PÉCSI TUDOMÁNYEGYETEM
TERMÉSZETTUDOMÁNYI KAR

“Methodology of Programming I”

Spring 2025

PTE

Zahra Ahmadipour

Zahra@gamma.ttk.pte.hu

Course objectives:

This course introduces the fundamental principles and methodologies of computer programming. By the end of the course, students shall be able to design and implement a structured program using Java programming language.

Course requirements and grading:

- Homework (25%)
- Exam (75%)
  - Practice
  - Theory

Course outline:

- Introduction, Set up development environment (JDK and Apache NetBeans)
- Programming languages categories, Paradigms, Variables, Data types
- Operators, Wrapper classes
- Arrays, Control flows, Loops
- Methods, Collections
- Object oriented programming basics, Classes, Modifiers
- Constructors, Encapsulation, Inheritance
- Abstraction, Polymorphism, Enums
- Exception handling
- Basic I/O
- XML Processing
- Generics
- Swing framework

Reasons for Studying Concepts of Programming Languages:

- Increased capacity to express ideas
- Improved Background for choosing appropriate languages
- Increased ability to learn new languages
- Better understanding of the significant implementation
- Better use of languages that are already known
- Overall advancement of computing

Programming Domains

Computers have been applied to a myriad of different areas, from controlling nuclear power plants to providing video games in mobile phones.

Because of this great diversity in computer use, programming languages with very different goals have been developed.

Some Areas of Computer Applications:

- Scientific Applications (Fortran, MATLAB, Python)
- Business Applications (COBOL, Java)
- Artificial Intelligence (LISP, Prolog, C, Python, R)
- Systems Programming (Assembly Language, PL/I, C)
- Web Software (HTML, Javascript, Java, PHP)

Scientific Applications:

The first digital computers, which appeared in the late 1940s and early 1950s, were invented and used for scientific applications.

Typically, the scientific applications of that time used relatively simple data structures, but required large numbers of floating-point arithmetic computations.

The first language for scientific applications was Fortran.

Business Applications:

The use of computers for business applications began in the 1950s. Special computers were developed for this purpose, along with special languages.

Business languages are characterized by facilities for producing elaborate reports, precise ways of describing and storing decimal numbers and character data, and the ability to specify decimal arithmetic operations.

The first successful high-level language for business was COBOL.

Artificial Intelligence:

AI is a broad area of computer applications characterized by the use of symbolic rather than numeric computations.

Symbolic computation is more conveniently done with linked lists of data rather than arrays. This kind of programming sometimes requires more flexibility than other programming domains.

The first widely used programming language developed for AI applications was the functional language LISP, which appeared in 1959.

Systems Programming:

The operating system and the programming support tools of a computer system are collectively known as its systems software. Systems software is used almost continuously and so it must be efficient.

In the 1960s and 1970s, some computer manufacturers, such as IBM, and Digital developed special machine-oriented high-level languages for systems software on their machines. For IBM mainframe computers, the language was PL/S, a dialect of PL/I; for Digital, it was BLISS, a language at a level just above assembly language.

Most system software is now written in more general programming languages, such as C and C++.

Web Software:

The World Wide Web is supported by an eclectic collection of languages, ranging from markup languages, such as HTML, which is not a programming language, to general-purpose programming languages, such as Java.

Because of the pervasive need for dynamic Web content, some computation capability is often included in the technology of content presentation. This functionality can be provided by embedding programming code in an HTML document. Such code is often in the form of a scripting language, such as JavaScript or PHP.

Programming paradigms

A programming paradigm is a style of programming. Programming languages can be classified into multiple paradigms.

Some paradigms are concerned mainly with implications for the execution model of the language; other paradigms are concerned mainly with the way that code is organized, such as grouping a code into units along with the state that is modified by the code; yet others are concerned mainly with the style of syntax and grammar.

Common Paradigms:

- Imperative
- Functional
- Procedural
- Logic/Declarative
- Object-oriented

Imperative Programming:

The basic architecture of computers has had a profound effect on language design. Most of the popular languages have been designed around the prevalent computer architecture, called the von Neumann architecture. These languages are called imperative languages.

Because of this architecture, the central features of imperative languages are variables, assignment statements, and the iterative form of repetition.

