VILNIAUS TECHNOLOGIJŲ IR VERSLO
PROFESINIO MOKYMO CENTRAS

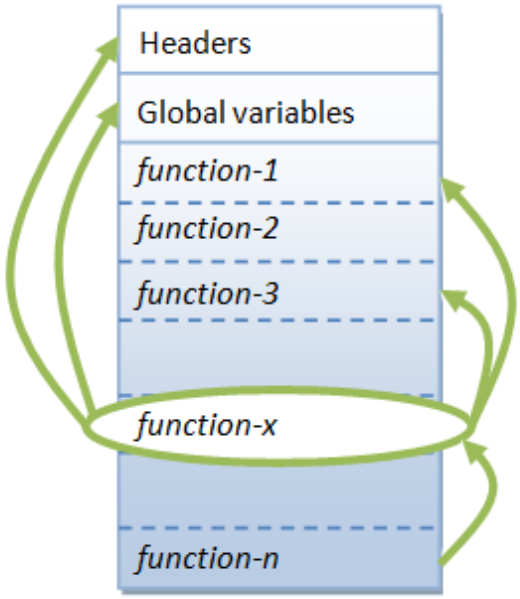
8 - OBJEKTINIS PROGRAMAVIMAS

Jaroslav Grablevski / Justina Balsė

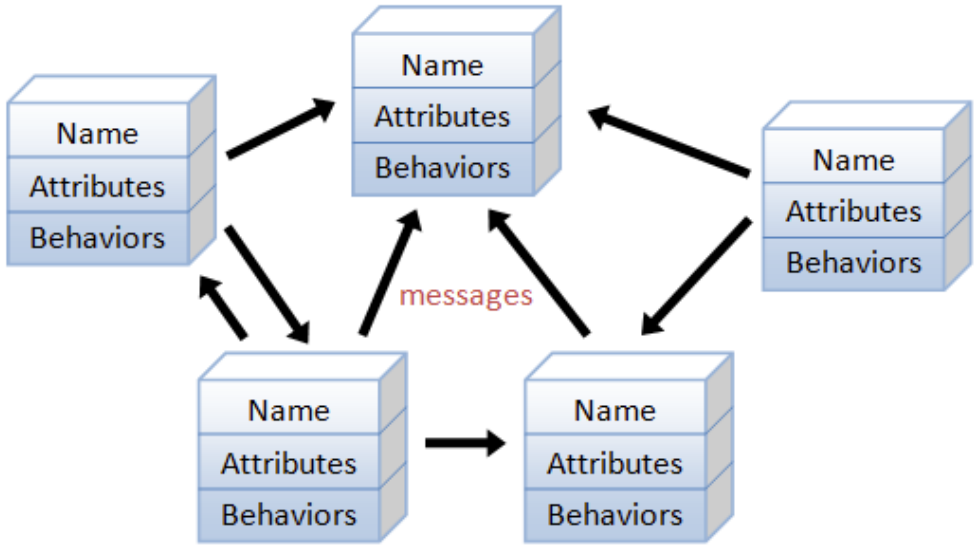
Turinys

- Pagrindinės sąvokos
- Objektas
- Klasė
- Kintamieji
- Metodai
- Konstruktoriai
- Inkapsuliacija

