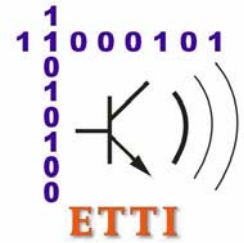


Catedra de Telecomunicatii

10/11/2007

Software pentru Retele de Telecomunicatii (SwRTc)

Laborator 1

Introducere in programarea Java. Dezvoltarea si modificarea programelor. Scrierea functiilor membru (metodelor)

1.1. Descrierea laboratorului

In aceasta lucrare de laborator vor fi acoperite urmatoarele probleme:

- **Dezvoltarea unui program Java.** Studiu de caz: programul *Salut.java*
  - **Automatizarea** dezvoltarii prin utilizarea unui fisier script
  - **Depanarea** programelor Java (creare erori, urmarirea efectului, interpretarea mesajelor)
  - Generarea automata a documentatiei. Obtinerea documentatiei API
- **Analiza, conceperea si modificarea programelor Java**
  - **Analiza** unui program. Adaugarea **interactivitatii**. Utilizarea **tablourilor**
  - **Studiu de caz: calculul unui polinom**. Adaugarea **interactivitatii**. **Delegarea functionala**.
- **Instalarea *kitului* de dezvoltare** Java, **configurarea variabilelor de mediu** *PATH* si *CLASSPATH*
- **Teme de casa** si teme suplimentare.
- Anexa. Accesul la consola standard de intrare. Delegarea functionala

1.2. Dezvoltarea unui program Java. Studiu de caz: programul Salut.java

1.2.1. Conceperea si editarea programului *Salut.java*

Conceperea (proiectarea) programului inseamna **cel putin scrierea pe hartie a codului sau a unui pseudocod** (schita a programului scrisa in limbaj natural), **dar poate include si aplicarea unor instrumente** (diagrame, limbaje de modelare cum ar fi UML) **si metodologii** mai complicate**.**

O cale de a incepe conceperea programelor simple este de a crea un asa numit ***pseudocod***, adica o *descriere intr-un limbaj apropiat de cel natural* a ceea ce trebuie sa faca programul. De exemplu:

clasa *care ilustreaza elementele esentiale ale unui program Java, contine:*
metoda principala *(punctul de intrare in program), care:*
afiseaza pe ecran textul *“Buna ziua”*

Urmatorul pas poate fi *transformarea pseudocodului* in ***comentarii***.

```
/** Clasa care ilustreaza elementele esentiale ale unui program Java.
 *  Trebuie sa fie publica pentru ca are metoda principala.
 */
  /** Metoda principala (punct de intrare in program).
   *  Este prima metoda executata de JVM (Java Virtual Machine).
   *  Primeste ca parametri argumentele din lina de comanda.
   *  Nu returneaza nici o valoare. Trebuie sa fie ‘public static’
   */
    // Corpul metodei afiseaza textul "Buna ziua" pe ecran
```

In continuare, se pot utiliza **pasi succesivi** in care se adauga ***codul Java***, pornind de la nivelul cel mai inalt (urmand o **strategie *top-down*** – de la nivel inalt la nivel jos – prin detaliere):

- declaratia clasei:

```
public class Salut {
```

- adaugarea **declaratiilor metodelor** (in cazul nostru metoda **`main()`**) **si atributelor** (nu este cazul):

```
  public static void main(String[] args) {
    // Corpul metodei afiseaza textul "Buna ziua" pe ecran
  }
```

- si a **corpurilor metodelor** (in cazul nostru metoda **`main()`**):

```
System.out.println("Buna ziua"); // Afisarea unui text pe ecran
```

Codul final este urmatorul:

```
1   /** Clasa care ilustreaza elementele esentiale ale unui program Java.
2    *  Trebuie sa fie publica pentru ca are metoda principala.
3    */
4   public class Salut {
5     /** Metoda principala (punct de intrare in program).
6      *  Este prima metoda executata de JVM (Java Virtual Machine).
7      *  Primeste ca parametri argumentele din lina de comanda.
8      *  Nu returneaza nici o valoare. Trebuie sa fie 'public static'
9      */
10    public static void main(String[] args) {
11      System.out.println("Buna ziua"); // Afisarea unui text pe ecran
12    }
13  }
```

Daca **inlaturam comentariile**, codul ramas este urmatorul:

```
1   public class Salut {
2     public static void main(String[] args) {
3       System.out.println("Buna ziua");
4     }
5   }
```

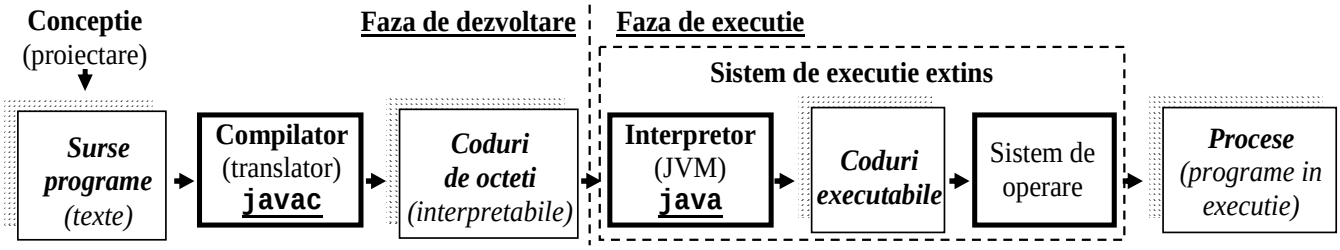
In laborator:

1. **Deschideti** editorul Notepad (sau alt editor de text simplu) si editati sau **copiati cele 5 linii de cod ale clasei Salut** (fara comentarii).
2. **Salvati** acest text (**codul sursa al clasei `Salut`**) intr-un **fisier** cu numele **`Salut.java`** intr-un director cu numele **`SwRTc2007\NumarulGrupeiSiSeria`** (de exemplu: **`SwRTc2007\421E`**).

Observatie: La **salvarea unui fisier cu extensia `.java` in Notepad** trebuie avut grija sa se selecteze "**All files**" in loc de "***.txt**", altfel fisierul va avea extensia **`.java.txt`**.

1.2.2. Compilarea programului *Salut.java*

In **sistemul de programare Java** *codurile sursa* sunt ***compilate*** (translatate) de la limbajul de programare Java la coduri executabile de procesorul software Java (JVM = *Java Virtual Machine*), numite coduri de octeti (*bytecodes*), pentru ca apoi *codurile de octeti* sa fie ***interpretate*** (executate de interpretorul Java, care este parte din JVM, si care ofera coduri executabile sistemului de operare).



Daca **se presupune utilizarea compilatorului Java din linia de comanda** (consola de intrare), atunci trebuie executata urmatoarea comanda, in directorul in care se afla fisierul *`Salut.java`*:

```
directorcurent> javac Salut.java
```

In urma acestei comenzi, **compilatorul Java va crea genera codul de octeti** corespunzator **intr-un fisier** cu numele **`Salut.class`**, in directorul in care se afla si fisierul `Salut.java`.