The execution of a machine code program on a von Neumann architecture computer occurs in a process called the fetch-execute cycle. Programs reside in memory but are executed in the CPU. Each instruction to be executed must be moved from memory to the processor. The address of the next instruction to be executed is maintained in a register called the program counter.

The fetch-execute cycle can be simply described by the following algorithm:

```
initialize the program counter
repeat forever
    fetch the instruction pointed to by the program counter
    increment the program counter to point at the next instruction
    decode the instruction
    execute the instruction
end repeat
```

Imperative programming consists of a set of detailed instructions that are given to the computer to execute in a given order. It's called "imperative" because as programmers we dictate exactly what the computer has to do.

Fortran, Java, C, C++ programming languages are examples of imperative programming.

Perl, JavaScript, and Ruby, are imperative languages in every sense.

Functional Programming:

A functional, or applicative, language is one in which the primary means of computation is applying functions to given parameters.

Functional languages offer potentially the greatest overall simplicity, because they can accomplish everything with a single construct, the function call, which can be combined simply with other function calls.

Other factors, such as efficiency, have prevented functional languages from becoming more widely used.

In functional programming, functions are treated as first-class citizens, meaning that they can be assigned to variables, passed as arguments, and returned from other functions.

LISP, Scheme, ML, Haskell, and F#, JavaScript, Python, Scala, Erlang, are examples of functional programming.

Procedural Programming:

Programming in both imperative and functional languages is primarily procedural, which means that the programmer knows what is to be accomplished by a program and instructs the computer on exactly how the computation is to be done.

In other words, the computer is treated as a simple device that obeys orders. Everything that is computed must have every detail of that computation spelled out.

FORTRAN, ALGOL, COBOL, BASIC, Pascal and C are examples of procedural programming.

Logic/Declarative:

Programs in logic programming languages are collections of facts and rules. Such a program is used by asking it questions, which it attempts to answer by consulting the facts and rules.

Logic programs are declarative rather than procedural, which means that only the specifications of the desired results are stated rather than detailed procedures for producing them.

The computer system can somehow determine how the result is to be computed.

Languages used for logic programming are called declarative languages, because programs written in them consist of declarations rather than assignments and control flow statements.

The syntax of logic programming languages is remarkably different from that of the imperative and functional languages.

Miranda, Erlang, Haskell, Prolog are a few popular examples of declarative programming.

Object-Oriented Programming:

The latest step in the evolution of data-oriented software development, which began in the early 1980s, is object-oriented design. Object-oriented methodology begins with data abstraction, which encapsulates processing with data objects and controls access to data, and adds inheritance and dynamic method binding.

We do not consider languages that support object-oriented programming to form a separate category of languages. The most popular languages that support object-oriented programming grew out of imperative languages.

Some of the newer languages that were designed to support object-oriented programming do not support other programming paradigms but still employ some of the basic imperative structures and have the appearance of the older imperative languages. Among these are Java and C#.

Object-oriented concepts have also found their way into functional programming as well as logic programming.

The core concept of OOP is to separate concerns into entities which are coded as objects. Each entity will group a given set of information and actions that can be performed by the entity.

Java, C#, Python, PHP, JavaScript, Ruby, Perl, are a few popular examples of Object-Oriented programming.

Frontend vs Backend:

Frontend and Backend are the two most popular terms used in web development.

The front end is what users see and interact with and the backend is how everything works.

Each side needs to communicate and operate effectively with the other as a single unit to improve the website’s functionality.

Frontend:

It includes everything that users experience directly.

It is also referred to as the ‘client side’ of the application.

Responsiveness and performance are the two main objectives of the Front End.

HTML, CSS, and JavaScript are the languages used for Front End development.

Backend:

The backend is the ‘server side’ of the website.

It is the part of the website that you cannot see and interact with.

It stores and arranges data, and also makes sure everything on the client side of the website works fine.

PHP, C++, Java, Python, and Node.js are some languages used for Back End development.

References:

Concepts of Programming Languages By Robert W. Sebesta

https://en.wikipedia.org/wiki/Programming_paradigm

https://www.freecodecamp.org/news/an-introduction-to-programming-paradigms/

https://www.geeksforgeeks.org/