PÉCSI TUDOMÁNYEGYETEM
TERMÉSZETTUDOMÁNYI KAR

“Methodology of Programming I”

Spring 2025

PTE

Zahra Ahmadipour

Zahra@gamma.ttk.pte.hu

OOP

Object-Oriented Programming or OOPs refers to languages that use objects in programming, they use objects as a primary source to implement what is to happen in the code.

The main aim of OOP is to bind together the data and the functions that operate on them so that no other part of the code can access this data except that function.

OOP Advantages:

OOP is fast and easy to execute

OOP provides a clear structure for the programs

OOP helps to keep the Java code DRY "Don't Repeat Yourself"

OOP makes the code easier to maintain, modify and debug

Object:

Objects are key to understanding object-oriented technology.

Real-world objects share two characteristics: **state** and **behavior**.

An object stores its state in fields (variables in some programming languages) and exposes its behavior through methods (functions in some programming languages).

Bundling code into individual software objects advantages:

- Modularity: The source code for an object can be written and maintained independently.
- Information-hiding: By interacting only with an object's methods, the details of its internal implementation remain hidden from the outside world.
- Code re-use: If an object already exists, you can use that object in your program.
- Pluggability and debugging ease: If a particular object turns out to be problematic, you can simply remove it or replace it.

Class:

A class is the blueprint from which individual objects are created.

An object is an instance of a class.

When the individual objects are created, they inherit all the variables and methods from the class.

Constructor:

Constructor is a special method that is used to initialize objects.

The constructor is called when an object of a class is created.

The constructor name must match the class name, and it cannot have a return type.

All classes have a constructor by default which you are not able to set initial values for object attributes.

Modifiers:

They are used to provide information about class functionalities to JVM. They are divided into two categories:

- Access modifiers:
  Control access mechanisms and help to restrict the scope of a class, constructor, variable, method, or data member.

- Non-access modifiers:
  Provide information about the characteristics of a class, method, or variable to the JVM.

Access Modifiers:

- default: No modifier specified. Accessible only within its package.
- **private**: Can be accessed only by its class members.
- **protected**: Accessible within its package and by its subclasses in other packages.
- **public**: The code is accessible from everywhere.

Non-access Modifier:

- **static**: The entity to which it is applied is available outside any particular instance of the class.
- **final**: The final keyword indicates that the specific class cannot be extended or a method cannot be overridden.
- **abstract**: Used to declare a class as partially implemented; meaning an object cannot be created directly from that class. Any subclass of an abstract class needs to either implement all the methods of the abstract class, or be an abstract class itself.

  The abstract keyword cannot be used with static, final, or private keywords because they prevent overriding.

- **synchronized**: Prevents a block of code from executing by multiple threads at once. It is very important for some critical operations.
- **volatile**: Used to make the class thread-safe. The volatile keyword is only applicable to a variable.
- **transient**: When we do not want to serialize the value of a variable, then we declare it as transient.
- **native**: To indicate that the method is implemented in a language other than Java.

For classes you can use default, public, final, and abstract modifiers.

For constructors you can use default, public, protected, private modifiers.

For methods and attributes all modifiers are applicable.

Pillars of OOP:

- Encapsulation
- Inheritance
- Polymorphism
- Abstraction

Encapsulation:

The word’s definition is the action of enclosing something in or as if in a capsule.

Encapsulation in computing is defined as the wrapping up of data under a single unit.

It is a protective shield that prevents the data from being accessed by the code outside this shield.

Encapsulation can be achieved by declaring all the variables in the class as private and writing public methods in the class to set and get the values of variables; mostly known as setter and getter methods.

By using get method user should be able to access the value of the variable.

By using set method user should be able to set or change the value of the variable.

Encapsulation advantages:

- Data Hiding and security
- Increased flexibility by making data read-only or write-only
- Reusability
- Easier testing

References:

https://docs.oracle.com/javase/tutorial/tutorialLearningPaths.html

https://www.w3schools.com/

https://www.geeksforgeeks.org/