In laborator:

1. **Compilati codul sursa** al clasei **`Salut`** in directorul curent (**`SwRTc 2007\NumarulGrupeiSiSeria`**).

1.2.3. Lansarea programului *Salut.java*

Pentru **executia programului**, acesta trebuie **lansat in interpretor**, folosind comanda:

```
directorcurent> java Salut
```

Rezultatul va fi aparitia mesajului **`Buna ziua!`** pe ecranul monitorului.

In final, pe ecran poate fi urmatoarea secventa de informatii:

```
directorcurent> javac Salut.java

directorcurent> java Salut
Buna ziua!

directorcurent>
```

In laborator:

1. **Lansati programul** rezultat din clasa **`Salut`** (al carui cod de octeti se afla in fisierul `Salut.class`) in directorul curent.

1.2.4. Automatizarea dezvoltarii prin utilizarea unui fisier script

O simplificare a lansarii in executie a programului este **crearea unui fisier script cu extensia .bat** (in acelasi directorul cu sursa java), numit de exemplu **`Run_Salut.bat`**, **cu urmatorul continut:**

```
javac Salut.java
java Salut
pause
```

si lansarea lui in executie. Pe ecran se va obtine:

```
directorcurent> Run_Salut.bat
directorcurent> javac Salut.java
directorcurent> java Salut
Buna ziua!
directorcurent> pause
Press any key to continue...
```

Optional, in laborator:

1. Creati fisierul script `Run_Salut.bat` cu continutul de mai sus in directorul curent.
2. Executati scriptul in Windows Explorer, prin dublu click pe numele fisierului.

1.2.5. Depanarea programului *Salut.java*

Daca in urma compilarii sau executiei apar erori, ele trebuie corectate (urmarind si **indicatiile din mesajele de eroare**), revenind la etapa conceperii si editarii.

In laborator:

Produceti urmatoarele modificari in codul sursa (varianta de 5 linii, fara comentarii), **urmariti efectele lor** (in special eventualele **rapoarte de eroare la executie sau compilare**), si **faceti interpretari** ale acestora (***aceste rapoarte de eroare si interpretari vor forma prima parte a temei de casa!***).

Dupa fiecare modificare codul va fi recompilat (iar **in caz de succes executat din nou**).

Inaintea unei noi modificari va fi **mai intai restabilit programul initial**.

Raspunsurile la urmatoarele intrebari vor usura interpretarea rezultatelor :

- *Ce se afiseaza pe ecran?*
- *Ce fel de problema apare (eroare de compilare, exceptie in timpul executiei, niciuna)?*
- *Care este cauza probabila?*
- *Cat de sugestiv este mesajul care apare?*
- *Ce concluzii se pot trage?*

I. Se va elimina prima acolada (din linia 1)

Exemplu de rezolvare: La compilare, pe ecran se afiseaza:

```
Salut.java:1: '{' expected
public class Salut
                ^
1 error
```

Problema: eroare de compilare. **Cauza**: compilatorul sesizeaza inexistenta acoladei deschise dupa numele clasei, acolada care marcheaza inceputul corpului clasei. Mesajul e **destul de sugestiv** incat sa ne permita corectarea rapida a erorii. **Concluzii**: …

II. Se va elimina acolada din linia 2

Exemplu de rezolvare: La compilare, pe ecran se afiseaza:

```
Salut.java:2: ';' expected
    public static void main(String[] args)
                                           ^
Salut.java:5: 'class' or 'interface' expected
}
^
Salut.java:6: 'class' or 'interface' expected
^
3 errors
```

Problema: eroare de compilare. **Cauza**: compilatorul sesizeaza inexistenta acoladei deschise dupa numele metodei main, acolada care marcheaza inceputul corpului metodei. **Prima parte a mesajului este utila pentru a corecta eroarea**, dar **restul mesajului poate produce confuzie in primul** moment. Dupa corectarea erorii din linia doi, “erorile” 2 si 3 “dispar”. **Concluzii**: Apare un fenomen de “**propagare a erorilor**”, lipsa acoladei respective putand avea mai multe interpretari.

III. Se va elimina simbolul punct si virgula (`;`) din linia 3

IV. Se vor elimina parantezele drepte, `[]`, din linia 2

V. Se va elimina cuvantul cheie `void` (din linia 2)

VI. Se va elimina cuvantul cheie **`static`** (din linia 2)

Exemplu de rezolvare: Dupa compilare, in momentul executiei, pe ecran se afiseaza:

```
Exception in thread "main" java.lang.NoSuchMethodError: main
```

Problema: situatie exceptionala (**`java.lang.NoSuchMethodError`**) in timpul executiei. **Cauza**: Masina Virtuala Java (JVM), cea care sta in spatele interpretorului **`java`** sesizeaza inexistenta unei metode Java cu semnatura completa: `public static void main(String[] args)` Mesajul **produce confuzie in primul** moment, deoarece da impresia ca metoda nu exista, si nu sugereaza cauza situatiei exceptionale. **Concluzii**: ***Este strict necesara declararea metodei `main()` ca fiind de tip `static`. Este recomandabila retinerea simptomelor acestei situatii exceptionale!***

VII. Se vor inlocui ghilimelele cu apostroafe in linia 3

VIII. Se va inlocui numele clasei **`System`** cu **`system`** in linia 3

IX. Se va inlocui numele clasei **`Salut`** cu **`salut`** in linia 1

X. Se va inlocui **`args`** din linia 2 cu cuvantul **`argumente`**

1.2.6. Generarea automata a documentatiei

O facilitate suplimentara, importanta, oferita de kitul de dezvoltare Java este generatorul de documentatie Java. Delimitatorii **`/**`** si **`*/`** sunt folositi pentru a arata ca textul trebuie tratat ca un comentariu de catre compilator, dar de asemenea ca textul este parte din documentatia clasei care poate fi generata folosind utilitarul **javadoc**.