Procedural vs. Object-Oriented



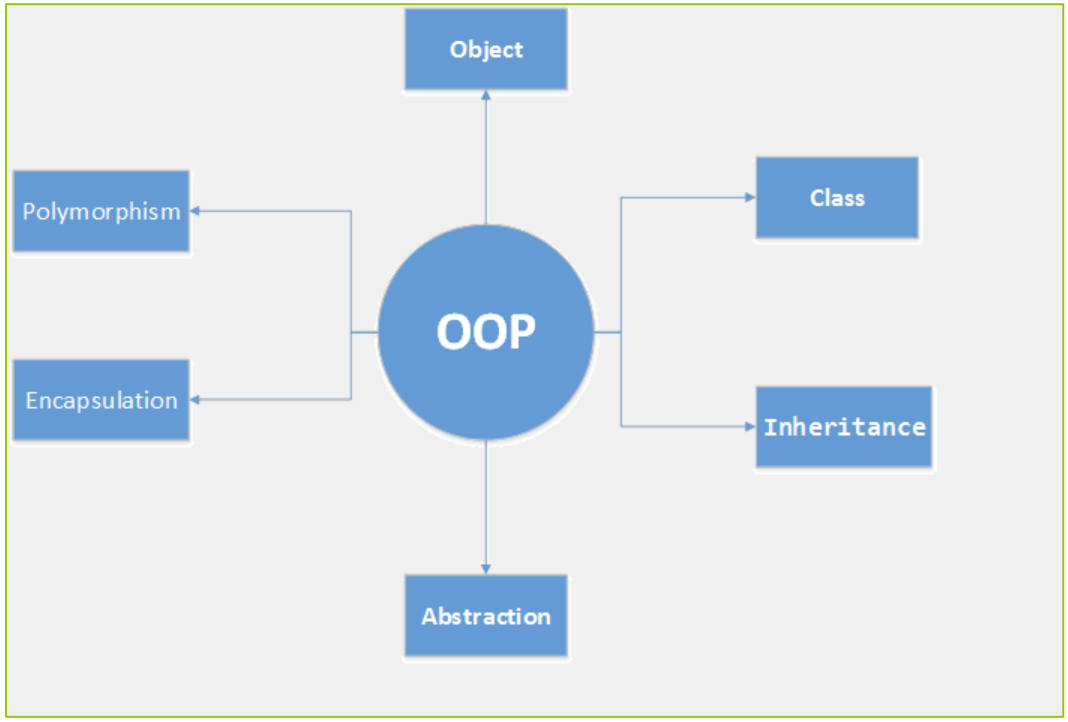
A function (in C) is not well-encapsulated



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

Pagrindinės OOP sąvokos ir terminai

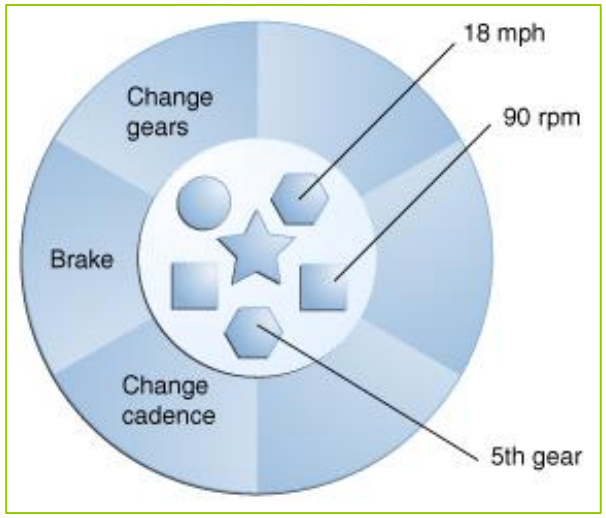
- Klasė
- Objektas
- Abstrakcija
- Inkapsuliacija
- Paveldėjimas
- Polimorfizmas



Objektas

Tai savarankiškas programinis komponentas turintis šias savybes:

- Tapatybė *(identity (or name))*
- Būsena *(**state** (or attributes, variables, fields))*
- Elgesys *(**behavior** (or methods))*



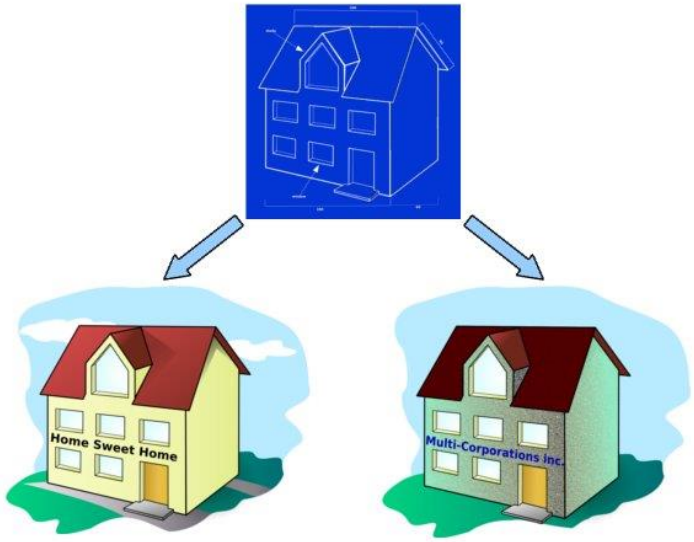
Bicycles have state (current gear, current pedal cadence, current speed) and behavior (changing gear, changing pedal cadence, applying brakes)

