

LUCRAREA NR. 1

Prezentarea mediului de dezvoltare Eclipse. Primele aplicații în Java

1. Scopul lucrării

Lucrarea de laborator are ca scop prezentarea mediului de dezvoltare Eclipse, ce va fi folosit la laboratoarele din cadrul disciplinei “Programarea calculatoarelor și limbaje de programare II”. De asemenea se vor descrie etapele ce trebuie parcurse pentru crearea unei aplicații Java independente. În prima parte se va prezenta pe scurt tehnologia Java.

2. Tehnologia Java – scurtă prezentare

Tehnologia Java nu este numai un limbaj de programare, este și o platformă, adică un mediu de programare ce oferă utilizatorului cadrul și uneltele necesare dezvoltării aplicațiilor Java.

Platforma Java are două componente:

- A) Mașina virtuală Java (JVM) - aceasta este o platformă virtuală, un interpretor pentru execuția codului de octeți (bytecode) Java.
- B) Interfața de programare a aplicațiilor (API) - pachetele API conțin clase și interfețe pentru construirea aplicațiilor.

La baza dezvoltării unui program Java stă mediul de dezvoltare pus la dispoziție de firma Oracle. Acesta este Java Development Kit (JDK), ultima versiune lansată fiind JDK 7u40. Orice mediu de dezvoltare (ex: Eclipse, NetBeans, JBuilder) care poate executa aplicații Java diferit de mediul JDK trebuie să includă API.

Prin urmare, pentru utilizarea mediului Eclipse este necesară instalarea JDK sau JRE.

Ce este JRE (Java Runtime Environment)?

- este o implementare a Java Virtual Machine, care execută de fapt programele Java;
- include JVM, librăriile de bază și alte componente adiționale pentru execuția aplicațiilor și applet-urilor Java.

Ce este JDK (Java Development Toolkit)?

- este o colecție de software folosită pentru dezvoltarea aplicațiilor Java;
- include JRE, un set de clase API, compilatorul Java și alte fișiere necesare pentru execuția aplicațiilor și applet-urilor Java.

LUCRAREA NR. 1

3. Instalare JDK și Eclipse SDK

Pachetul JDK poate fi descărcat de pe site-ul firmei Oracle :

<http://www.oracle.com/technetwork/java/index.html>

Se recomandă instalarea într-un director, de exemplu C:\Program Files\Java\jdk1.7.0_17. După instalare se adaugă în variabilele de mediu ale sistemului calea către directorul bin, ca în Figura 1.

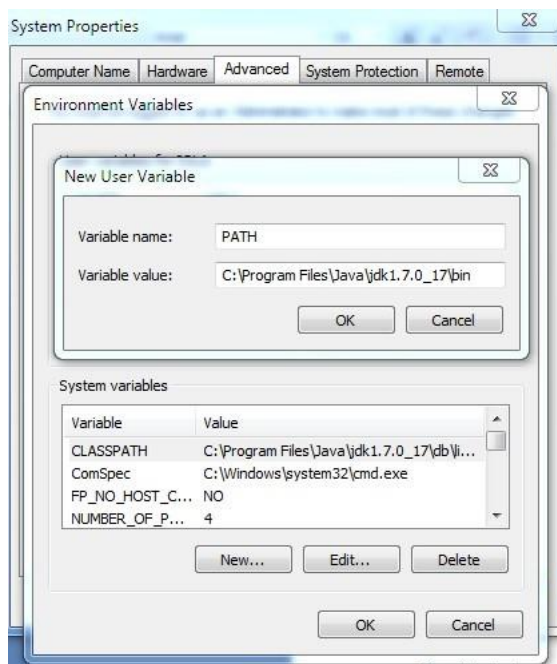
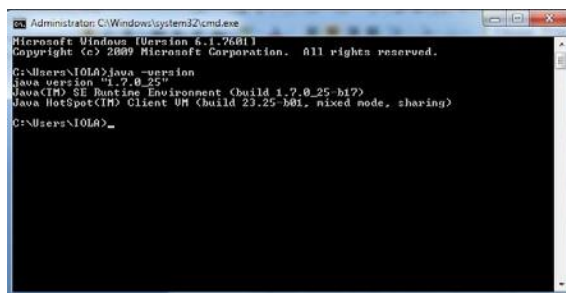


Figura 1. Editarea variabilelor de mediu

Se poate verifica funcționarea JDK-ului instalat prin scrierea în linia de comandă a comenzilor: **java -version** (returnează versiunea JDK-ului instalat) și **javac**.



După instalarea JDK-ului se poate descărca Eclipse, de pe site-ul <http://www.eclipse.org/downloads/>. Se dezarchivează fișierul Eclipse SDK într-un director și se execută fișierul eclipse.exe.

LUCRAREA NR. 1

4. Prezentare Eclipse SDK

Selectarea spațiului de lucru (workspace)

La pornirea mediului Eclipse, se selectează spațiul de lucru (workspace) (Figura 2), ce reprezintă locația pe disc unde vor fi stocate proiectele Java create. Dacă vor exista mai multe directoare de lucru, este posibilă trecerea dintr-un workspace în altul folosind meniul **File-> Switch Workspace-> Others**.