Pentru generarea documentatiei unui program, se foloseste comanda:

```
directorcurent> javadoc <NumeClasa>.java
```

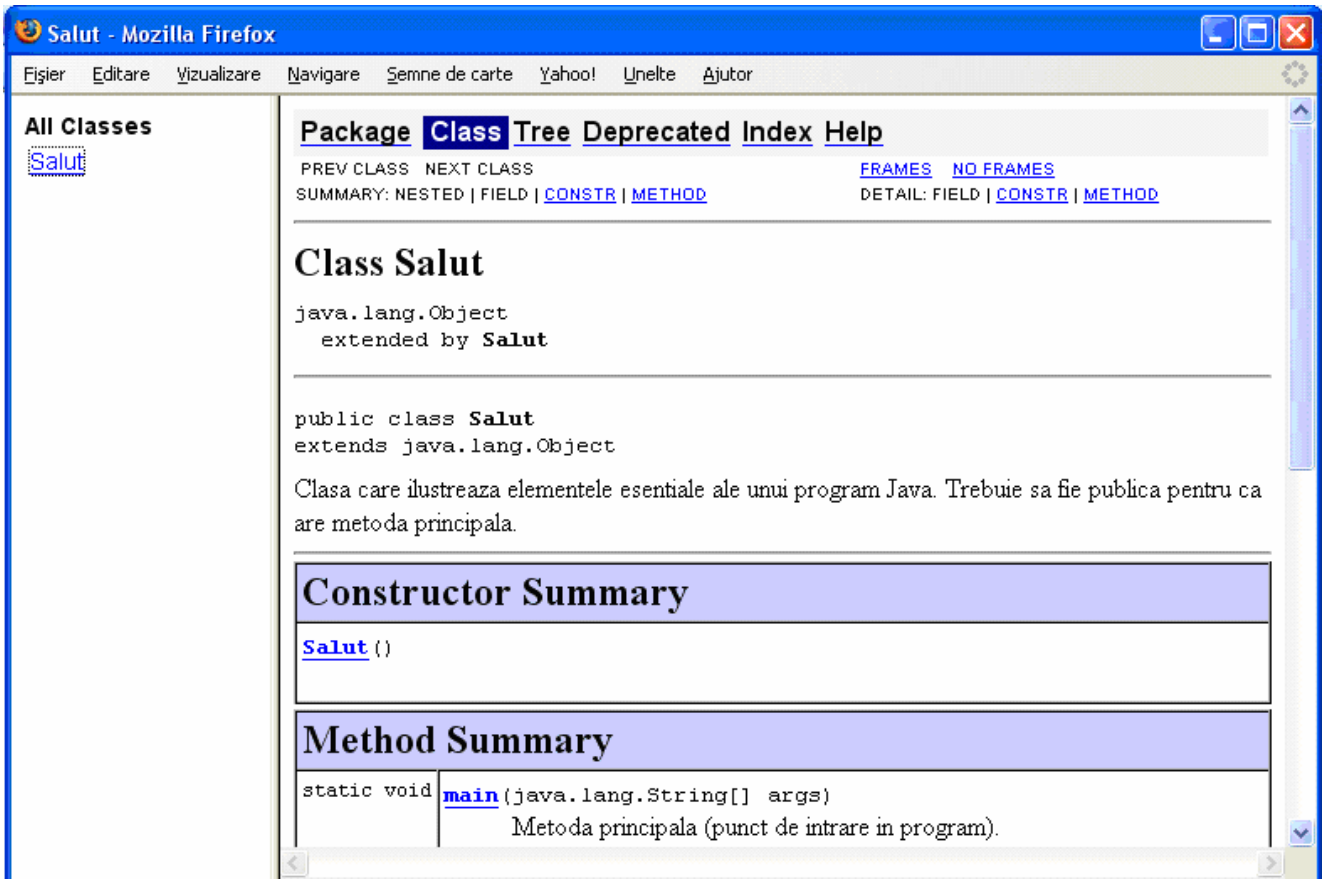
De exemplu, daca generam documentatia pentru programul Salut:

```
 1   /** Clasa care ilustreaza elementele esentiale ale unui program Java.
 2    *  Trebuie sa fie publica pentru ca are metoda principala.
 3    */
 4   public class Salut {
 5     /** Metoda principala (punct de intrare in program).
 6      *  Este prima metoda executata de JVM (Java Virtual Machine).
 7      *  Primeste ca parametri argumentele din lina de comanda.
 8      *  Nu returneaza nici o valoare. Trebuie sa fie 'public static'
 9      */
10     public static void main(String[] args) {
11       System.out.println("Buna ziua"); // Afisarea unui text pe ecran
12     }
13   }
```

cu comanda **javadoc:**

```
directorcurent>javadoc  Salut.java
Loading source file Salut.java...
Constructing Javadoc information...
Standard Doclet version 1.5.0_07
Building tree for all the packages and classes...
Generating Salut.html...
Generating package-frame.html...
Generating package-summary.html...
Generating package-tree.html...
Generating constant-values.html...
Building index for all the packages and classes...
...
```

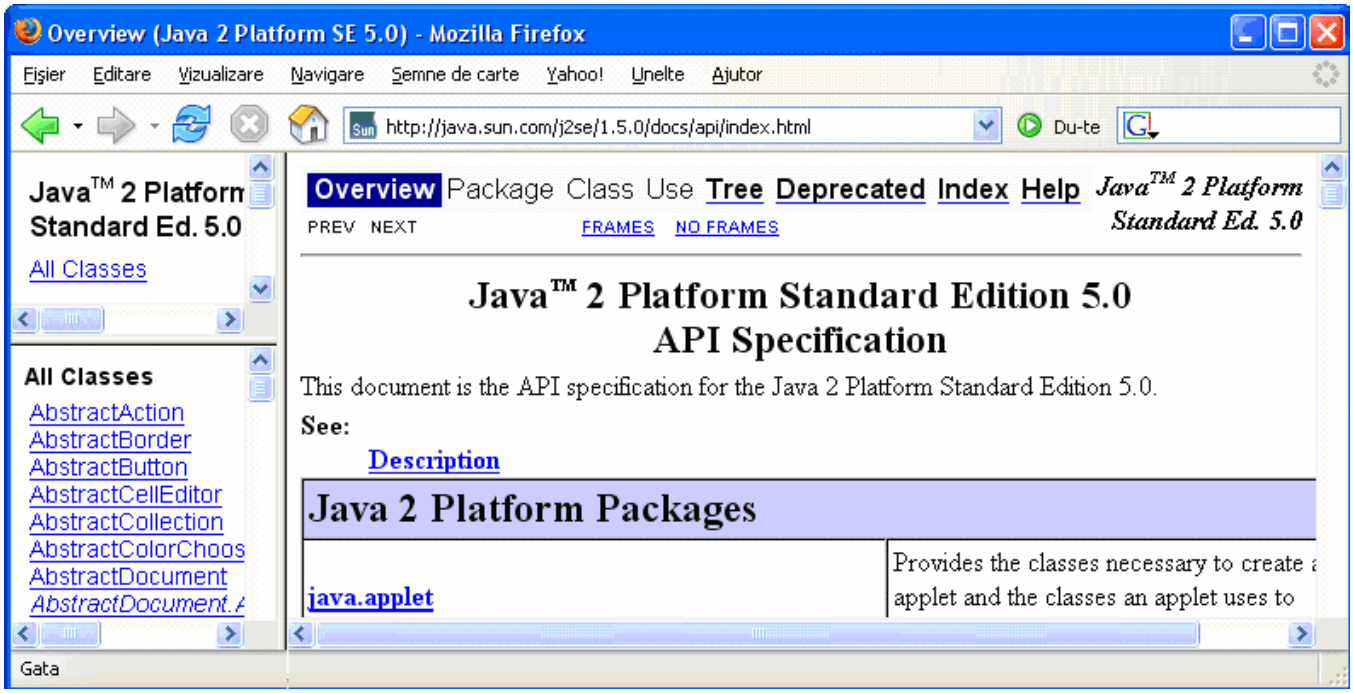
sunt create pagini Web care arata astfel:



Acesta este si **formatul sub care pot fi gasite documentatiile claselor de biblioteca Java.**

Documentatia Java (***Java SE 5.0 Documentation***), poate fi **accesata online** la adresa http://java.sun.com/j2se/1.5.0/**docs**/index.html.