Abstrakcija (abstraction)

- Tai yra apibendrinimo principas, kuris nusako, kad konkrečius dalykus galime apibendrinti, išskiriant esmines ir pačias svarbiausias jų savybes bei atsiribojant nuo specifinių detalių.
- Tai yra viena iš pagrindinių priemonių kovojant su programinės įrangos sudėtingumu.

Klasė

- Klasė – tai objekto šablonas (brėžinys), kurį sudaro duomenys (kintamieji, laukai) ir metodai.
- Iš vienos klasės galima sukurti kiek norima objektų, tačiau kiekvienas objektas yra sukurtas iš vienos konkrečios klasės.



Klasė

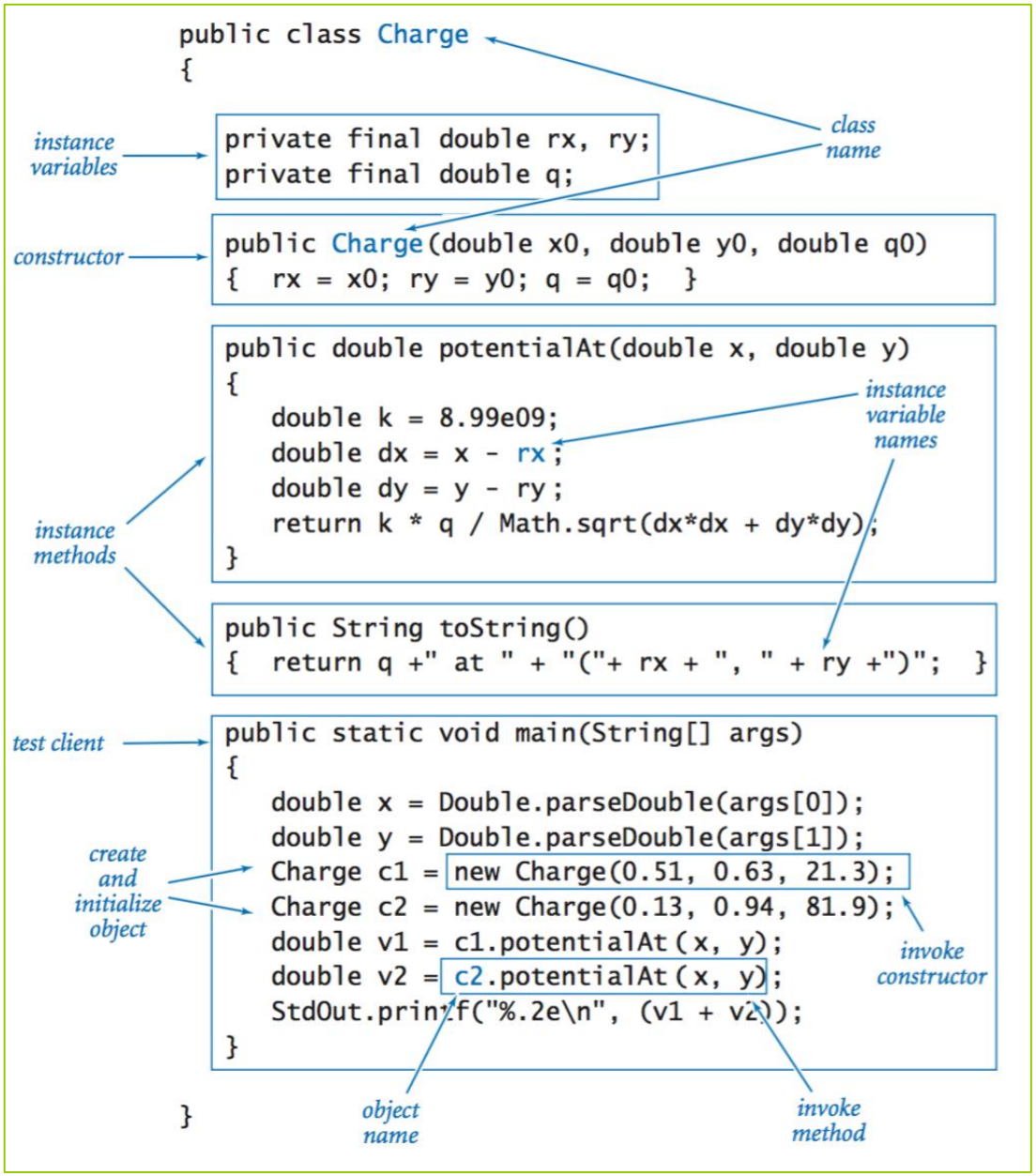
- Klasės kūnas susideda iš tokių elementų:
  - Objekto kintamųjų
  - Konstantų
  - Konstruktorių
  - Metodų
  - Statinių inicializavimo blokų
  - Inicializavimo blokų
  - Klasių
  - Interfeisų

