**Intalnirea 1**

Introducere in Java si IntelliJ

1. Ce este Java?

• Este un limbaj de programare bazat pe paradigma Programare Orientata Obiect ce ne permite scrierea unor functionalitati folosind un set de reguli si instructiuni ce au la baza keyword-uri speciale. (Numita si sintaxa)

• Sintaxa este usor de inteles din moment ce nu folosim metode numerice pentru a scrie un program. Acest limbaj se foloseste de cuvinte cheie pentru a marca anumite instructiuni.

• Programare Orientata pe Obiect (POO) / Object Oriented Programming (OOP)

2. De ce Java?

• Portabilitate: datorită platformei JVM Java poate rula pe aproape orice tip de sistem.

i. Interesting news: Java este, de asemenea, cel mai popular limbaj de Android, astfel ca majoritatea aplicatiilor Android este construită în Java.

• Scalabilitate: Java a fostr construit astfel incat extinderea sa fie facuta cu usurinta. (Asadar nu avem nicio sansa sa ne plictisim de acest limbaj deoarece este intr-o continua dezvoltare).

• Comunitate mare: Stim ca este un limbaj versatil si necesita timp, dar informatia se gaseste mai usor

• Compatibilitate: Nu trebuie sa ne facem griji daca scriem cod java intr-o versiune mai veche decat cea pe care o folosim atunci cand rulam. Java ne garanteaza conceputul de *backwards compatibility*

3. Ce inseamna compilarea codului sursa?

• In general unn compilator este un program care traduce codul sursa care poate fi citit de om in codul masinii executabil de computer (In cazul Java il compileaza in *Bytecode* pentru *JVM*) . Pentru a face acest lucru cu succes, codul care poate fi citit de om trebuie sa respecte regulile de sintaxa ale limbajului de programare în care este scris. Compilatorul este doar un program si nu va poate repara codul. Daca faceți o greseala, trebuie sa corectati sintaxa sau nu se va compila.

4. Clase si pachete

• Clasele sunt elemente fundamentale ale oricarei aplicatii Java. Ele reprezinta structuri logice care definesc comportamentul si starea obiectelor.

• O clasa este definita pentru a descrie un nou tip de date. Dupa definirea clasei, aceasta poate fi utilizata pentru a crea obiecte de tipul respectiv. Deci, o clasa reprezinta un sablon pentru un obiect, iar un obiect reprezinta o instanta a unei clase.

Pentru innceput o sa ne asumam un template de creare al unei clase pana la capitolul dedicat. Tinem minte ca recomadarea cea mai buna referitoare la crearea unei clase, este ca numele acesteia sa coincida cu numele fisierului.

package com.itfactory;

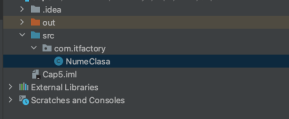
public class NumeClasa {

}

• In exemplul de mai sus avem o simpla declaratie a unel clase in care numele acesteia este *NumeClasa.*

• Delimitarea corpului unei clase o facem prin acolade.

• Mai jos observam numele fisierului create (care coincide cu numele clasei din declaratie)



• De asemenea observam declararea pachetului din care face parte, ca fiind prima linie din acel fisier\*\*\* *package com.itfactory;*

• In general, pentru a putea executa un program in Java, avem nevoie de metoda *main* pentru a putea face aceasta actiune. Metoda main se defineste astfel.

• Atentie, se poate crea doar o singura metoda *main* intr-o clasa iar aceasta nu trebuie definita in afara clasei. Mai mult de atat, o sa vedem ca nu putem scrie niciun fel de cod in Java in afara unei clase (de aici si faptul ca este orientat POO / OOP)

public static void main(String[] args) {

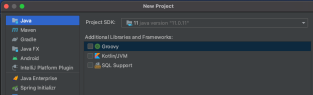
}

5. Cum cream un proiect in IntelliJ daca avem deja deschis un alt proiect?

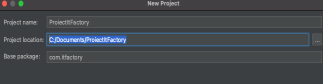
• File -> New -> Project ->

o Aici o sa apara fereastra de selectie a tipului de proiect

o Selectam Java



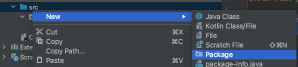
• -> Create Project from Template -> Command Line App ->

• -> Finish

6. Cum cream un pachet?

• Click dreapta pe *src* -> New -> Package

• Indicatie: Din moment ce avem pachetul de baza *com.itfactory* sugestia este sa creati pachete sub acesta. (Click dreapta pe *com.itfactory* in loc de src)



7. Cum cream o clasa?

• Click dreapta pe pachetul dorit -> New -> Java Class