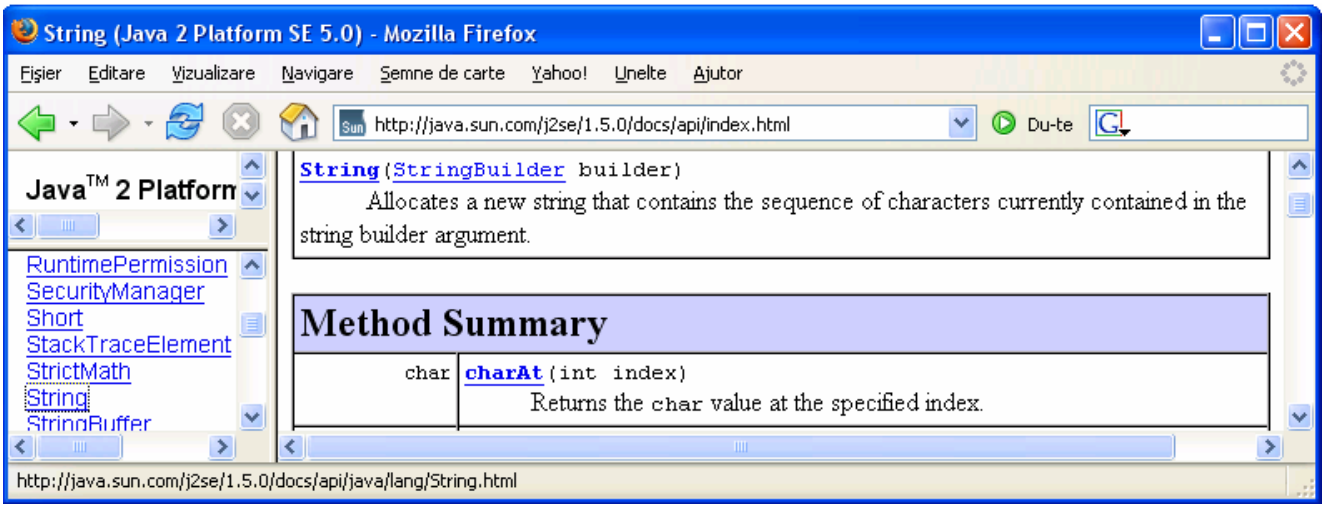
Specificatia API (documentatia claselor bibliotecii standard) poate fi **accesata online** la adresa http://java.sun.com/j2se/1.5.0/**docs/api**/index.html.



Daca **se selecteaza pachetul `java.lang`** in subfereastra din **stanga sus si apoi** in subfereastra din **stanga jos**, se obtine urmatorul continut:

Daca **se selecteaza clasa `String`** in subfereastra din **stanga jos**, se obtine urmatorul continut:

Daca **se navigheaza prin clasa `String`** in subfereastra din **dreapta jos**, se gasesc documentatiile campurilor (atributelor), constructorilor si metodelor:



Documentatia Java poate fi **descarcata sub forma unei arhive .zip din** sectiunea **J2SE 5.0 Documentation** de la adresa http://java.sun.com/javase/downloads/index_jdk5.jsp. Din sectiunea **J2SE 5.0 JDK Source Code** poate fi descarcat si vizualizat si **codul sursa al tuturor claselor Java**.

Documentatia va fi plasata in directorul **`C:\Program Files\Java\jdk1.5.0_12\docs`**.

Pe langa documentatia de baza, se poate **accesa online tutorialul Java oferit de Sun**, aflat la adresa **http://java.sun.com/docs/books/tutorial/java/TOC.html.**

Acesta si exemplele lui pot fi descarcate sub forma de arhive **.zip** (**tutorial.zipb, etc.**) de la adresa *http://java.sun.com/docs/books/tutorial/information/download.html*.

Alte referinte Web interesante:

Bruce Eckel, **Thinking in Java, 3rd ed. Rev. 4.0**, **http://www.faqs.org/docs/think_java/TIJ3_c.htm**

- PDF: *http://www.planetpdf.com/codecuts/pdfs/eckel/TIJ3.zip* (beta),
- HTML and source code: *http://www.mindviewinc.com/downloads/TIJ-3rd-edition4.0.zip*)

Kevin Taylor, **Java Progr. Tutorial**, http://java.about.com/od/beginningjava/a/beginjavatutor.htm

1.3. Analiza, conceperea si modificarea programelor Java

1.3.1. Analiza unui program simplu

Urmatorul program, **`SalutUtilizator0.java`**, il saluta pe utilizator folosind ca nume al utilizatorului **un argument pasat programului de la lansare**.

```
 1   public class SalutUtilizator0 {
 2     public static void main(String[] args) {
 3       String modSalut = "Salut ";     // variabila locala metodei (functiei)
 4
 5       // Obtinerea numelui utilizatorului
 6       String nume = args[0];          // utilizarea unui argument
 7
 8       // Salutul
 9       System.out.println(modSalut + nume + "!");    // concatenare
10     }
11   }
```

In laborator:

1. **Creati fisierul** sursa **`SalutUtilizator0.java`** cu continutul de mai sus in directorul curent.
2. **Executati urmatoarele comenzi** in consola, in directorul curent:
   `> javac SalutUtilizator0.java`
   `> java SalutUtilizator0 <numele studentului care executa programul>`

folosind un nume simplu, care sa nu includa spatii (exemplu: **`Xulescu`**).

3. **Executati comenzile de la pct. 2** cu un **nume care sa includa spatii** (exemplu: **`Xulescu Ygrec`**). Ce observati?
4. **Executati comenzile de la pct. 2** cu un nume care **sa includa spatii dar care sa fie incadrat de ghilimele** (exemplu: **`"Xulescu Ygrec"`**). Ce se schimba?

In laborator:

1. **Analizati codul** clasei **`SalutUtilizator0`**.
2. **Care sunt elementele de noutate fata de codul** clasei **`Salut`**?

Variabilele locale sunt acele variabile **declarate in interiorul blocurilor de cod** (cuprinse intre acolade), sunt **create in momentul declararii lor** si sunt **accesibile din locul declararii si pana la sfarsitul blocului de cod in care au fost declarate** (pentru variabilele referinta in momentul declararii este creata doar referinta, obiectul referit fiind creat dinamic cu **`new`**).