**Nė vienas nėra privalomas*

Klasė

```
public class Bicycle {

    // fields
    int cadence;
    int gear;
    int speed;

    // constructor
    Bicycle(int cadence, int gear, int speed) {
        this.cadence = cadence;
        this.gear = gear;
        this.speed = speed;
    }

    // method
    public void starting(){
    }
}
```

Klasė


public class Charge
{
private final double rx, ry;
private final double q;
public Charge(double x0, double y0, double q0)
{ rx = x0; ry = y0; q = q0; }
public double potentialAt(double x, double y)
{
double k = 8.99e09;
double dx = x - rx;
double dy = y - ry;
return k * q / Math.sqrt(dx*dx + dy*dy);
}
public String toString()
{ return q +" at " + "("+ rx + ", " + ry +")"; }
public static void main(String[] args)
{
double x = Double.parseDouble(args[0]);
double y = Double.parseDouble(args[1]);
Charge c1 = new Charge(0.51, 0.63, 21.3);
Charge c2 = new Charge(0.13, 0.94, 81.9);
double v1 = c1.potentialAt(x, y);
double v2 = c2.potentialAt(x, y);
StdOut.printf("%.2e\n", (v1 + v2));
}
}
instance variables
class name
constructor
instance methods
instance variable names
test client
create and initialize object
invoke constructor
object name
invoke method

Objekto kintamieji (instance variables)

- Deklaruojami klasės ribose (bet ne metoduose, konstruktoriuose ar blokuose)
- Vadinami objekto laukais *(fields)*
- Priklauso konkrečiam klasės egzemplioriui/objektui (kiekvienas objektas turi nuosavas šių kintamųjų kopijas)
- Atmintis šiems kintamiesiems išskiriama kuriant objektą
- Prieinami:
  - Metoduose;
  - Konstruktoriuose;
  - Blokuose;
- Galima naudoti prieinamumo modifikatorius
- Priskiriamos numatytosios reikšmės
- Pasiekiami naudojant: <objektoNuoroda>.kintamasis
- Objekto kintamieji nėra pasiekiami statiniams metodams

Klasės kintamieji (static class members)

- Deklaruojami klasės ribose (bet ne metoduose, konstruktoriuose ar blokuose)
- Priklauso konkrečiai klasei (viena kopija kintamųjų vienai klasei, nepriklausomai nuo to, kiek sukurta klasės objektų)
- Atmintis šiems kintamiesiems išskiriama programos startavimo metu
- Pasiekiami naudojant: <KlasėsPavadinimas>.kintamasis
- Dažniausiai naudojami kaip konstantos, prieinamos ir kitiems objektams

Kintamųjų galiojimo zona (scope)

- Kintamieji, paskelbti klasėje, galioja visai klasei, t.y. galioja visiems klasės metodams ir klasės vidinėms klasėms.
- Metodo kintamieji galioja tik tam metodui.
- Metodo parametrai galioja tik tam metodui.