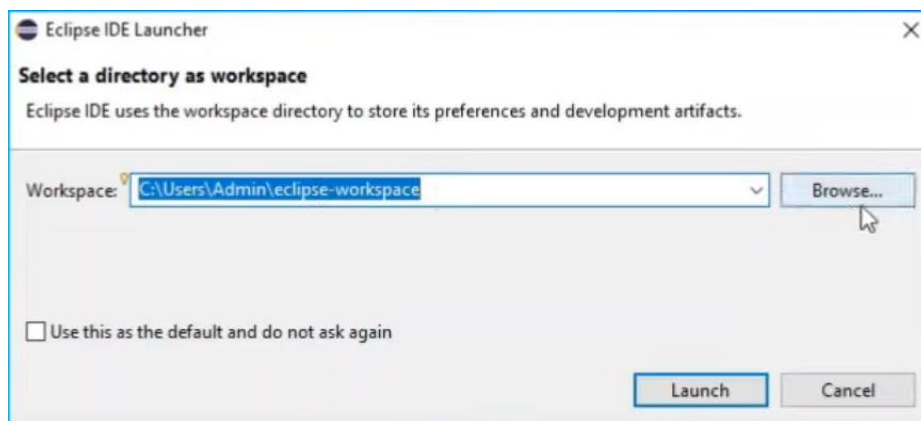


Figura 2. Selectarea spațiului de lucru

După pornirea mediului Eclipse se va deschide fereastra Welcome (Figura 3)

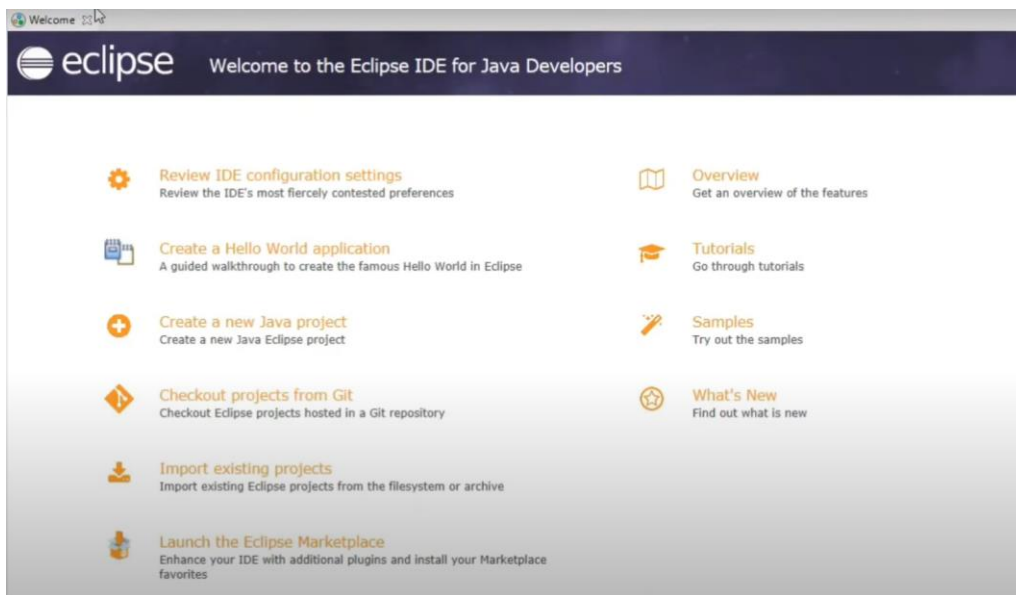


Figura 3. Fereastra Welcome

LUCRAREA NR. 1

Această fereastră poate fi deschisă și ulterior, din meniul **Help->Welcome**. Verificați ce reprezintă fiecare icoană din această fereastră.

Alegerea modului de vizualizare (perspectivă) standard

O perspectivă definește setarea inițială a modului de vizualizare în fereastră. Perspectiva controlează ceea ce apare în anumite meniuri și bare de instrumente. De exemplu, o perspectivă Java conține vizualizări pe care utilizatorul le-a folosit frecvent pentru editarea fișierelor sursă Java, în timp ce perspectiva Debug conține moduri de vizualizare ce sunt folosite pentru depanarea programelor Java. Pentru a crea programe Java se setează perspectiva Java din meniul **Window ->Open Perspective -> Java (default)**, noua interfață utilizator fiind prezentată în Figura 4.

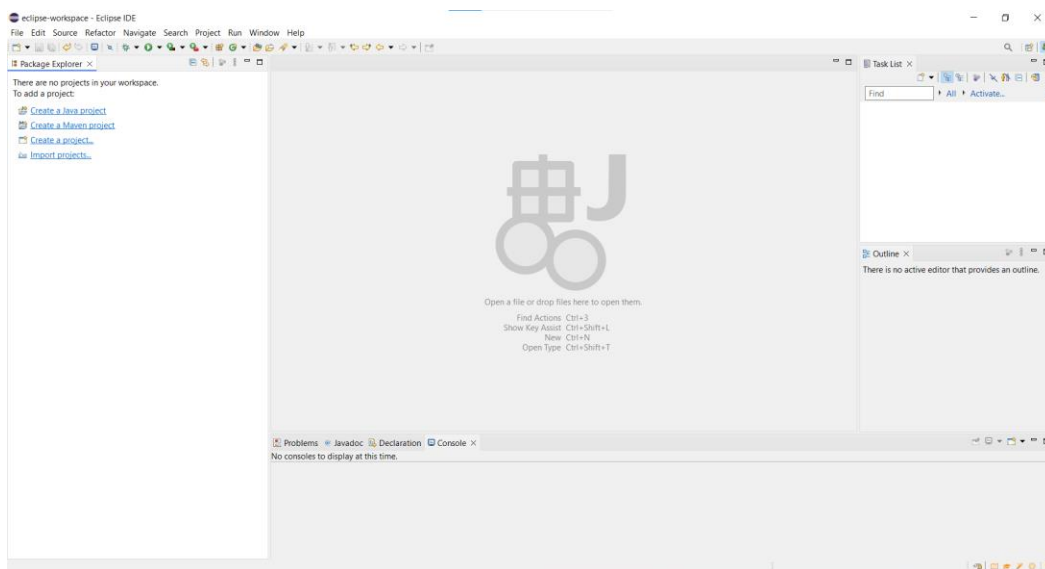


Figura 4. Perspectiva Java

Crearea unui proiect

Pentru a crea un proiect în Eclipse se selectează din meniul **File ->New -> Java project**. Se alege versiunea de JRE dorită, se dă un nume proiectului, de exemplu Laborator1 (vezi Figura 5), se selectează “Create separate folders for sources and class files” și apoi se apasă **Finish**.