Concatenarea sirurilor de caractere **`modSalut`** si **`nume`** este realizata cu ajutorul **operatorului `"+"`.**

1.3.2. Modificarea programului anterior prin adaugarea interactivitatii

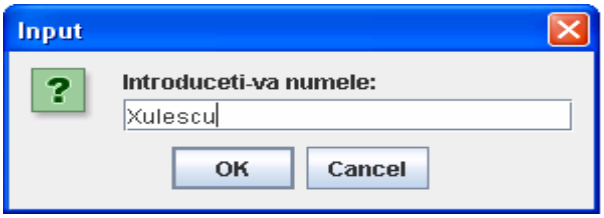
Fereastra de dialog este cel mai simplu si elegant **mod de obtinere a unei intrari de la utilizator** atunci cand este nevoie. Metodei **`showInputDialog()`** i se ofera ca **parametru mesajul catre utilizator**, iar metoda **returneaza intrarea de la utilizator** ca sir de caractere (obiect **`String`**).

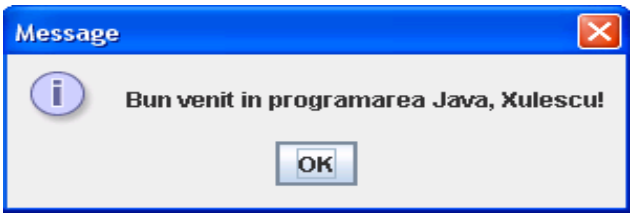
Pentru **a obtine de la utilizator o valoare de tip sir de caractere (`String`)** se foloseste sintaxa:

```
// Obtinerea numelui utilizatorului folosind fereasta de dialog
String nume = JOptionPane.showInputDialog("Introduceti-va numele:");
```

Pentru **a se afisa un mesaj catre utilizator** se foloseste sintaxa:

```
// Afisarea unui mesaj catre utilizator folosind fereasta de dialog
JOptionPane.showMessageDialog(null, "Bun venit in lumea Java, Xulescu!");
```





Ambele situatii impun importul clasei `JOptionPane` din pachetul de clase de biblioteca grafice `javax.swing`, **sau a intregului pachet de clase de biblioteca `javax.swing`**.

Pentru **a se importa clasa** `JOptionPane` din pachetul de clase `javax.swing` se foloseste sintaxa:

```
// Importul unei clase se declara inaintea declaratiei clasei
import javax.swing.JOptionPane;
```

Pentru **a se importa pachetul de clase** `javax.swing` se foloseste sintaxa:

```
// Importul unui pachet de clase se declara inaintea declaratiei clasei
import javax.swing.*;
```

Urmatorul program, **`SalutUtilizator1.java`**, il saluta pe utilizator cu **numele pe care il obtine interactiv direct de la acesta**. Programul are un mod de salut diferit de cel al programului anterior.

```
 1  import javax.swing.*; // Importul unui pachet/biblioteca de clase grafice
 2
 3  public class SalutUtilizator1 {
 4    public static void main(String[] args) {
 5      String modSalut = "Bun venit in lumea Java, ";
 6
 7      // Obtinerea numelui utilizatorului
 8      String nume = JOptionPane.showInputDialog("Introduceti-va numele: ");
 9
10      // Salutul
11      String text = modSalut + nume + "!";
12      JOptionPane.showMessageDialog(null, text);
13    }
14  }
```

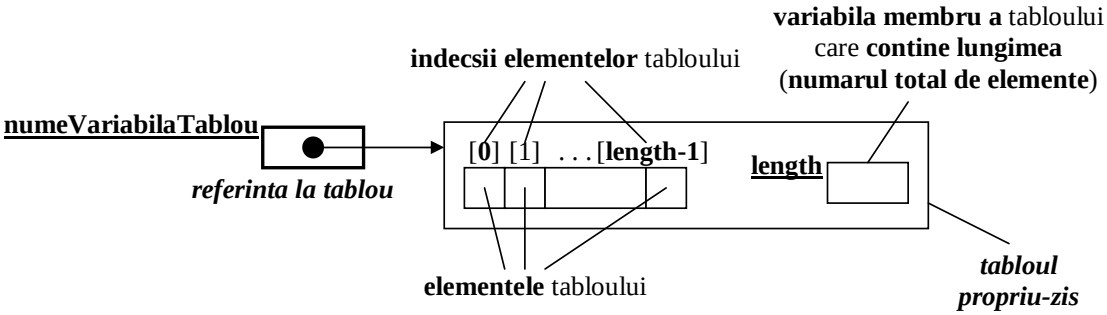
In laborator:

1. **Creati fisierul** sursa **`SalutUtilizator1.java`** cu continutul de mai sus in directorul curent.
2. **Compilati si executati programul** in consola, in directorul curent.

Ca alternativa la utilizarea ferestrelor de dialog se poate utiliza **perechea de console standard**, tastatura si ecranul in mod text (vezi in **anexa** sunt **programele** **`SalutUtilizator2.java`**, si **`SalutUtilizator2.java`** care folosesc varianta cu clasa **`Scanner`** din pachetul **`java.util`**, si varianta cu clasele **`BufferedReader`** si **`InputStreamReader`** din pachetul **`java.io`**).

1.3.3. Utilizarea tablourilor in Java

Un tablou Java este **o structura** care **contine mai multe valori de acelasi tip**, numite **elemente**. **Lungimea** unui tablou este fixa, **stabilita in momentul crearii tabloului** (cu operatorul **`new`**).



Urmatorul program, **`SumaArgumenteIntregi.java`**, calculeaza **suma valorilor pasate ca argumente la lansarea programului**.

```
 1   public class SumaArgumenteIntregi {
 2     public static void main(String[] args)  {
 3       System.out.println("Au fost primite " + args.length + " argumente");
 4
 5       if (args.length > 0) {
 6         int suma = 0;
 7         for (int index = 0; index < args.length; index++) {
 8           suma = suma + Integer.parseInt(args[index]);
 9         }
10        System.out.println("Suma valorilor primite este " + suma);
11       }
12       else {
13         System.out.println("Utilizare tipica:");
14         System.out.println("\t java SumaArgumenteIntregi 12 31 133 -10");
15       }
16     }
17   }
```

In laborator:

1. **Analizati codul** clasei **`SumaArgumenteIntregi`**.
2. **Ce instructiuni recunoasteti** si **ce rol au** ele? **Ce elemente din program** va sunt **necunoscute**?

Pentru **a obtine numarul de elemente ale unui tablou** se foloseste:

```
// Obtinerea dimensiunii tabloului de argumente pasate de utilizator
int numarArgumentePasateDeUtilizator = args.length;
```

Pentru **a converti o valoare de la `String` la `int`** se foloseste sintaxa:

```
// Conversia unei valori de la tip int la tip String
int numarStudenti = Integer.parseInt("25");
```

In laborator:

1. **Creati fisierul** sursa **`SumaArgumenteIntregi.java`** cu continutul de mai sus in directorul curent.
2. **Compilati si executati programul** in consola, in directorul curent, **fara sa ii oferiti argumente**.
3. **Executati apoi programul** oferindu-i **3 argumente intregi**.
4. **Executati apoi programul** oferindu-i **un singur argument 'a'**. Ce observati?