Klasės kintamieji (static class members)

```
// Static variable used to maintain a count of the number of
// Employee objects in memory.

public class Employee {
    private static int count = 0; // number of Employees created
    private String firstName;
    private String lastName;

    // initialize Employee, add 1 to static count and
    // output String indicating that constructor was called
    public Employee(String firstName, String lastName) {
        this.firstName = firstName;
        this.lastName = lastName;

        ++count; // increment static count of employees
    }

    // static method to get static count value
    public static int getCount() {
        return count;
    }

}
```

Konstantos

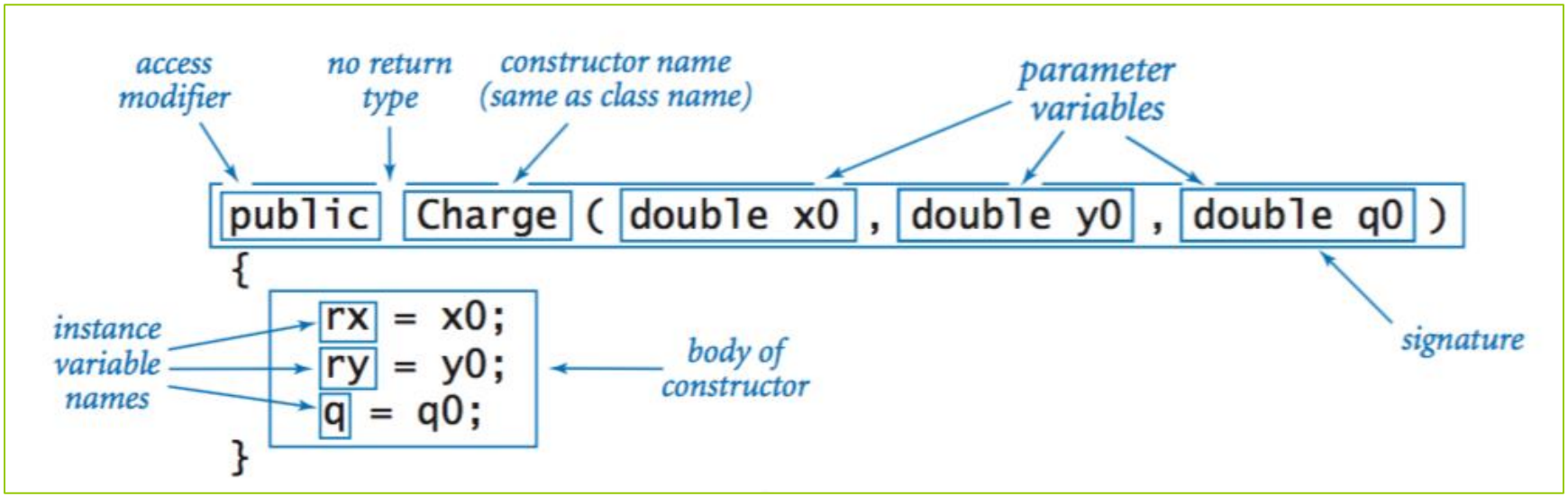
- Java programavimo kalboje konstanta yra realizuojama naudojant žodelį ***final***, kuris nusako, kad reikšmė, kuri yra priskirta nekintama.

```
//klasės konstantos
public static final float PI = 3.14f;
public static final double G = 6.674 * Math.pow(10,-11);
public static final int C = 299_792_458;
public static final int ZERO = 0;

//objekto konstantos
private final String name;
private final int radius;
```

Konstruktorius

- Konstruktoriai skirti klasės inicializavimui - tai yra klasės būsenos paruošimui tuomet, kai sukuriamas objektas.
- Klasės konstruktoriai yra pagrindinė priemonė klasės objektams kurti.
- Panašus į metodą
- Vardas sutampa su klasės
- Neturi grąžinamo tipo
- Gali būti keli (0..*)

Konstruktorius



Konstruktorius

```
// fields
int cadence;
int gear;
int speed;

// constructors
Bicycle() {
    this(0, 0, 0);
}

Bicycle(int cadence, int gear) {
    this.cadence = cadence;
    this.gear = gear;
}

Bicycle(int cadence, int gear, int speed) {
    this.cadence = cadence;
    this.gear = gear;
    this.speed = speed;
}
```

Konstruktorius

- Klasė gali turėti daug konstruktorių, kurie gali būti perkrauti *(overloaded)*
- Būtina sąlyga – konstruktoriai turi skirtis savo antraštėmis, t.y. parametrų skaičiumi arba parametrų tipais.