LUCRAREA NR. 1

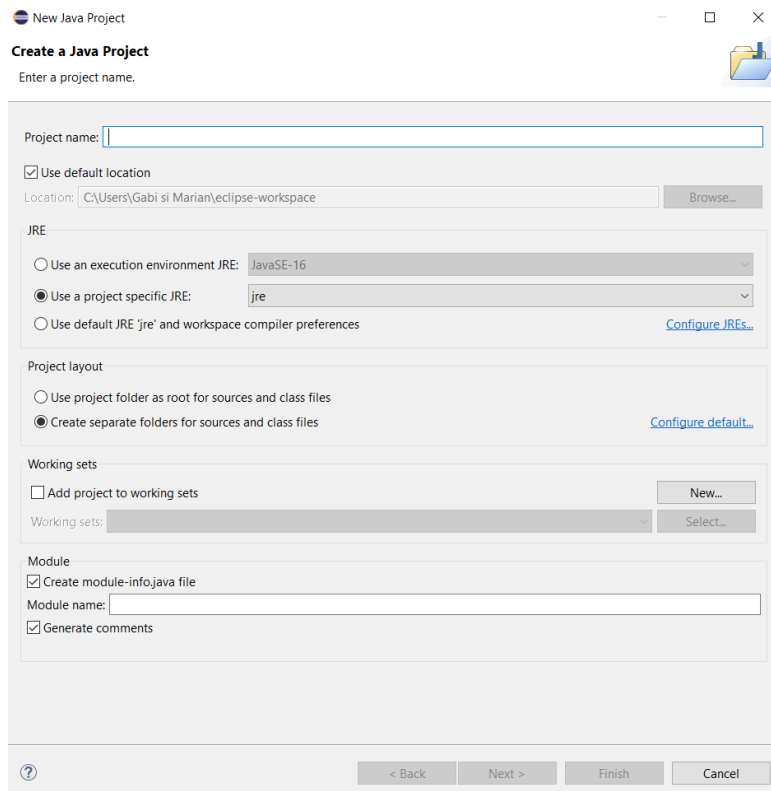


Figura 5. Crearea unui proiect Java

Proiectul va apare în fereastra Package Explorer și se creează automat un director numit *src* ce va stoca fișierele sursă (Figura 6)

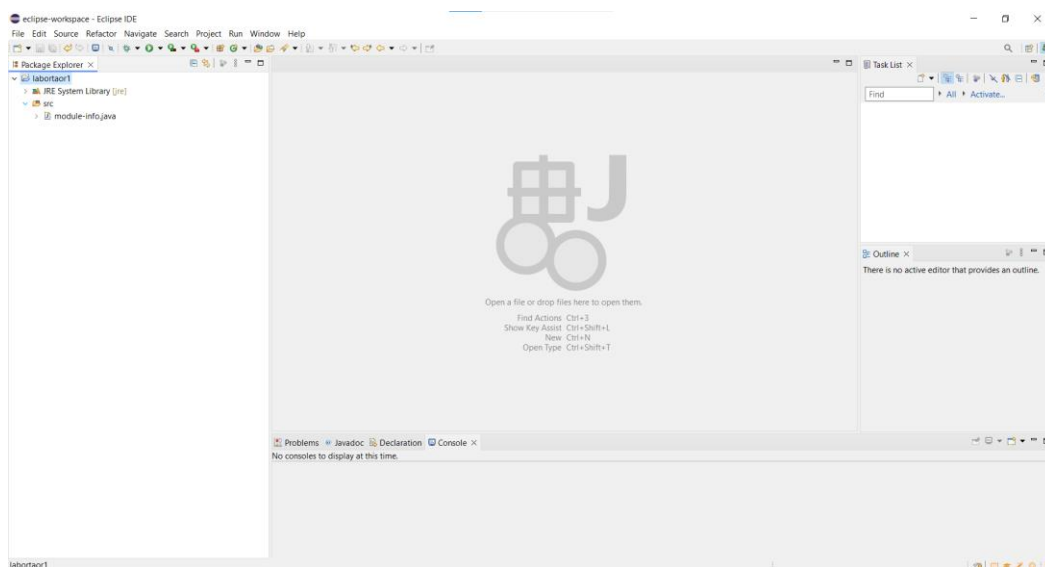


Figura 6. Crearea automată a directorului src

LUCRAREA NR. 1

Crearea unui pachet (package)

Înainte de a scrie codul sursă trebuie să adăugăm un pachet nou în proiectul creat anterior. Pentru aceasta se selectează **File-> New-> Package** și se dă un nume pachetului, de regulă același nume utilizat pentru proiect (Figura 7) apoi se apasă Finish.