1.4. Studiu de caz mai complex: program de calcul al unui polinom

1.4.1. Specificatia initiala (varianta cu valori prestabilite)

Se doreste scrierea unui program Java numit **`Polinom0`**, care:
- sa **creeze un polinom** cu grad prestabilit (4) si coeficienti de **valori prestabilite** (1, 2, 3, 2, 1),
- sa **afiseze polinomul**,
- sa **calculeze valoarea polinomului** pentru **o anumita valoare prestabilita** a necunoscutei (2), si
- sa **afiseze aceasta valoare**.

Un posibil pseudocod al programului este urmatorul:

```
1   public class Polinom0 {
2
3     public static void main(String[] args) {
4       // Declararea si initializarea variabilei intregi,
5       // care contine gradul polinomului, N=4
6
7       // Crearea tabloului coeficientilor (de dimensiune N+1), numit C,
8       // folosind valorile prestabilite : 1, 2, 3, 2, 1
9
10      // Afisarea polinomului P(X)
11      // - mai intai termenul liber Co
12      // - apoi termenii Ci*X^i, unde i=1,N
13
14      // Declararea si initializarea variabilei intregi numita necunoscuta,
15      // care contine valoarea necunoscutei, X=2
16
17      // Afisarea valorii necunoscutei (X)
18
19      // Declararea si initializarea variabilei intregi numita P,
20      // care contine valoarea polinomului, P(X)
21
22      // Calculul polinomului P(X) = suma(Ci * X^i), unde i=0,N
23         // - calculul valorii X^i, unde i=1,N
24         // - actualizarea valorii polinomului
25
26      // Afisarea valorii polinomului P(X)
27    }
28  }
```

Pentru simplitate, la inceput sa ne ocupam doar de prima sectiune a programului, in care **se creeaza polinomul** (prin gradul si coeficientii sai) **si se afiseaza rezultatul**:

```
1   public class Polinom0 {
2
3     public static void main(String[] args) {
4       // Declararea si initializarea variabilei intregi,
5       // care contine gradul polinomului, N=4
6
7       // Crearea tabloului coeficientilor (de dimensiune N+1), numit C,
8       // folosind valorile prestabilite : 1, 2, 3, 2, 1
9
10      // Afisarea polinomului P(X)
11      // - mai intai termenul liber Co
12      // - apoi termenii Ci*X^i, unde i=1,N
13
14      // Cod ce urmeaza a fi scris ulterior
15    }
16  }
```

In laborator:

1. **Creati fisierul** sursa **`Polinom0.java`** cu **continutul de mai sus (adica inclusiv comentariile!!!)** in directorul curent.
2. **Adaugati sub fiecare grup de comentarii codul necesar (*Acest cod este parte a temei de casa!*)**:
- inspirandu-va din exemplul anterior (**`SumaArgumenteIntregi`** pentru lucrul cu tabloul) si
- **folosind indicatiile care urmeaza**.

Formatul afisarii coeficientilor polinomului in laborator este la libera alegere a studentilor.

3. **Compilati si executati programul** in consola, in directorul curent. Corectati eventualele erori.

Tema suplimentara de dificultate sporita:

Studentii care doresc pot incerca sa scrie codul pentru afisarea polinomului in formatul:

```
P(X) = 1 + 2*X^1 + 3*X^2 + 2*X^3 + 1*X^4
```

ca parte suplimentara a temei de casa (pentru un bonus).

Indicatii:

- pentru a se **crea un tablou** cu valorile `1, 2, 3, 4, 3, 2, 1` se foloseste **sintaxa simplificata**:

```
// Crearea unui tablou de 7 valori intregi, varianta simplificata
int[] tab = { 1, 2, 3, 4, 3, 2, 1 };
```

- acelasi efect se obtine folosind **sintaxa complexa pentru crearea unui tablou**:

```
// Crearea unui tablou de 7 valori intregi, varianta complexa
int[] tab = new int[7];    // declararea variabilei si alocarea memoriei
tab[0]= 1;                 // popularea tabloului
tab[1]= 2;                 // popularea tabloului
tab[2]= 3;                 // popularea tabloului
tab[3]= 4;                 // popularea tabloului
tab[4]= 3;                 // popularea tabloului
tab[5]= 2;                 // popularea tabloului
tab[6]= 1;                 // popularea tabloului
```

Acum ca prima parte a codului este functional sa ne ocupam si de **ultima parte a programului**, in care **se obtine necunoscuta, se calculeaza valoarea polinomului si se afiseaza rezultatul**:

```
 1  public class Polinom0 {
 2
 3    public static void main(String[] args) {
 4      // Cod ce a fost scris anterior
 5
 6      // Declararea si initializarea variabilei intregi numita necunoscuta,
 7      // care contine valoarea necunoscutei, X=2
 8
 9      // Afisarea valorii necunoscutei (X)
10
11      // Declararea si initializarea variabilei intregi numita P,
12      // care contine valoarea polinomului, P(X)
13
14      // Calculul polinomului P(X) = suma(Ci * X^i), unde i=0,N
15         // - calculul valorii X^i, unde i=1,N
16         // - actualizarea valorii polinomului
17
18      // Afisarea valorii polinomului P(X)
19    }
20  }
```

In laborator:

1. Continuati scrierea codului clasei **`Polinom0`**, **adaugand sub fiecare grup de comentarii de mai sus codul necesar**, si **folosind indicatiile care urmeaza**. ***Acest cod este parte a temei de casa!***
2. **Compilati si executati programul** in consola, in directorul curent. **Corectati eventualele erori.**

Indicatii:

- pentru a calcula ***X^i*** (in variabila **xLaI**) se poate folosi functia matematica **pow()** din biblioteca matematica **Math** aflata in pachetul de clase **java.lang** (pachet care contine si **String** si **System** si nu necesita importul),

- metoda **pow()** **primeste doua argumente (primul este valoarea de ridicat la putere iar al doilea este puterea)** si care **returneaza o valoare de tip** **double**,

- de aceea este necesara **conversia explicita (cast) a valorii returnate la tipul** **int**.

```
      int xLaI;
      for (int i=0; i<=N; i++) {
         // - calculul valorii X^i, unde i=1,N
         xLaI = (int) Math.pow(X, i);
      }
```

Se observa ca in aceasta forma, **la fiecare executie programul va genera aceleasi iesiri.**

1.4.2. Modificarea programului anterior prin adaugarea interactivitatii

Pornind de la programul **Polinom0** sa se scrie un program numit **Polinom1**, care:
- **sa creeze un polinom** cu grad si coeficienti avand **valorile obtinute de la utilizator,**
- **sa afiseze polinomul**,
- **sa calculeze valoarea polinomului** pentru o **valoare** a necunoscutei **specificata de utilizator**, si
- **sa afiseze aceasta valoare**.

