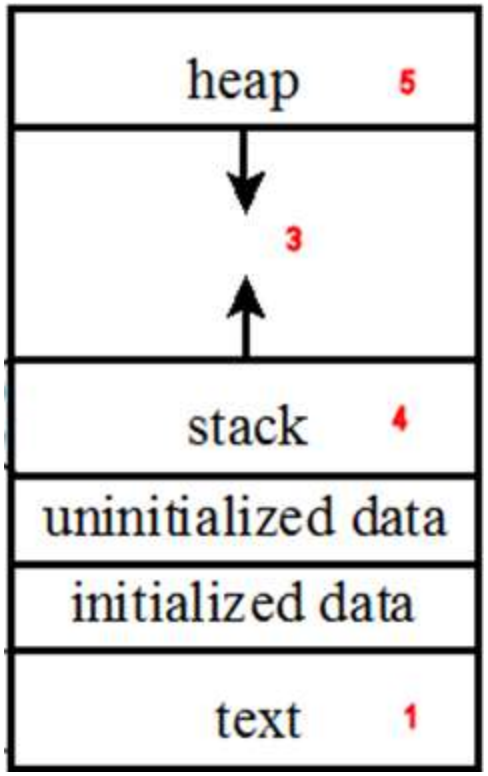
Optimizarea programelor din perspectiva obiectuala

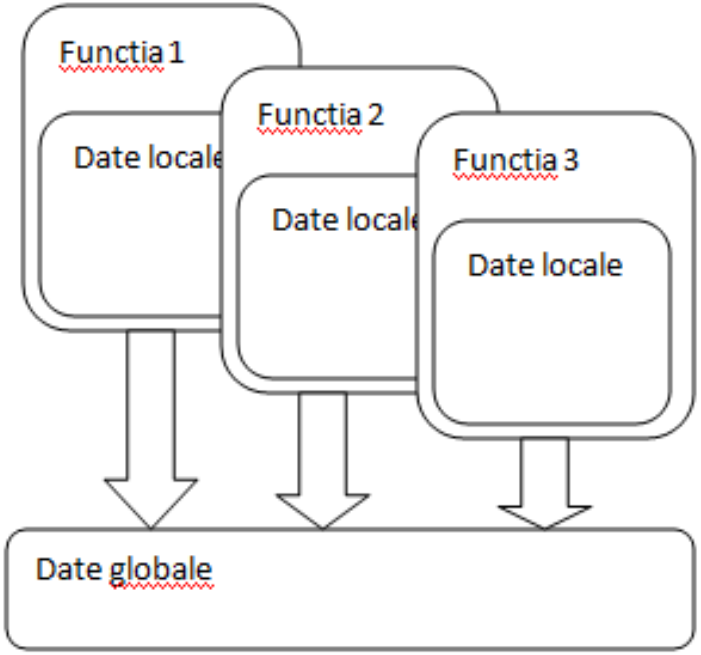
Radu Manea

Organizarea memoriei

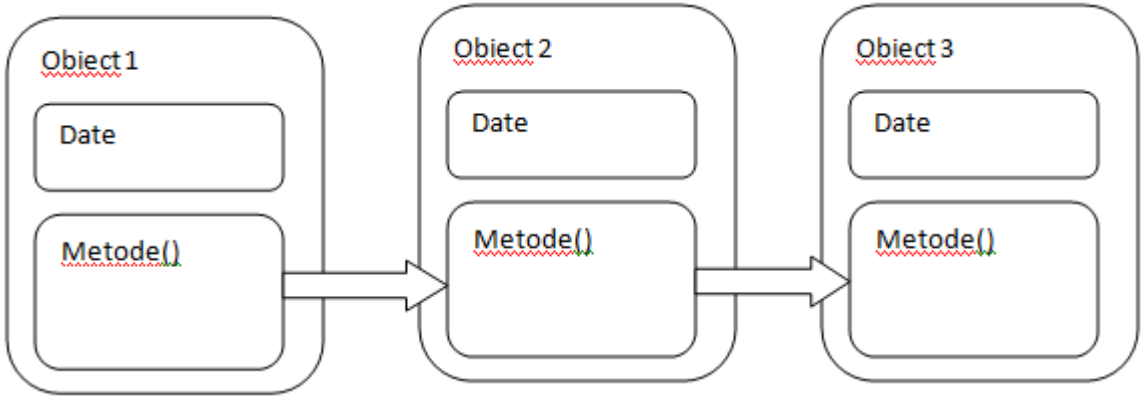
1. Segmentul destinat codului (CS)
2. Segmentul destinat variabilelor globale
3. Zona libera (Free store)
4. Segmentul alocat functiilor, parametrilor si variabilelor locale (DS)
5. Heap alocat



