

Ingineria Sistemelor de Programare

Bazele OOP

mihai.hulea@aut.utcluj.ro

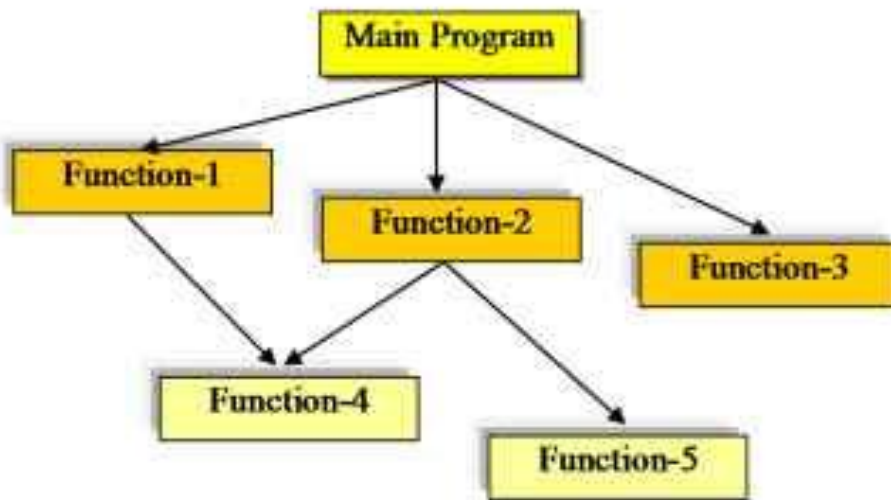
2017

Cuprins

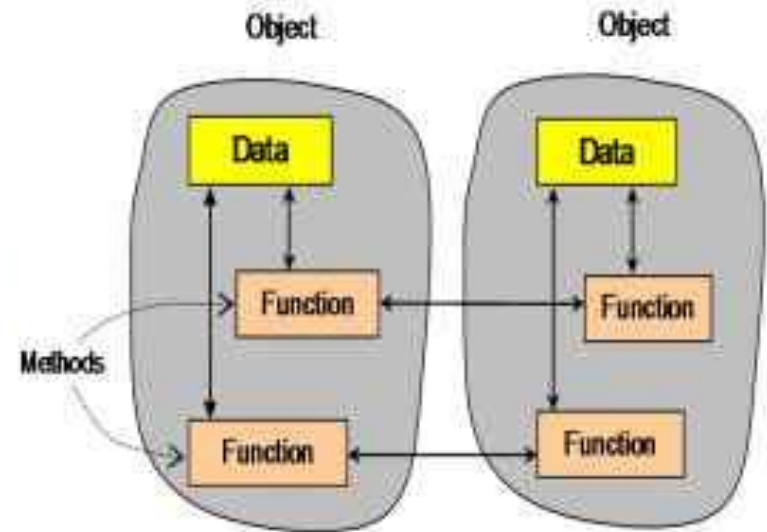
- Concepte OOP
- Clasa si obiect
- Variabile de instanta
- Metode de instanta
- Metode si variabile statice
- Constructori