Programul **Polinom1** reprezinta o **generalizare a programului anterior** (versiune mai flexibila), deoarece gradul polinomului, valorile coeficientilor si necunoscutei sunt **obtinute de la utilizator**. In aceasta forma, **la fiecare executie programul poate genera alte iesiri, in functie de valorile de intrare**.

In pseudocodul care urmeaza comentariile scrise intensificat (**bold**) reprezinta **elementele de noutate fata de programul anterior**.

```
 1   import javax.swing.JOptionPane;
 2
 3   public class Polinom1 {
 4     public static void main(String[] args) {
 5       // Declararea variabilei intregi N gradul polinomului
 6       // Obtinerea gradului polinomului de la utilizator, conversie String-int
 7
 8       // Declararea si crearea tabloului coeficientilor, C
 9       // Obtinerea de la utilizator a coeficientilor Ci, unde i=0,N
10
11       // Afisarea polinomului P(X)
12       // - mai intai termenul liber Co
13       // - apoi termenii Ci*X^i, unde i=1,N
14
15       // Declararea variabilei intregi X
16       // Obtinerea valorii necunoscutei de la utilizator, conversie String-int
17
18       // Afisarea valorii necunoscutei (X)
19
20       // Declararea si initializarea variabilei intregi numita polinom,
21       // care contine valoarea polinomului, P(X)
22
23       // Calculul polinomului P(X) = suma(Ci * X^i), unde i=0,N
24          // - calculul valorii X^i, unde i=1,N
25          // - actualizarea valorii polinomului
26
27       // Afisarea valorii polinomului P(X)
28     }
29   }
```

In laborator:

1. **Creati fisierul** sursa **`Polinom1.java`** cu **continutul de mai sus (adica inclusiv comentariile!!!)** in directorul curent.

2. **Adaugati sub fiecare grup de comentarii codul necesar (*Acest cod este parte a temei de casa!*)**:

- copiind codul aflat sub comentariile scrise neintensificat din exemplul anterior (**`Polinom0`**),
- inspirandu-va din exemplele anterioare (**`SalutUtilizator0`** pentru interactivitatea bazata pe ferestre de dialog), si
- **folosind indicatiile care urmeaza**.

3. **Compilati si executati programul** in consola, in directorul curent. **Corectati eventualele erori.**

Indicatii:

- combinand doua sintaxe deja cunoscute se poate scrie:

```
int N = Integer.parseInt(JOptionPane.showInputDialog(
                                "Introduceti gradul polinomului"));
```

- **pentru a popula (atribui valori initiale) un tablou de intregi** se poate folosi o instructiune **`for`**.

1.4.3. Varianta cu delegare functionala

Pornind de la programul **`Polinom1`** sa se scrie codul unei clase Java numita **`Polinom2`**, a carei structura interna contine:

- **o metoda** (declarata `public static`) numita **`obtineGrad()`**, care
  - obtine de la utilizator valoarea gradului polinomului, si
  - o returneaza ca intreg de tip **`int`**,
- **o metoda** (declarata `public static`) numita **`stabilesteCoeficienti()`**, care
  - primeste un parametru intreg de tip **`int`**, numit **`N`**, reprezentand gradului polinomului,
  - creeaza un nou tablou al coeficientilor (cu **`N+1`** elemente),
  - obtine de la utilizator valori pentru coeficientii polinomului si
  - populeaza cu ei tabloul nou creat, apoi
  - returneaza taboul de tip **`int[]`** creat,
- **o metoda** (declarata `public static`) numita **`obtineNecunoscuta()`**, care
  - obtine de la utilizator valoarea necunoscutei, si
  - o returneaza ca intreg de tip **`int`**,
- **o metoda** (declarata `public static`) numita **`afisarePolinom()`**, care
  - primeste un parametru intreg de tip **`int`**, numit **`N`**, reprezentand gradului polinomului, si un parametru de tip **`int[]`**, numit **`C`**, reprezentand coeficientii polinomului, si
  - afiseaza polinomul corespunzator valorilor primite,
- **o metoda** (declarata `public static`) numita **`valoarePolinom()`**, care
  - primeste un parametru intreg de tip **`int`**, numit **`N`**, reprezentand gradului polinomului, un parametru de tip **`int[]`**, numit **`C`**, reprezentand coeficientii polinomului, si un parametru intreg de tip **`int`**, numit **`X`**, reprezentand necunoscuta,
  - calculeaza valoarea polinomului corespunzatoare valorilor primite si
  - o returneaza ca intreg de tip **`int`**,
- o **metoda principala, de test**, care:
  - deleaga catre metoda **`obtineGrad()`** obtinerea valorii gradului polinomului,
  - deleaga catre metoda **`stabilesteCoeficienti()`** stabilirea valorilor coeficientilor,
  - deleaga catre metoda **`afisarePolinom()`** afisarea polinomului,
  - deleaga catre metoda **`obtineNecunoscuta()`** obtinerea valorii necunoscutei,
  - afiseaza valoarea necunoscutei,
  - deleaga catre metoda **`valoarePolinom()`** calculul valorii polinomului,
  - afiseaza valoarea polinomului.

In pseudocodul din codul **care urmeaza comentariile scrise intensificat** (**bold**) reprezinta **elementele de noutate fata de programul anterior**.

```
 1   import javax.swing.JOptionPane;
 2   public class Polinom2 {
 3
 4     // Metoda care obtine de la utilizator gradul polinomului
 5     public static int obtineGrad() {
 6       // Obtinerea de la utilizator a gradului polinomului
 7       int N = Integer.parseInt(JOptionPane.showInputDialog(
 8                                          "Introduceti gradul polinomului"));
 9       // Returnarea valorii gradului polinomului
10       return N;
11     }
12
13     // Metoda care obtine de la utilizator coeficientii polinomului
14     public static int[] stabilesteCoeficienti(int N) {
15       // Declararea si crearea tabloului coeficientilor, numit C
16       // Obtinerea de la utilizator a coeficientilor Ci, unde i=0,N
17
18       // Returnarea tabloului coeficientilor
19     }
20
21     // Metoda care afiseaza polinomul P(X)
22     public static void afisarePolinom(int N, int[] C) {
23       // Afisarea polinomului P(X)
24       // - mai intai termenul liber Co
25       // - apoi termenii Ci*X^i, unde i=1,N
26     }
27
28     // Metoda care obtine de la utilizator valoarea necunoscutei
29     public static int obtineNecunoscuta() {
30       // Obtinerea de la utilizator a valorii necunoscutei
31
32       // Returnarea valorii necunoscutei
33     }
34
35     // Metoda care calculeaza valoarea polinomului pt o valoare a necunoscutei
36     public static int valoarePolinom(int N, int[] C, int X) {
37       // Declararea si initializarea variabilei intregi numita P
38
39       // Calculul polinomului P(X) = suma(Ci * X^i), unde i=0,N
40          // - calculul valorii X^i, unde i=1,N
41          // - actualizarea valorii polinomului
42
43       // Returnarea valorii polinomului
44     }
45
46     // Metoda principala. Utilizata pentru testarea celorlalte metode.
47     public static void main(String[] args) {
48       // Apelul metodei care obtine de la utilizator gradul polinomului
49       int N = obtineGrad();
50
51       // Apelul metodei care obtine de la utilizator coeficientii polinomului
52
53       // Apelul metodei care afiseaza polinomul
54
55       // Apelul metodei care obtine o valoare a necunoscutei
56       // Afisarea valorii necunoscutei
57
58       // Apelul metodei care calculeaza polinomul pentru necunoscuta data
59       // Afisarea valorii polinomului
60       System.exit(0); // Inchiderea interfetei grafice
61     }
62   }
```