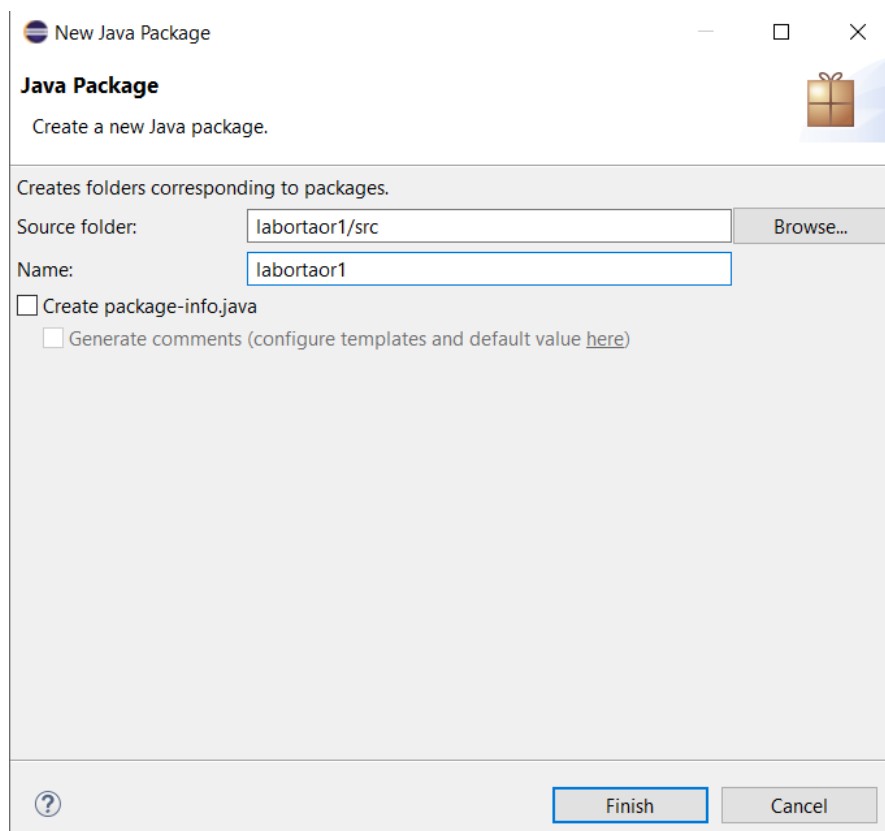


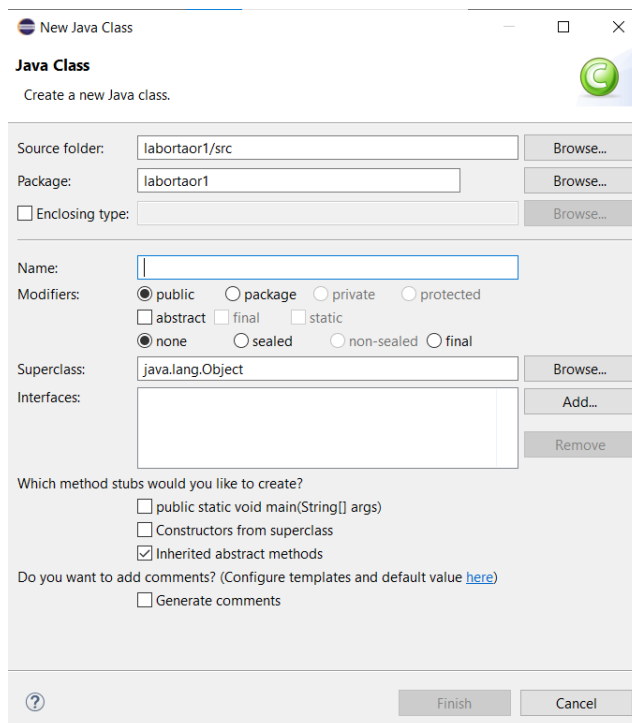
Figura 7. Crearea unui pachet

Un pachet Java reprezintă o entitate logică care permite organizarea claselor și gestionarea spațiului de nume. Astfel se pot grupa clase și interfețe care au legătură între ele. Prin folosirea pachetelor se pot evita conflictele de nume între clase, fiind permisă existența în cadrul aceleiași aplicații a mai multe clase având același nume dar organizate în pachete diferite.

Crearea unei clase

După ce s-a creat un pachet, se poate crea o nouă clasă Java. Astfel, se apasă click dreapta pe pachetul creat și se selectează **New-> Class**. În câmpul Name se scrie numele clasei, pentru exemplul ce va fi prezentat în continuare s-a ales numele **Welcome** (Figura 8).

LUCRAREA NR. 1



New Java Class

Create a new Java class.

Source folder: labortaor1/src Browse...

Package: labortaor1 Browse...

☐ Enclosing type: Browse...

Name:

Modifiers: ☒ public ☐ package ☐ private ☐ protected
☐ abstract ☐ final ☐ static
☒ none ☐ sealed ☐ non-sealed ☐ final

Superclass: java.lang.Object Browse...

Interfaces: Add... Remove

Which method stubs would you like to create?

☐ public static void main(String[] args)
☐ Constructors from superclass
☒ Inherited abstract methods

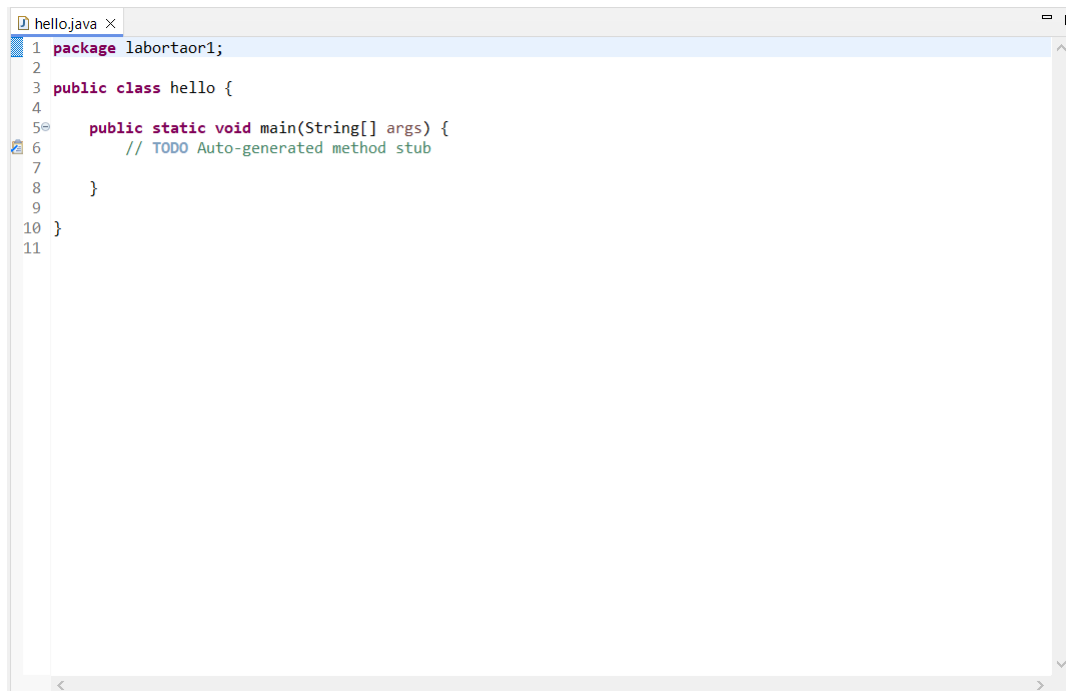
Do you want to add comments? (Configure templates and default value [here](#))