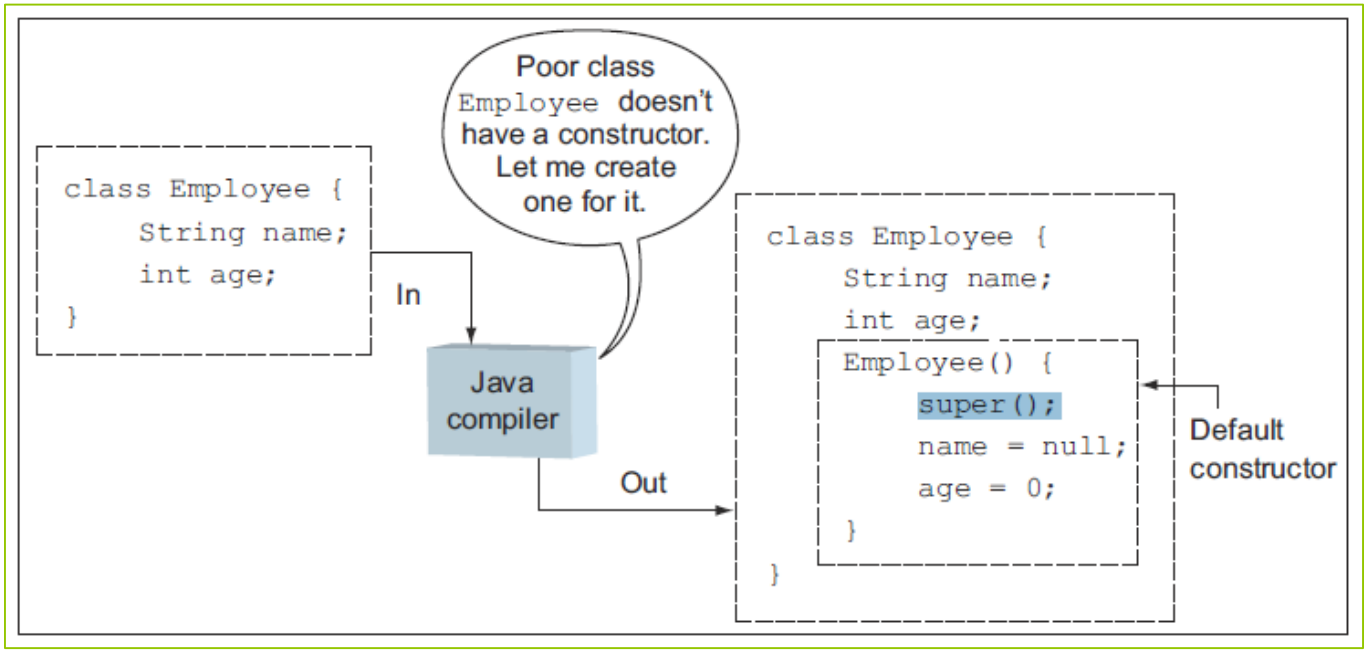
Konstruktorius > this

- Sakinys ***this***([<<argumentai>>]) yra naudojamas klasės konstruktoriuose iškviesti kitiems konstruktoriams.
- Šis sakinys turi būti pats pirmasis konstruktoriuje.