In laborator, in limita timpului disponibil, sau acasa:

1. **Creati fisierul** sursa **`Polinom2.java`** cu **continutul de mai sus (adica inclusiv comentariile!!!)**.
2. **Adaugati sub fiecare grup de comentarii codul necesar (*Acest cod este parte a temei de casa!*)**:

- copiind codul aflat sub comentariile scrise neintensificat din exemplul anterior (**`Polinom1`**), si
- inspirandu-va din exemplul dat (metoda **`obtineGrad()`**).

3. **Compilati si executati programul** in consola, in directorul curent. **Corectati eventualele erori.**

Tema suplimentara de dificultate sporita:

Studentii care doresc pot incerca sa scrie codul necesar executiei repetate a programului, astfel incat sa poata calcula valoarea polinomului pentru mai multe valori succesive ale necunoscutei.

1.5. Instalarea si configurarea platformei de dezvoltare Java SE vers. 5

1.5.1. Instalarea platformei de dezvoltare Java SE (*Second Edition*)

Pentru obtinerea platformei de dezvoltare Java JDK 5.0 (incluzand kitul de dezvoltare Java - JDK) trebuie accesata pagina site-ului Sun **http://java.sun.com/javase/downloads/index_jdk5.jsp**

- cautata **JDK 5.0 Update 12** care contine Java SE Development Kit (JDK)
- click pe **>>Download**
- selectat **OAccept** License Agreement
- click pe v Windows Offline Installation, Multi-language (jdk-1_5_0_12-windows-i586-p.exe)

Instalarea kitului de dezvoltare Java sub Windows este in general simpla (vezi si http://java.sun.com/j2se/1.5.0/install.html pentru Linux si Windows). Dupa lansarea in executie a kitului **se pot accepta toate optiunile implicite pe care le ofera kitul**. Directorul de baza al instalarii va fi **`C:\Program Files\Java\jdk1.5.0_12`**.

1.5.2. Configurarea variabilelor de mediu *PATH* si *CLASSPATH*

Procedura incepe cu selectarea ***Start*** -> ***Control Panel*** -> ***System*** -> ***Advanced*** -> ***Environment Variables*** -> ***System Variables*** (**nu** *User Variables*).

1. Pentru a se putea executa programele care compun kitul de dezvoltare Java, trebuie adaugata la variabila de mediu **PATH calea catre** directorul **\bin** al kitului Java (in care se gasesc executabilele care formeaza kitul Java), folosind comanda **-> *Edit***:

- se va adauga calea **`C:\Program Files\Java\jdk1.5.0_12\bin`**.

2. Pentru a fi vizibile clasele de biblioteca standard Java, precum si clasele din directorul curent folosit pentru dezvoltarea programelor, trebuie creata o noua variabila de mediu **CLASSPATH**, folosind comanda **-> *New***:

- acestei variabile i se da valoarea **`C:\Program Files\Java\jdk1.5.0_12\lib;.;`** (unde **\lib** este directorul bibliotecii, iar **punctul** este **calea catre directorul curent, necesara executiei programelor create de noi**).

1.6. Teme pentru acasa (inclusive teme suplimentare)

Tema de casa pentru data viitoare:

I. Interpretarea rezultatelor obtinute prin modificarea programului `Salut.java` (pag 4).
II. Codurile sursa ale programelor `Polinom0`, `Polinom1` si `Polinom2` completate (pag 11-16).

Tema suplimentara pentru data viitoare (pentru bonus):

I. Codul pentru afisarea polinomului in format dat (pag 12).
II. Codul pentru executia repetata a calculului polnomului (pag 16).

Anexa

1. Accesul la consola standard de intrare

Ca alternativa la utilizarea ferestrelor de dialog se poate utiliza **perechea de console standard**, tastatura (de intrare) si ecranul in mod text (de iesire).

Programul `SalutUtilizator2.java` foloseste varianta cu clasa **`Scanner`** din pachetul **`java.util`**. Varianta este **disponibila de la versiunea Java 5 (JDK 1.5.0).**

```
 1   import java.util.*; // Importul pachetului/bibliotecii de clase utilitare
 2
 3   public class SalutUtilizator2 {
 4     public static void main(String[] args) {
 5       String modSalut = "Bun venit in programarea orientata spre obiecte, ";
 6
 7       Scanner sc = new Scanner(System.in);
 8
 9       System.out.print("Introduceti-va numele: ");
10       String nume = sc.nextLine();
11
12       System.out.println(modSalut + nume + "!");
13     }
14   }
```

Programul `SalutUtilizator2.java` foloseste varianta cu clasele **`BufferedReader`** si **`InputStreamReader`** din pachetul **`java.io`**. Varianta este cea **utilizata inca de la prima versiune Java**, dar este mai complicata decat cea bazata pe clasa **`Scanner`**.

```
 1   import java.io.*;   // Importul pachetului de clase de intrare-iesire (IO)
 2
 3   public class SalutUtilizator3 {
 4     public static void main(String[] args) throws IOException {
 5       String modSalut = "Bun venit in programarea Java, ";
 6
 7       BufferedReader b =new BufferedReader(new InputStreamReader(System.in));
 8
 9       System.out.print("Introduceti-va numele: ");
10       String nume = b.readLine();
11
12       System.out.println(modSalut + nume + "!");
13     }
14   }
```

In acest caz, citirea consolei standard de intrare se face prin intermediul unui **flux de intrare pentru caractere** (in Java, caracterele sunt codate UNICODE pe 2 octeti), **`BufferedReader`**, **conectat la fluxul standard de intrare** (**`System`**`.in`, care ofera datele de la tastatura sub forma de octeti simpli) prin intermediul unui flux de intrare care face conversia octet-caracter (**`InputStreamReader`**).

2. Delegarea functionala

Metodele sunt **cazul cel mai simplu si mai des intalnit de delegare a functionalitatii**. Separarea codului in functii (metode) **serveste gestiunii mai simple** a codului, **modularizarii codului la nivel de secventa** de instructiuni, **reutilizarii codului la nivel de functie**.

In general **functia principala `main()` deleaga anumite sarcini** catre alte functii, **acestea la randul lor** catre altele, s.a.m.d.