Programarea Orientata pe Obiecte

Procedure-oriented Programming

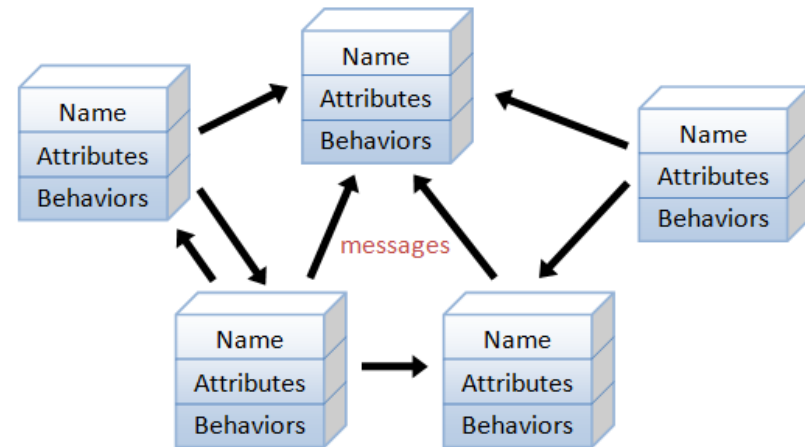
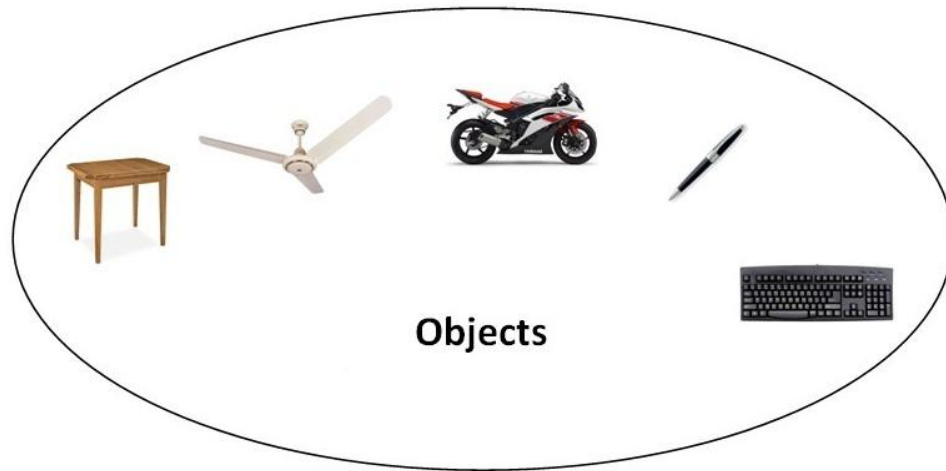


Object-oriented Programming



<http://www.c-sharpcorner.com/UploadFile/8a67c0/oops-vs-procedural-programming/>

Conceptul de obiect

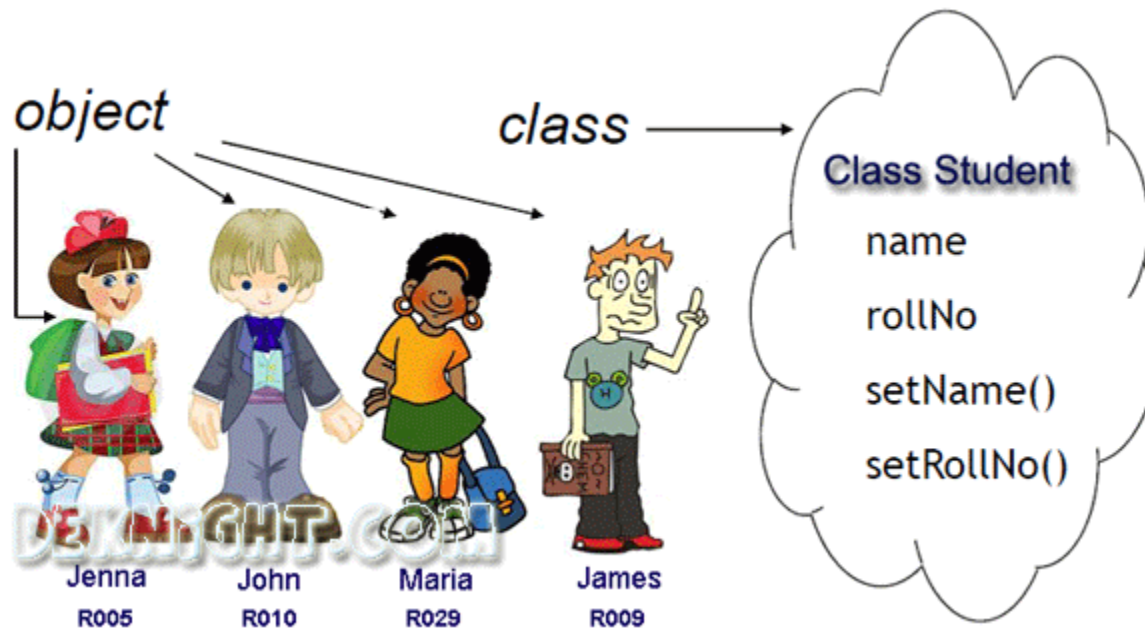


An object-oriented program consists of many well-encapsulated objects and interacting with each other by sending messages