Programare functionala

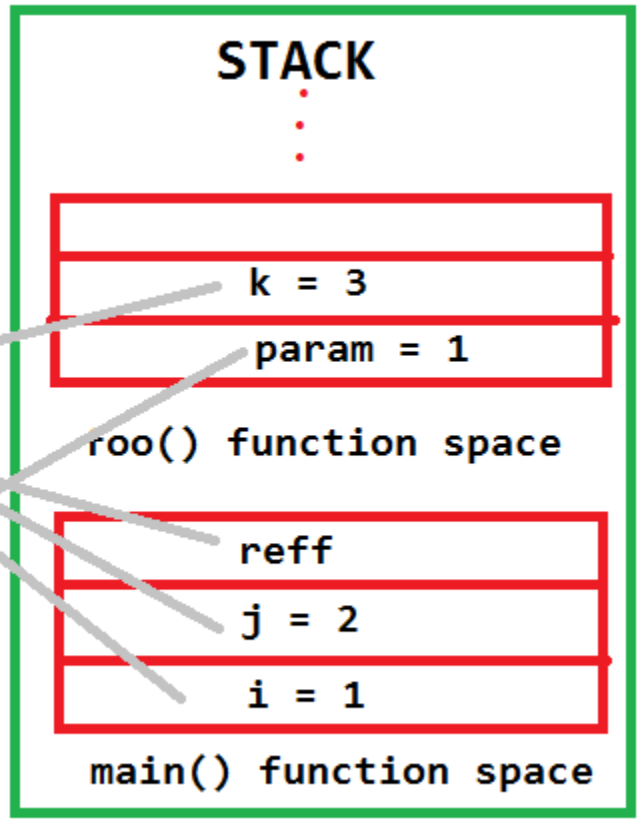


Programare orientata obiect



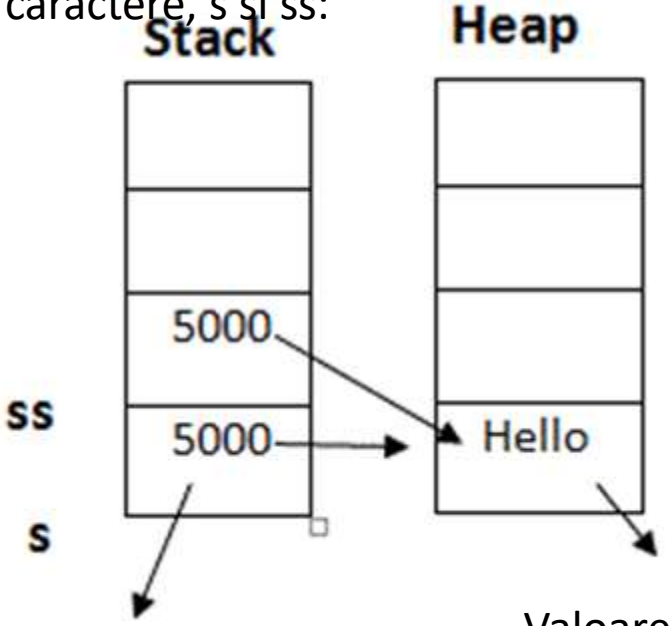
Modelul memoriei pentru limbaje OO

```
public class Stack_Test {

 public static void main(String[] args) {

        int i=1;
        int j=2;

        Stack_Test reff = new Stack_Test();
        reff.foo(i);
    }

     void foo(int param) {
        int k = 3;
        System.out.println(param);
    }

}
```



