st. gr. FAF-131, Adauge Marius

APPOO – Laboratory work nr. 0

Task: Analysis of two programming languages based on core OOP concepts – inheritance, encapsulation, polymorphism

	Java	Python

Encapsulation	strong encapsulation;
	
	explicit access modifiers which allows to make data in a class private and the methods public which allows access to data
		more based on the programmer's self-consciousness not to mess with class data;
		
		however, there is a naming convention of variables for denoting access attributes:
		_varName = protected
		__varName = private

Inheritance	Do not support multiple inheritance
	
	However, a class can implement one or more interfaces. This has made Java get rid of the impossibility of multiple inheritance
	
	In Java, children classes **must** call the parent's constructor (refinement overriding)
		Supports multiple inheritance
		class A(B,C):
		    …
		
		In Python, children classes can override the constructor and not call the parent's constructor (replacement overriding)

Polymorphism	Polymorphic collections:
	`Shape[] s = { new Circle(), new Square(), … };`
	
	Straight-forward **overloading**:
	`public int add(int a, int b)`
	`{ return a + b;}`
	
	`public double add(double a, double b)`
	`{ return a + b; }`
		Polymorphic collections:
		`s = [ Circle(), Square(), … ]`
		
		Overloading can be done like this:
		`def funtionName(a, b):`
		`        if not isinstance(a, int) or not isinstance(b, int):`
		`            #do this`
		`            return something`

The core OOP concepts in my code is pointed with corresponding comments. See the source code.