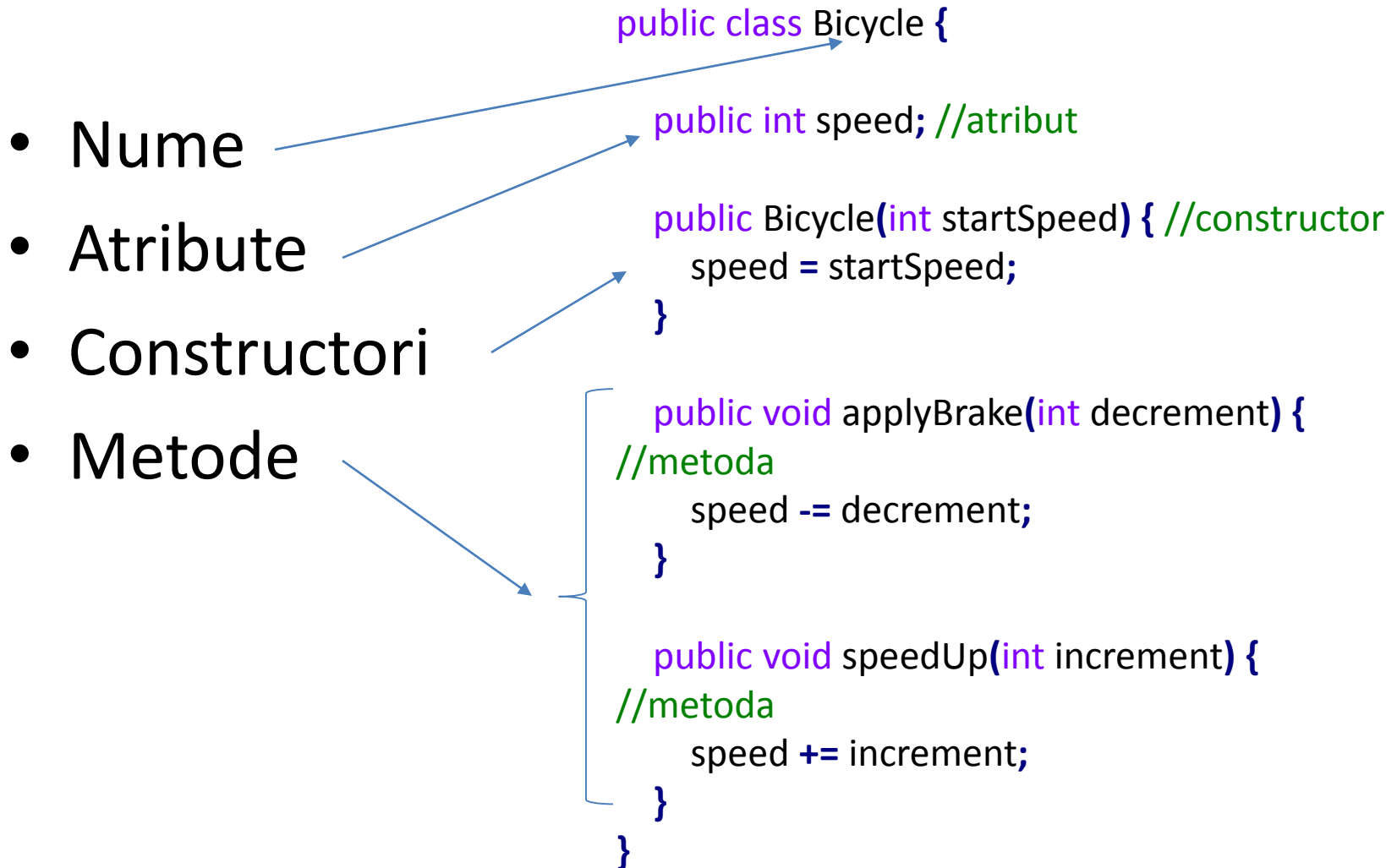
- Obiect (eng. Object) = O entitate caracterizata printr-o stare si un comportament.

