Seminar 3. Introducere in programare Java

Marian Necula

25.10.2024

Introducere

Timp de lucru estimat: 5 minute

Acest document reprezintă Seminarul 3 pentru cursul de introducere în programare Java. El conține o parte teoretică despre structurile de programare `if-else`, `for` și `while`, teorie despre proiectarea problemelor în pseudocod, precum și o parte de aplicații practice cu exemple și probleme propuse, inclusiv implementări în pseudocod și Java.

1 Partea Teoretică

Timp de lucru estimat: 60 minute

1.1 Proiectarea problemelor în pseudocod

Pseudocodul este un instrument esențial în proiectarea algoritmilor și programelor. El permite descrierea logicii unui algoritm într-un mod simplu și clar, fără a fi nevoie de sintaxa specifică unui limbaj de programare.

1.1.1 Avantajele utilizării pseudocodului

- **Claritate**: Ajută la înțelegerea logicii unui algoritm fără detalii de implementare.
- **Comunicare**: Facilitează comunicarea între programatori și non-programatori.

- **Planificare**: Permite identificarea problemelor și optimizarea logicii înainte de codificare.

1.1.2 Reguli generale pentru scrierea pseudocodului

- Folosiți un limbaj simplu, aproape de limbajul natural.
- Utilizați indentarea pentru a evidenția structurile de control.
- Evitați detaliile specifice limbajelor de programare (tipuri de date, sintaxă).
- Fiți consecvenți în utilizarea termenilor și a structurii.

1.2 Structuri de programare în Java

1.2.1 Structura condițională `if-else`

Structura `if-else` permite executarea unor blocuri de cod în funcție de evaluarea unei condiții.

Sintaxa în Java:

```
if (conditie) {
        // bloc de cod daca conditia este adevarata
} else {
        // bloc de cod daca conditia este falsa
}
```

Exemplu în pseudocod:

```
Dacă (conditie) atunci
// instructiuni dacă conditia este adevarata
Altfel
// instructiuni dacă conditia este falsa
Sfarsit daca
```

1.2.2 Structura repetitivă `for`

Structura `for` este utilizată pentru a repeta un bloc de cod de un număr cunoscut de ori.

Sintaxa în Java:

```
for (initializare; conditie; incrementare) {
        // bloc de cod de executat
}
```

Exemplu în pseudocod:

```
Pentru (initializare; conditie; incrementare) executa
// instrucțiuni
Sfârșit pentru
```

1.2.3 Structura repetitivă `while`

Structura `while` repetă un bloc de cod atâta timp cât o condiție este adevărată.

Sintaxa în Java:

```
while (conditie) {
        // bloc de cod de executat
}
```

Exemplu în pseudocod:

```
Cât timp (condiție) execută
// instrucțiuni
Sfârșit cât timp
```

1.3 Exemple

1.3.1 Exemplul 1: Determinarea maximului dintre două numere

Pseudocod:

```
Algoritm MaximDintreDouaNumere
Citește număr1
Citește număr2
Dacă număr1 > număr2 atunci
Afișează "Numărul maxim este", număr1
Altfel
Afișează "Numărul maxim este", număr2
Sfârșit dacă
Sfârșit algoritm
```

Implementare în Java:

```java
import java.util.Scanner;

public class MaximDintreDouaNumere {
        public static void main(String[] args) {
                Scanner scanner = new Scanner(System.in);
                System.out.print("Introduceti primul numar: ");
                double numar1 = scanner.nextDouble();
                System.out.print("Introduceti al doilea numar: ");
                double numar2 = scanner.nextDouble();

                if (numar1 > numar2) {
                        System.out.println("Numarul maxim este " + numar1);
                } else if (numar2 > numar1) {
                        System.out.println("Numarul maxim este " + numar2);
                } else {
                        System.out.println("Numerele sunt egale.");
                }
        }
}
```

1.3.2 Exemplul 2: Afișarea numerelor pare de la 1 la N

Pseudocod:

```
Algoritm AfisareNumerePare
Citește N
Pentru i de la 1 la N execută
Dacă i % 2 == 0 atunci
Afișează i
Sfârșit dacă
Sfârșit pentru
Sfârșit algoritm
```

Implementare în Java:

```java
import java.util.Scanner;

public class AfisareNumerePare {
        public static void main(String[] args) {
                Scanner scanner = new Scanner(System.in);
                System.out.print("Introduceti valoarea lui N: ");
                int N = scanner.nextInt();

                for (int i = 1; i <= N; i++) {
                        if (i % 2 == 0) {
                                System.out.println(i);
                        }
                }
        }
}
```

1.3.3 Exemplul 3: Calcularea sumei cifrelor unui număr

Pseudocod:

```
Algoritm SumaCifrelor
Citește număr
suma <- 0
Cât timp număr > 0 execută
cifra <- număr % 10
suma <- suma + cifra
număr <- număr / 10
Sfârșit cât timp
Afișează "Suma cifrelor este", suma
Sfârșit algoritm
```

Implementare în Java:

```java
import java.util.Scanner;

public class SumaCifrelor {
        public static void main(String[] args) {
                Scanner scanner = new Scanner(System.in);
                System.out.print("Introduceti un numar intreg pozitiv: ");
                int numar = scanner.nextInt();
                int suma = 0;

                while (numar > 0) {
                        int cifra = numar % 10;
                        suma += cifra;
                        numar = numar / 10;
                }

                System.out.println("Suma cifrelor este " + suma);
        }
}
```

2 Partea Practică

Timp de lucru estimat: 60 minute

2.1 Probleme propuse

1. Determinarea dacă un număr este pozitiv, negativ sau zero

Scrieți un program care citește un număr și determină dacă acesta este pozitiv, negativ sau zero.