☐ Generate comments

? Finish Cancel

Figura 8. Crearea unei clase

Se selectează opțiunea **public static void main(String[] args)** și apoi se apasă butonul **Finish** pentru a genera un template pentru codul sursă Welcome.java (Figura 9a).



```

1 package labortaor1;
2
3 public class hello {
4
5     public static void main(String[] args) {
6         // TODO Auto-generated method stub
7     }
8 }
9
10
11

```

Figura 9a. Template generat pentru codul sursă

LUCRAREA NR. 1

În cele ce urmează se va scrie codul pentru o simplă aplicație care va tipări la consolă mesajul “Welcome to Java”.

În metoda **main** se va scrie următorul rând: **System.out.println (“Welcome to Java”);** (vezi Figura 9b).

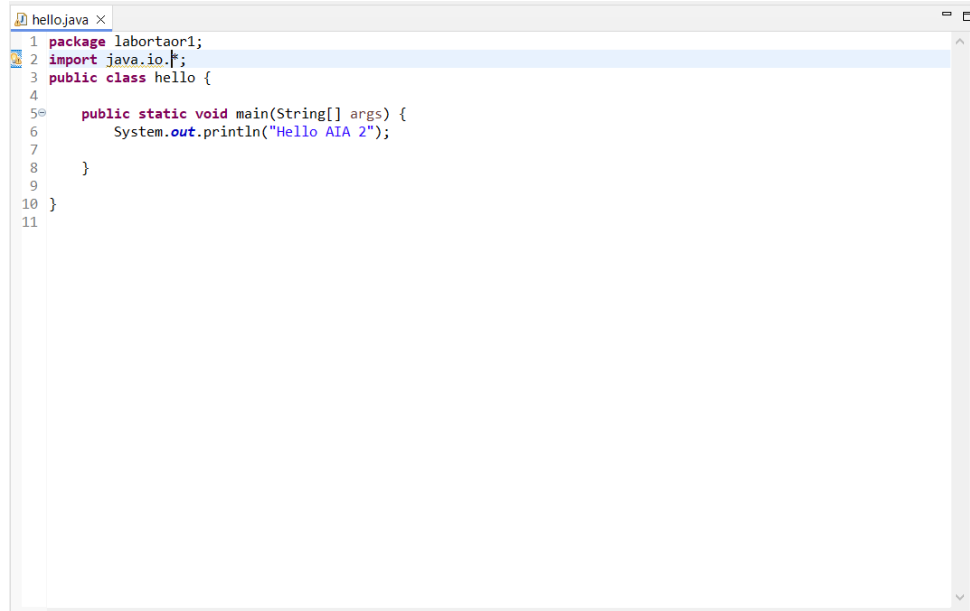


Figura 9b. Codul sursă Java

Se observă că pe măsură ce se scrie cod, automat în ajutorul utilizatorului vor apare diferite sugestii pentru completarea codului. Se salvează fișierul și se trece la următorul pas, execuția proiectului Java.

Compilarea și execuția proiectului Java

În mod implicit, codul sursă este compilat dinamic, pe măsură ce este scris. Dacă sunt erori de sintaxă acest lucru este marcat printr-un cerculeț roșu în partea stângă a liniei de cod eronate.

Pentru execuția aplicației se selectează din meniul **Run**, comanda **Run As-> Java Application** sau prin click dreapta pe clasa din proiect în fereastra “Package Explorer” și se alege **Run as-> Java Application**.

Rezultatul execuției va apărea în fereastra consolă (Figura 10).

LUCRAREA NR. 1

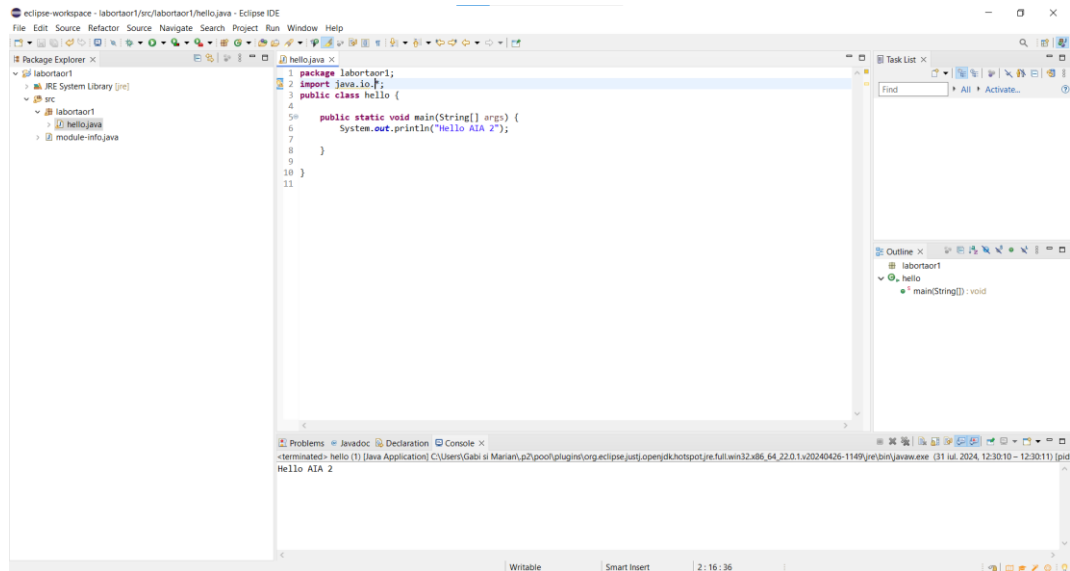


Figura 10. Execuția programului Java

În fereastra **Outline** este descrisă structura internă a clasei. Fereastra **Problems** conține erorile de compilare, dacă există.