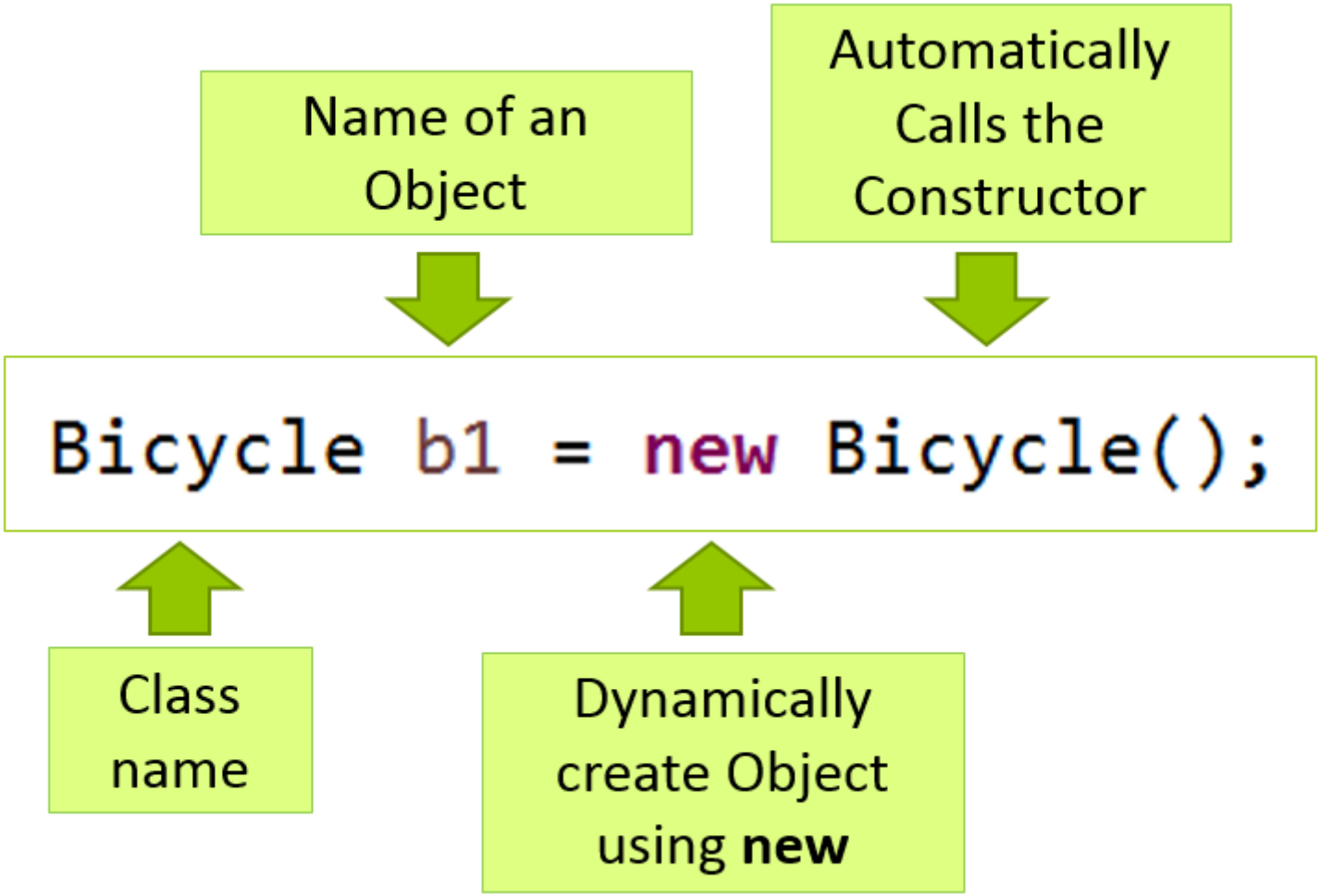
```
class Person{
String name;
int age;

Person() {
        this("Anonymous", 0);
    }

Person(String name, int age) {
        this.name = name;
        this.age = age;
    }
}
```

Konstruktorius

- Jeigu konstruktorius nėra naudojamas, kuriant objektą kviečiamas numatytasis konstruktorius, kuris visiems klasės kintamiesiems (laukams) suteikia ,,nulines“ reikšmes (pagal tipą).



