Laborator 5.

Avram Remus, Grupa 7801

1. Moştenirea este o modalitate performantă de reutilizare a codului, dar nu este întotdeauna cel

mai bun instrument pentru îndeplinirea acestui obiectiv.

2. Polimorfism este posibilitatea ca un apel de funcţie (metodă , operaţie) să genereze răspunsuri diferite în funcţie de contextul în care a fost formulat.

3. Domeniul de vizibilitate al unei variabile este blocul de cod in care variabila este accesibila. Locul in care este declarata o variabila determina domeniul ei de vizibilitate. O variabila poate fi:

- o variabila membru a unui obiect

- un parametru al unei metode

- o variabila locala (declarata in corpul unui bloc de instructiuni)

- un parametru al unui handler de exceptii.

4. Supraincarcare si suprascrierea metodelor este posibilitatea de a scrie functii care au acelasi nume, returneaza acelasi tip de data, dar se pot deosebi prin tipul si numarul parametrilor functiei.

**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**.initializare(newDollars, newCents);

kindOfMoney = newKind;

}

**public** String kindIs()

{

**return** kindOfMoney;

}

**public** **void** print()

{

**super**.print();

System.*out*.println(" is " + kindOfMoney);

}

}

5. Metodele supraincarcate sunt: kindIs(){...},print(){...};

6. Metodele suprascrise sunt: ExtMoney (long newDollars, long newCents, String newKind){...}.

7. Atributele ascunse sunt kindOfMoney;

**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.initializare(4000, 44);

extMoney1.initialize(5000, 99, "play");

money1.print();

extMoney1.print();

}

}

8. Ce se afiseaza?

Initialized by default constructors

variables instantiated

$0.0$0.0 is real

Initialized by other constructor

$2000.22$3000.88 is monoply

initialized at run time

$4000.44$5000.99 is play