Pentru opțiuni avansate pentru execuția unui program Java se deschide din meniul **Run**, comanda **Run Configurations...** sau prin click dreapta pe clasa din proiect în fereastra “Package Explorer” și se alege **Run As ->Run Configurations...**(Figura 11a).

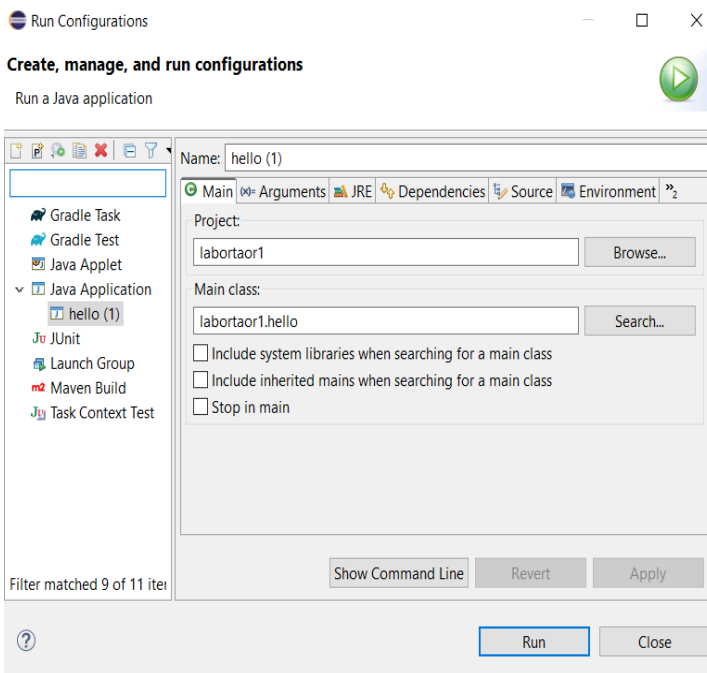


Figura 11a Opțiuni avansate

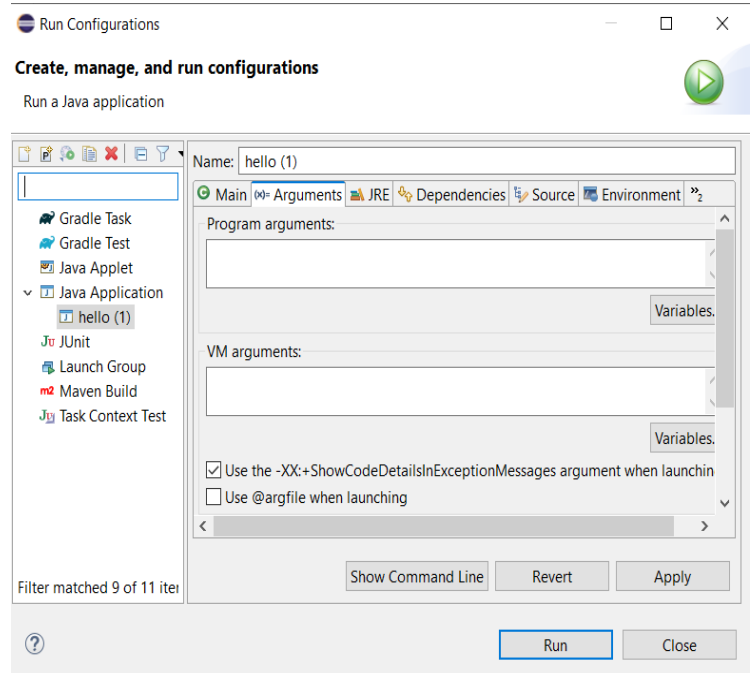


Figura 11b Opțiuni avansate

LUCRAREA NR. 1

În această fereastră se pot schimba/adăuga oricare dintre următoarele: argumente JVM, argumente în linia de comandă (Figura 11b), setări Classpath, variabile de mediu, JRE utilizat.

Perspectiva debug

Pentru depanarea aplicațiilor Java se alege modul de vizualizare (perspectiva) Debug, selectând din meniul **Window**, comanda **Open Perspective->Debug** (Figura 12).