Conceptul de clasa

- Clasa (eng. Class)= un sablon pe baza caruia se construiesc obiecte



Structura unei clase Java



Principiile programarii OOP

- Cele 4 principia ce stau la baza OOP
 - Abstractizarea
 - Incapsularea
 - Polimorfismul
 - Mostenirea

ATTRIBUTE DE INSTANTA

Motivatie

- Obiectele stocheaza date. Acestea se mai numesc: campuri, attribute sau variabile de instanta

- Sintaxa:

```
modifiers class MyClass{  
    modifiers SomeType filed1;  
}
```

- Atributele stocheaza starea obiectului

Exemplu

- Robot
 - Attributele
 - int x
 - int y
 - String name

Conventii Java

- Numele claselor incep cu litera mare
- Numele atributelor incep cu litera mica

Obiectele si referintele

- Dupa ce o clasa a fost definita pot fi declarate variabile (referinte) de acel tip
- Variabile de tip obiect au valoarea **null** daca nu refera catre nici un obiect
- Se utilizeaza operatorul **new** pentru a construi un obiect:

```
ClassName var1 = new ClassName();
```

Accesarea atributelor de instanta

- Se utilizeaza ‘.’ intre numele referintei (obiectului) si numele variabilei
`variableName.fieldName`
- Exemplu:

```
Robot r1 = new Robot();  
r1.x = 10;  
r1.y = 12;  
int d = r1.x - r1.y;
```
- Exceptie:
 - `variableName` poate lipsi daca suntem in interiorul unei metode de instanta din cadrul clasei

Aplicarea principiilor OOP

- Principiul incapsularii
 - Atributele de instanta ale unei clase sunt declarate **private**
 - Nu sunt accesibile din exterior de catre alte clase

METODE DE INSTANTA

Motivatie

Definitie: Functiile definite in cadrul unei clase se numesc metode (metode de instanta)

- Sintaxa:

```
modifiers class MyClass{  
    modifiers ReturnType myMethod(...){...}  
}
```

- Implementeaza comportamentul obiectului

Exemplu

- Robot
 - Attributele: x,y,speed
 - Metodele: move, printLocation, setLocation

Definirea metodei

```
methodModifiers returnType  
    methodName (parameter list) [throws exceptionList ]  
    {  
        declarations and statements  
        return something;  
    }
```

Exemplu:

```
public void move(){  
    x = x + speed;  
    y = y + speed;  
}
```

Supraincercarea metodelor

- Eng. Overloading
- Definirea in cadrul unei clase a mai multor metode cu numar sau tip diferit de argumente

Conceptul de getter si setter

- Metodele
 - get...() returneaza valoarea unui variabile membru private
 - set...(...) modifica valoarea unei variabile membru private
 - Eficienta ?

Accesarea metodelor interne si externe

- Accesarea metodelor din aceiasi clasa
- Accesarea metodelor din alte clase

Visibilitatea Metodelor si Variabilelor

- `private` – accesibile doar in interior
- `public` – accesibil de oriunde
- `protected` – accesibil in interior si in clasele derivate
- `package` – accesibil din pachetul curent

VARIABILE SI METODICE STATICE

Variabile si metode statice

- metoda de clasa / variabila de clasa
- Accesibile prin numele clasei
- Echivalentul metodelor si variabilelor globale din alte limbaje
- Variabilele statice:
 - O singura locatie accesibila din toate obiectele prin numele clasei
- Exemplu utilizare: clasa Math
- Exemplu utilizare: sablonul de proiectare Singleton

CONSTRUCTORII

Motivatie

- Constructurii sunt metode speciale ce sunt apelate automat atunci cand obiectele sunt create
- Sintaxa:

```
public class MyClass{  
    public MyClass(...){...}  
}
```
- Numele identic cu numele clasei
- Nu au tip de return
- Sunt utilizati pentru initializarea starii obiectului sau pentru a efectua anumite operatii specifice la initializarea obiectului

Constructorul implicit

- Este constructorul fara argumente
- Este automat adaugat de compiler daca nu este definit de programator

Exemplu

- Exemplu constructor implicit pentru clasa Robot

Constructori cu argumente

- O clasa poate avea mai multi constructori
- Difera prin numarul si tipul argumentelor
- Cuvantul cheie “this” pentru a apela attributele din cadrul instantei
- Se utilizeaza “this” pentru a forta apelare unui constructor din alt constructor

Exemplu

- Exemplu constructori cu argumente si this

SFARSIT