/2)

- Object-oriented programming gained momentum in the 1970s and in the early 1980s. Bjorn Stroustrup integrated object-oriented programming into the C language. The resulting language was called **C++** and it became the first object-oriented language to be widely used commercially.
- In the early 1990s, at Sun Microsystems a simpler version of C++ was developed called **Java** that was meant to be a programming language for video-on-demand applications. This project was re-oriented to focus on (and was marketed as being the language for) programming Internet applications. The language gained widespread popularity as the Internet has boomed, although its market penetration has been limited by its (earlier) inefficiency.

OOP Languages

- Simula, Smalltalk
- C++, Java, C#
- Scripting languages with OOP Support
  - Python, PHP, etc

➔All languages after 1990 are OO, many earlier ones have been "*retro-objectified*"!

- Object-Oriented Style
  - Involves programming with *Abstract Data Types*
    - ADTs specify/describe behaviors.
  - The basic program unit is a ***Class***
    - Implementation of an ADT.
    - Abstraction is enforced by encapsulation.
  - The basic Run-time unit is an ***Object***
    - Instance of a class.
    - Has an associated *state*.

EXAM Question

- With the use of an example, demonstrate the difference between Structured programming technique Vs. Object Oriented programming approach
  - Student management information system (Structured approach)

HINT OOP Definition

- Object-oriented programming (OOP) is a programming model organized around *objects rather than "actions"* and *data rather than logic*.
- Historically, a program has been viewed as a logical procedure that takes input data,

- Student attempts at the answer

Procedural vs Object-Oriented programming

- Procedural:
  - Emphasis on procedural abstraction.
  - Top-down design; Step-wise refinement.
  - Suited for programming in the small.
- Object oriented
  - Emphasis on data abstraction.
  - Bottom-up design; Reusable libraries.
  - Suited for programming in the large.

Lecturer example to demonstrate / explain the contents of this slide

Merits and Demerits of OOP

- Merits
  - Modularity (dealing with complexity)
  - Maintainability
  - Separation of concerns (best practices)
- Demerits
  - Learning curve
  - Inefficiency (including garbage collection at run time)

Assignment 1

A. We make a comparison between writing a structured program and an OOP program
   i. Write a program that reads from the user records of five students. Afterwards, the program prints on the screen the student records. Each student record stores: Serial Number (int), First Name (String), Course of Study(String), Grade Scored in Maths (char), Grade Scored in English (char), Grade Scored in Physics (char), Grade Scored in Computer Studies (char)

- Using C/C++
- Using Java

Assignment 1 – Learning assessment

Skill	Guide
Configure Visual Studio Code to use the GCC (or other) C++ compiler (g++) and GDB debugger from mingw-w64 to create programs that run on Windows.	Ref: https://code.visualstudio.com/docs/cpp/config-mingw Alternatively you can use Dev-C++ as another IDE for CPP specific files
Progressive adoption of OOP approach - Structured in OOP - Real OOP	IDE and Lecturer Guide

Thank you

- NEXT
  - Basic Concepts of OOP
    Getting started with Java