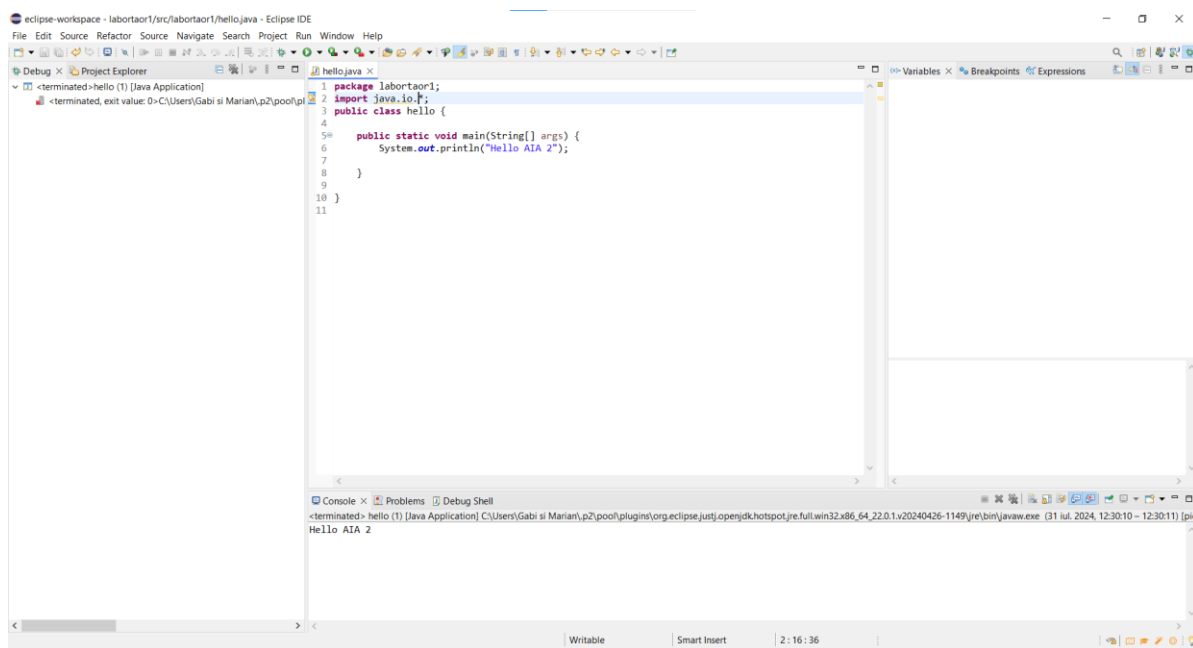


Figura 12 Perspectiva Debug

În fereastra Debug identificați următoarele:

- modalitatea de adăugare a punctelor de întrerupere (Breakpoints) pentru oprirea execuției/începerea execuției de la o anumită linie a codului
- monitorizarea valorilor expresiilor folosind fereastra de urmărire (Watch)
- modul de execuție pas cu pas al programului, folosind comenzile Step-Into, Step-Over.

6. Execuția unei aplicații Java din linia de comandă

Pașii ce trebuie parcurși pentru execuția aplicației **Welcome** sunt prezentați mai jos:

- Editarea programului într-un editor de texte;
- Salvarea programului sub numele **Welcome.java**, unde Welcome este numele clasei ce conține metoda `main()`. Reamintim că într-un program Java trebuie să existe o singură clasă care să conțină o metodă `main()`. Cu alte cuvinte, numele clasei trebuie să coincidă cu numele fișierului.

LUCRAREA NR. 1

- Cu ajutorul comenzii DOS **cd cale** se schimbă directorul curent cu cel în care s-a salvat fișierul `Welcome.java`.
- Compilarea programului se face cu ajutorul comenzii: **javac Welcome.java**
- Execuția programului se face cu ajutorul comenzii:
java Welcome

7. Explicatii cod sursa

7.1. Package

Un pachet (package) în Java este un mecanism pentru a organiza clasele și interfețele în module funcționale și logice, astfel încât să fie mai ușor de gestionat și de utilizat. Pachetele sunt folosite pentru a evita conflictele de nume și pentru a controla accesul la clasele și interfețele definite în ele. Iată o descriere detaliată.

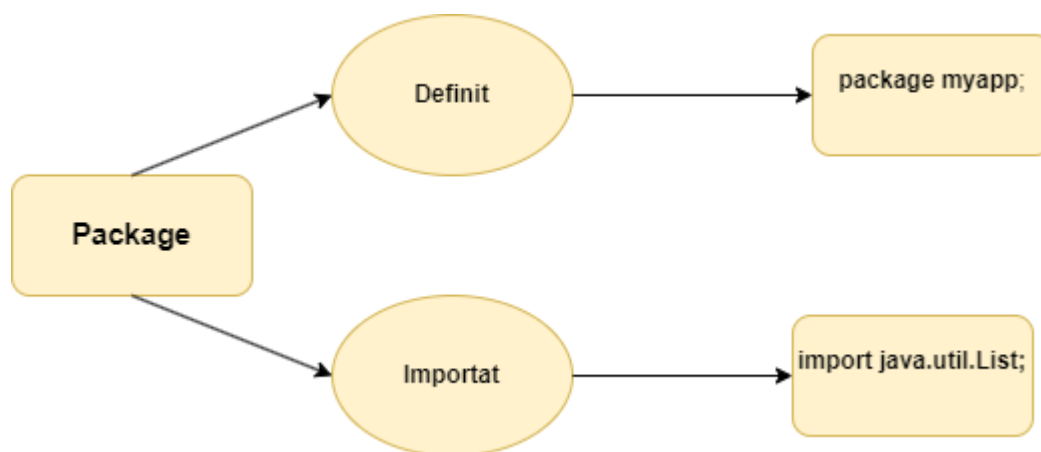


Figura 13. Implementarea pachetelor în Java

7.2. Pachetul **java.io.***

Pachetul `java.io.*` din Java este un pachet fundamental care oferă o gamă completă de clase și interfețe necesare pentru a efectua operațiuni de intrare/ieșire (I/O), cum ar fi citirea și scrierea de date în și din fișiere, manipularea fluxurilor de date (streams), și comunicarea prin canale, oferind astfel funcționalități esențiale pentru dezvoltarea aplicațiilor care

LUCRAREA NR. 1

necesită manipularea eficientă și sigură a datelor. Pachetul include clase importante precum `'File'`, `'FileInputStream'`, `'FileOutputStream'`, `'BufferedReader'`, și `'BufferedWriter'`, care facilitează crearea, citirea, scrierea și manipularea fișierelor și fluxurilor de date în diverse moduri, suportând atât operațiuni orientate pe bytes, cât și pe caractere, și oferind mecanisme pentru gestionarea excepțiilor și a erorilor care pot apărea în timpul operațiunilor I/O.

- Clasa `File` reprezintă un fișier sau un director (folder) din sistemul de fișiere. Aceasta permite operațiuni precum crearea, ștergerea, redenumirea fișierelor și directoarelor, precum și obținerea informațiilor despre ele (de exemplu, dimensiune, permisiuni, existență).
- Clasa `FileInputStream` este utilizată pentru a citi date binare (bytes) dintr-un fișier. Este folosită atunci când lucrăm cu date neprocesate, cum ar fi imagini, fișiere audio, sau date binare.
- Clasa `FileOutputStream` este utilizată pentru a scrie date binare (bytes) într-un fișier. Este folosită pentru a salva date neprocesate.
- Clasa `BufferedReader` este utilizată pentru a citi text dintr-un flux de intrare (de obicei un fișier) într-un mod eficient, folosind un buffer. Aceasta permite citirea liniilor complete de text, nu doar caracter cu caracter.
- Clasa `BufferedWriter` este utilizată pentru a scrie text într-un fișier într-un mod eficient, folosind un buffer. Aceasta permite scrierea liniilor complete de text.