Pseudocod:

```
Algoritm DeterminareSemn
Citește număr
Dacă număr > 0 atunci
Afișează "Numărul este pozitiv"
```

```
Altfel dacă număr < 0 atunci
Afișează "Numărul este negativ"
Altfel
Afișează "Numărul este zero"
Sfârșit dacă
Sfârșit algoritm
```

2. Calcularea factorialului unui număr

Scrieți un program care calculează factorialul unui număr N introdus de utilizator.

Pseudocod:

```
Algoritm CalculFactorial
Citește N
factorial <- 1
Pentru i de la 1 la N execută
factorial <- factorial * i
Sfârșit pentru
Afișează "Factorialul este", factorial
Sfârșit algoritm
```

3. Verificarea dacă un număr este prim

Scrieți un program care verifică dacă un număr N este prim.

Pseudocod:

```
Algoritm VerificareNumarPrim
Citește N
estePrim <- adevărat
Dacă N <= 1 atunci
estePrim <- fals
Altfel
Pentru i de la 2 la N - 1 execută
Dacă N % i == 0 atunci
estePrim <- fals
Ieși din ciclu
Sfârșit dacă
Sfârșit pentru
Sfârșit dacă
```

```
Dacă estePrim atunci
Afișează "Numărul este prim"
Altfel
Afișează "Numărul nu este prim"
Sfârșit dacă
Sfârșit algoritm
```

4. Afișarea tabelului de înmulțire pentru un număr

Scrieți un program care afișează tabelul de înmulțire pentru un număr N introdus de utilizator.

Pseudocod:

```
Algoritm TabelInmultire
Citește N
Pentru i de la 1 la 10 execută
produs <- N * i
Afișează N, "x", i, "=", produs
Sfârșit pentru
Sfârșit algoritm
```

5. Calcularea sumei numerelor de la 1 la N

Scrieți un program care calculează suma numerelor de la 1 la N.

Pseudocod:

```
Algoritm SumaNumerelor
Citește N
suma <- 0
Pentru i de la 1 la N execută
suma <- suma + i
Sfârșit pentru
Afișează "Suma este", suma
Sfârșit algoritm
```

6. Generarea numerelor Fibonacci până la N

Scrieți un program care generează și afișează numerele Fibonacci până la N.

Pseudocod:

```
Algoritm NumarFibonacci
Citește N
a <- 0
b <- 1
Cât timp a <= N execută
Afișează a
c <- a + b
a <- b
b <- c
Sfârșit cât timp
Sfârșit algoritm
```

7. Inversarea unui număr

Scrieți un program care citește un număr întreg pozitiv și afișează inversul acestuia.

Pseudocod:

```
Algoritm InversareNumar
Citește număr
invers <- 0
Cât timp număr > 0 execută
cifra <- număr % 10
invers <- invers * 10 + cifra
număr <- număr / 10
Sfârșit cât timp
Afișează "Numărul inversat este", invers
Sfârșit algoritm
```

8. Verificarea dacă un număr este palindrom

Scrieți un program care verifică dacă un număr întreg pozitiv este palindrom (numărul este egal cu inversul său).

Pseudocod:

```
Algoritm VerificarePalindrom
Citește număr
numărOriginal <- număr
invers <- 0
Cât timp număr > 0 execută
```

```
cifra <- număr % 10
invers <- invers * 10 + cifra
număr <- număr / 10
Sfârșit cât timp
Dacă numărOriginal == invers atunci
Afișează "Numărul este palindrom"
Altfel
Afișează "Numărul nu este palindrom"
Sfârșit dacă
Sfârșit algoritm
```

9. Calcularea puterii unui număr

Scrieți un program care calculează valoarea lui x^y, unde x și y sunt introduse de utilizator.

Pseudocod:

```
Algoritm CalculPutere
Citește x
Citește y
rezultat <- 1
Pentru i de la 1 la y execută
rezultat <- rezultat * x
Sfârșit pentru
Afișează x, " la puterea ", y, " este ", rezultat
Sfârșit algoritm
```

10. Ghicirea unui număr

Scrieți un program care generează un număr aleator între 1 și 100 și permite utilizatorului să încerce să ghicească numărul. Programul oferă indicii ("Prea mic", "Prea mare") și se oprește când utilizatorul ghicește numărul.

Pseudocod:

```
Algoritm JocGhicireNumar
Generați numărSecret între 1 și 100
ghicit <- fals
Cât timp nu ghicit execută
Afișează "Introduceți numărul ghicit:"
Citește numărIntrodus
```

```
Dacă numărIntrodus == numărSecret atunci
Afișează "Felicitări! Ați ghicit numărul."
ghicit <- adevărat
Altfel dacă numărIntrodus < numărSecret atunci
Afișează "Prea mic. Încercați din nou."
Altfel
Afișează "Prea mare. Încercați din nou."
Sfârșit dacă
Sfârșit cât timp
Sfârșit algoritm
```

2.2 Instrucțiuni pentru rezolvare

Pentru fiecare problemă:

- Analizați enunțul și identificați cerințele.
- Studiați pseudocodul furnizat pentru a înțelege algoritmul.
- Implementați algoritmul în Java, respectând structura logică din pseudocod.
- Testați programul cu diferite valori de intrare pentru a verifica corectitudinea.