Numele studentului şi grupa:

Obiectivele laboratorului

Introducere în moştenire, polimorfism şi sintaxa Java asociată.

Elemente teoretice necesare

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

- Ce este moştenirea?

- Ce este polimorfismul?

- Care este domeniul de vizibilitate al unei variabile în Java?

- Ce este aceea supraîncărcare (overloading) şi suprascriere (overriding) a metodelor?

- Când putem spune că ascundem (hiding) un atribut într-un lanț de moştenire (de derivare)?

Activități efective

Luând în considerare clasa Money folosită în primele laboratoare şi pachetul money care o conține, să se studieze următorul cod Java:

```
package extMoney;
import money.*;


public class ExtMoney extends Money
{
  private String kindOfMoney;

  public ExtMoney()
```

```
  // Constructor: default kind is "real"
  {
    super();
    kindOfMoney = "real";
  }
  public ExtMoney
    (long newDollars, long newCents, String newKind)
  // Constructor: kind set to newCurrency
  {
    super(newDollars, newCents);
    kindOfMoney = newKind;
  }
  public void initialize
    (long newDollars, long newCents, String newKind)
  {
    this.initialize(newDollars, newCents);
    kindOfMoney = newKind;
  }

  public String kindIs()
  {
    return kindOfMoney;
  }

  public void print()
  {
    super.print();
    System.out.println(" is " + kindOfMoney);
  }
}
```

- Care sunt metodele supraîncărcate (overloaded)?
- Care sunt metodele suprascrise (overridden)?
- Care sunt atributele ascunse?
- Observați cuvântul cheie **extends** care specifică faptul că ExtMoney este o clasă care extinde Money, precum şi directiva **super()** care apelează constructorii sau metodele similare din clasa părinte. De asemenea, în programul principal de mai jos, observați apelul polimorfic al metodei print(), în sensul că inițial este apelată în contextul unui obiect de tip Money, iar apoi pentru un obiect de tip ExtMoney.

Se consideră acum următorul program principal:

```
import money.Money;
import extMoney.ExtMoney;
public class UseMoney
// A driver class using Money and ExtMoney
{
  public static void main(String[] args)
  {
    Money money1;
    Money money2;
    ExtMoney extMoney1;
    ExtMoney extMoney2;

    System.out.println("Initialized by default constructors");
    money1 = new Money();
    extMoney1 = new ExtMoney();
    System.out.println("variables instantiated");
    money1.print();
    extMoney1.print();
```

```
    System.out.println("Initialized by other constructor");
    money2 = new Money(2000, 22);
    extMoney2 = new ExtMoney(3000, 88, "monoply");

     //illustration of the polymorphism
    money2.print();
    extMoney2.print();

    System.out.println("initialized at run time");
    money1.initialize(4000, 44);
    extMoney1.initialize(5000, 99, "play");
    money1.print();
    extMoney1.print();
  }
}
```

Ce se afişează?

- Scrieți o miniaplicație prin care să vă verificați însuşirea sintaxei minimale pentru scrierea de cod POO în Java. Aceasta va conține trei clase: o clasă părinte, o clasă derivată, care o moşteneşte (**extends**) şi o clasă principală, care conține metoda main() şi foloseşte celelalte două clase. De asemenea, va fi inclus şi (cel puțin) un apel polimorfic, ca în exemplul precedent. Studentul este liber să creeze şi să implementeze scenariul acestei aplicații-test aşa cum doreşte.

Ataşați la această foaie de laborator fragmentele de cod relevante (tipărite la imprimantă sau scrise de mână).

Timp de lucru: o săptămână.

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