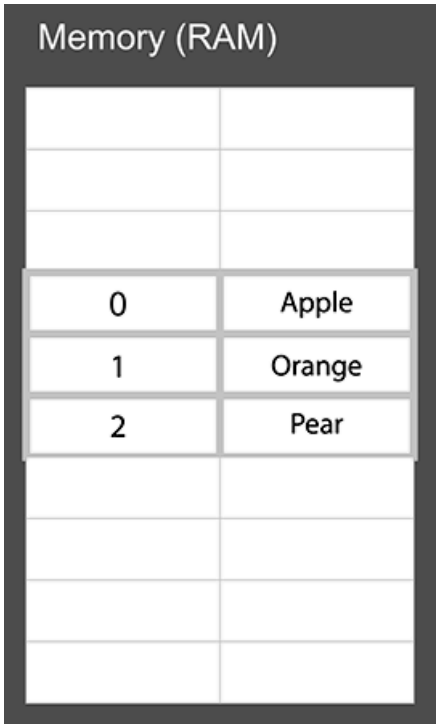
Interactiunea dintre Stack si Heap

Declaram doua variable de tip sir de caractere, s si ss:
String s = “Hello”;
String ss = s;



Vectori vs Obiecte

Organizarea memoriei in cazul vectorilor



Accesul la date se face direct, de ex:
Fructe[0] -> ‘Apple’;
Fructe[1] -> ‘Orange’;
Fructe[2] -> ‘Pear’;

Organizarea memoriei in cazul obiectelor

Accesul la date se face prin referinta intoarsa de o functie hash care permite maparea spatiului de adrese in mod aleator:

Persoana.name -> Vivek
Persoana.age -> 13
Persoana.class -> 8

Solutia: tipul Record

Înregistrările sunt tipuri de date imutabile care necesită doar tipul și numele câmpurilor:

Metodele equals, hashCode și toString, precum și câmpurile private, finale și constructorul public, sunt generate de compilatorul Java.

Mostenire vs Agregare

- Realizam mostenirea atunci cand detinem un obiect complex si dorim sa-l specializam ----->
- Agregarea se foloseste atunci cand dorim sa obtinem un obiect complex pornind de la obiecte mai simple <>-----
- O forma speciala de mostenire (prototipului) care vizeaza comportamentul obiectelor este implementarea ------|>

Agregarea si asocierea

- Doua clase sunt asociate atunci cand o metoda dintr-o clasa primeste un obiect apartinand altei clase ----------;
- In cazul agregarii vom avea o proprietate dintr-o clasa avand tipul altei clase.

Dependency Injection

- Cand primim un parametru de tip obiect gata instantiat spunem ca aplicam injectarea unui comportament
- Injectacarea dependentelor (DI) se poate face intre obiecte (manual) sau dintr-un framework (automat - IoC) ex. Struts, Spring
- Pentru a nu forta cunoasterea exacta a tipului obiectului de injectat in momentul compilarii se foloseste interfata lui.

Functional Programming

- In programarea functionala nu putem accesa obiecte in afara scopului functiei (pure)
- Practic vom putea modifica starea unui obiect doar apeland alta functie, aplicand compozitia, de ex:

  Functia_1(Functia_2())

  , in loc de:

  Functia_1(Obiect_2)
- Iteratiile sunt realizate folosind apelurile recursive
- Incapsularea datelor presupune folosirea variabilelor locale

Clojure vs Class

```
var a = (function () {
  let msg = 'Message in a
    bottle';
  function f() {
    console.log(msg);
  }
  return f;
})();
```

```
public class a {
private String msg =
    “Message in a bottle”;
public void f(){
System.out.println(msg);
}
```

Expresii lambda

```
public interface Inc {
int method(int a);
}
```

```
public class Main {

public static void main(String[] args) {
// TODO Auto-generated method stub
        int b=1;
        Inc a = (c) -> {return ++c;};

        System.out.println(a.method(b));
}
}
```

```
import java.util.function.Function;

public class Main {

public static void main(String[] args) {
      int b=1;
       Function<Integer,Integer> a = (c) -> {return ++c;};

       System.out.println(inc(b,a));
}

public static int inc(int arg, Function<Integer, Integer> fn) {
      return fn.apply(arg);
}
}
```

Concluzii

- P structurata separa datele de prelucrari
- P obiectuala separa datele intre obiecte
- P functionala trateaza functiile ca pe orice tip de data, o functie putand returna alta functie.
- P functionala se foloseste cand se cunosc toate tipurile de obiecte la momentul compilarii
- FP utila cand dorim sa transferam comportamentul asupra obiectelor de tip POJO (fara metode implementate), in acest sens exista expresii lambda.

Multumesc!