Ce este un Buffer?

Un buffer este o zonă de memorie temporară utilizată pentru a stoca date pe măsură ce sunt transferate de la un loc la altul, în mod special pentru a gestiona diferențele de viteză dintre diferite componente ale sistemului sau pentru a face operațiile de I/O mai eficiente.

LUCRAREA NR. 1

1. Afisare Numar

```

1 package labortaor1; // Declaratia pachetului în care se află clasa. Asigură-te că este corectă (de obicei ar fi "laborator1" în loc de
2
3 import java.io.*; // Importă clasele necesare pentru citirea de la tastatură și manipularea I/O.
4
5 public class Afisare {
6
7     // Metoda principală a programului, punctul de intrare în aplicație.
8     public static void main(String[] args) throws IOException {
9
10        // Crează un obiect InputStreamReader care va citi date de la sistemul de intrare (tastatură).
11        InputStreamReader isr = new InputStreamReader(System.in);
12
13        // Crează un obiect BufferedReader pentru a citi textul de la InputStreamReader.
14        BufferedReader br = new BufferedReader(isr);
15
16        // Afisează un mesaj pe consolă cerând utilizatorului să introducă un număr natural.
17        System.out.println("Introduceți un numar natural: ");
18
19        // Citeste linia de text introdusă de utilizator.
20        String s = br.readLine();
21
22        // Converteste linia de text într-un număr întreg (int).
23        int n = Integer.parseInt(s);
24
25        // Afisează numărul întreg pe consolă.
26        System.out.println(n);
27    }
28 }

```

2. Suma primelor n nr naturale

```

hello.java  *sumanrnaturale.java ×
1 // Definirea pachetului 'laborator1'
2 package laborator1;
3 // Importarea tuturor claselor din pachetul java.io
4 import java.io.*;
5 // Definirea clasei publice 'sumanrnaturale'
6 public class sumanrnaturale {
7     // Punctul de intrare al programului Java
8     public static void main(String[] args) throws IOException {
9         // Crearea unui InputStreamReader pentru a citi date de la tastatură
10        InputStreamReader isr = new InputStreamReader(System.in);
11        // Crearea unui BufferedReader pentru a citi text într-un mod eficient
12        BufferedReader br = new BufferedReader(isr);
13        // Afisarea unui mesaj pentru a cere utilizatorului să introducă un număr natural
14        System.out.println("Introduceți un numar natural: ");
15        // Citirea unei linii de text de la tastatură
16        String s = br.readLine();
17        // Conversia liniei de text într-un număr întreg
18        int n = Integer.parseInt(s);
19        // Inicializarea variabilei pentru suma numerelor naturale
20        int suma = 0;
21        // Calcularea sumei numerelor naturale de la 0 până la n
22        for (int i = 0; i <= n; i++) {
23            suma += i;
24        }
25        // Afisarea rezultatului
26        System.out.println("Suma numerelor naturale este: " + suma);
27    }
28 }
29

```

LUCRAREA NR. 1

3. Suma Cifrelor

```

1 package labortaor1;
2
3 //Importarea tuturor claselor din pachetul java.io pentru operatiuni de I/O
4 import java.io.*;
5
6 //Definirea clasei publice SumaCifre
7 public class SumaCifre {
8     // Punctul de intrare al programului Java
9     public static void main(String[] args) throws IOException {
10         // Crearea unui InputStreamReader pentru a citi date de la tastatură
11         InputStreamReader isr = new InputStreamReader(System.in);
12         // Crearea unui BufferedReader pentru a citi text într-un mod eficient
13         BufferedReader br = new BufferedReader(isr);
14         // Afisarea unui mesaj pentru a cere utilizatorului să introducă un număr natural nenul
15         System.out.println("Introduceti un numar natural nenul:");
16         // Citirea unei linii de text de la tastatură
17         String s = br.readLine();
18         // Conversia liniei de text într-un număr întreg
19         int n = Integer.parseInt(s);
20         // Afisarea sumei cifrelor numărului utilizând metoda SumaCifreNumar
21         System.out.println("Suma cifrelor numarului este: " + SumaCifreNumar(n));
22         // Afisarea numărului obținut prin inversarea cifrelor sale utilizând metoda InvCifreNumar
23         System.out.println("Numarul obținut prin inversarea cifrelor sale este: " + InvCifreNumar(n));
24         // Afisarea descompunerii numărului în factori primi utilizând metoda FactoriPrimi
25         System.out.println("Descompunerea numarului in factori primi este: " + FactoriPrimi(n));
26     }
27     // Metodă statică pentru calcularea sumei cifrelor unui număr
28     public static int SumaCifreNumar(int nn) {
29         int suma = 0;
30         // Calcularea sumei cifrelor
31         while (nn != 0) {
32             suma += nn % 10;
33             nn /= 10;
34         }
35         return suma;
36     }

```

LUCRAREA NR. 1

```
// Metodă statică pentru inversarea cifrelor unui număr
public static int InvCifreNumar(int nn) {
    int invnn = 0;
    // Inversarea cifrelor
    while (nn != 0) {
        invnn = invnn * 10 + nn % 10;
        nn /= 10;
    }
    return invnn;
}

// Metodă statică pentru descompunerea unui număr în factori primi
public static String FactoriPrimi(int nn) {
    int i = 2;
    String s = " ";
    // Descompunerea în factori primi
    do {
        int mult = 0;
        // Determinarea numărului de apariții al unui factor prim
        while (nn % i == 0) {
            mult++;
            nn /= i;
        }
        // Adăugarea factorului și a exponentei sale la șirul de rezultat
        if (mult != 0)
            s = s + i + "^" + mult + " ";
        i++;
    } while (nn != 1);
    return s;
}
}
```