Numele studentului şi grupa:

Obiectivele laboratorului

După parcurgerea acestui laborator, studentul trebuie să înțeleagă semnificația conceptelor de *clasă* şi *metodă*, astfel încât să fie capabil sa dezvolte aplicații obiect orientate simple.

Elemente teoretice necesare

Înainte de a începe rezolvarea activităților de laborator propuse, trebuie să răspundeți la următoarele întrebări:

- Ce este o clasă?

- Ce este un constructor al unei clase şi care este rolul său?

- Ce sunt metodele şi cum se folosesc parametrii acestora?

- Ce sunt pachetele?

Fiecare dintre aceste întrebări de control necesită, pe lângă stabilirea şi notarea pe această foaie de laborator a definiției propriu zise, înțelegerea utilității practice a acestor concepte fundamentale ale programării orientate pe obiecte.

Activități efective

Fiecare dintre întrebările care vor urma trebuie să fie procesată, inițial, fără ajutorul calculatorului. Apoi, pentru a verifica corectitudinea răspunsurilor, codul va fi compilat şi rulat. Corespund răspunsurile? În acest sens, dacă există diferențe, se vor scrie ambele variante, atât prima (cea greşită) cât şi cea generată de calculator. Încercați să înțelegeți care este sursa greşelii şi adăugați pe foaia de răspuns explicația.

Fişierul Money.java

```
package money;
public class Money
{
  // Data fields
  public final char SYMBOL = '$';
  private long dollars;
  private long cents;

  public  Money()
  // Constructor:  Sets dollars and cents to 0
  {
    dollars = 0;
    cents   = 0;
  }
  public Money(long initDollars, long initCents)
  // Constructor: Sets dollars and cents to parameter values
  {
    dollars = initDollars;
    cents   = initCents;
  }

  public void  initialize(long newDollars, long newCents)
  // Initializes dollars and cents
  {
    dollars = newDollars;
    cents = newCents;
  }

  public long  dollarsAre()
  // Returns dollars
  {
    return dollars;
  }

  public long  centsAre()
  // Returns cents
  {
    return cents;
  }

  public void print()
  // Prints dollars and cents on System.out
  {
    System.out.print("" + SYMBOL + dollars + '.' + cents);
  }

  public Money  add(Money  value)
  // Returns sum of object plus value.
  {
    Money  result = new Money();
    result.cents = cents + value.cents;
    result.dollars = dollars + value.dollars;
    return result;
  }
}
```

Fişierul UseMoney.java

```
import money.*;

public class UseMoney
{
  public static void main(String[] args)
  {
    Money  money1;
    Money  money2;
    Money  money3;
    money1 = new Money(10, 59);
    money2 = new Money(20, 70);
```

```
        money3 = money1.add(money2);
        System.out.println("" + Money.SYMBOL + money3.dollarsAre() + '.'
          + money3.centsAre());
      }
    }
```

- Ce afişează clasa *UseMoney*?
- De ce are metoda *add* un singur parametru?

Definiţia clasei *Money* este încapsulată în pachetul *money*. Clasa *UseMoney* importă acest pachet şi manipulează obiecte (instanţe) ale clasei *Money*. În acest context, precizaţi care sunt câmpurile de date ale *clasei*, ale *instanţei*, precum şi câmpurile de date *locale* (câmpurile de date ale clasei se referă la acele atribute ale clasei care **nu** intervin nemijlocit în funcţionarea unei instanţe a clasei, cele ale instanţei la acele atribute ale clasei care intervin nemijlocit in funcţionarea oricărei instanţe, iar datele locale sunt reprezentate de oricare alte câmpuri de date auxiliare, pe lângă cele legate strict de clasa în discuţie.

Câmpuri de date ale clasei:
Câmpuri de date ale instanţei:
Câmpuri de date locale:

Timp de lucru (documentare plus parcurgerea activităţilor propuse) : o săptămână. Studentul va tipări această foaie de laborator şi o va preda, completată, la termenul stabilit.

Notă : se poate folosi, la alegere, unul dintre următoarele medii de programare: Eclipse (varianta preferată) sau NetBeans.