Kaip sukurti objektą?



this

- Žodelis this, tai nuorodą į dabartinį objektą - nuoroda objekto, kurio metodas ar konstruktorius konkrečiu metu yra vykdomas.
- Dažniausiai naudojamas tais atvejais, kai metodo ar konstruktoriaus parametrų pavadinimai sutampa su objekto kintamųjų vardais.
- Papildomai, jis gali būti naudojamas tose vietose, kur reikalinga dabartinio objekto nuoroda.

```
public class Point {
    int x = 0;
    int y = 0;

    public Point(int x, int y) {
        this.x = x;
        this.y = y;
    }
}
```

Metodai

- Metodai gali būti dvieju tipų:
  - **Klasės** (statiniai) metodai
    - Priklauso klasei
    - Norint iškviesti, nebūtinas objekto sukūrimas - užtenka klasės pavadinimo: *<klasėsPavadinimas>.<metodoPavadinimas>([<<argumentai>>])*
    - Gali dirbti tik su klasės (statinius) kintamaisiais
    - Statiniai metodai pažymimi žodeliu ***static***
  - **Objekto** metodai
    - Priklauso objektui
    - Norint iškviesti, būtinas objekto sukūrimas
    - Gali dirbti tiek su klasės, tiek su objekto kintamaisiais

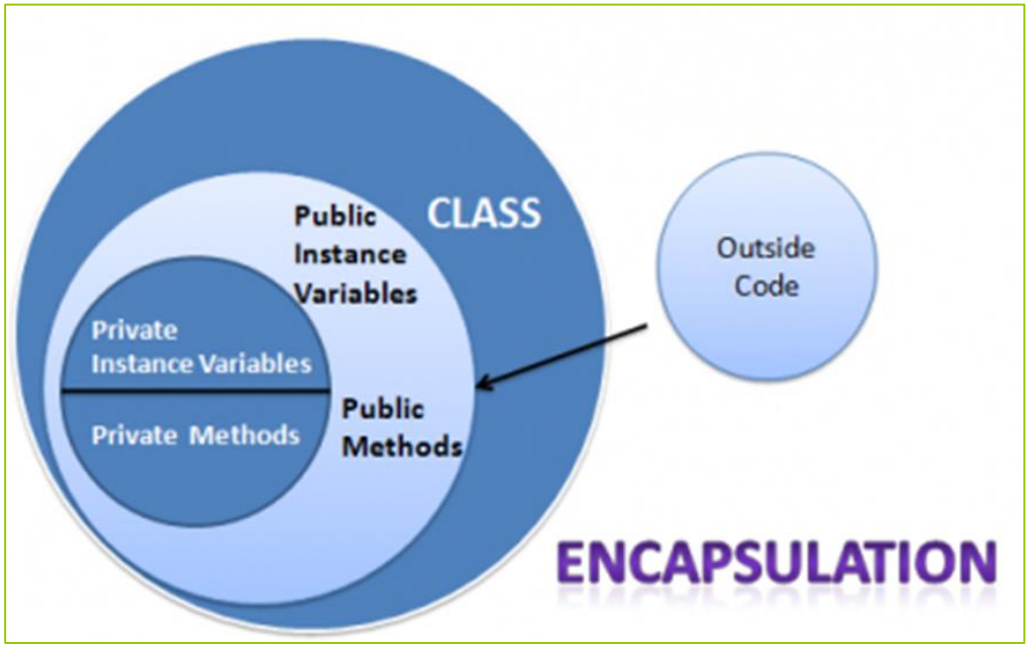
Metodai (overloading)

- Metodai gali būti perkrauti, t.y. klasėje gali būti deklaruoti daugiau nei vienas metodas su tuo pačiu pavadinimu.
- Svarbu, kad būtų tenkinamos šios taisyklės:
  - Parametrų kiekis turi skirtis
  - Turi skirtis parametrų tipai

```
class Person{
    String name;
    int age;

    void speak(){

    void speak(int times){

    void speak(String lastName){

    void speak(int times, String lastName){

}
```

Inkapsuliacija (encapsulation)

- Veiksmai ir duomenys supakuoti kartu
- Programuotojas renkasi ką rodyti, o ką slėpti nuo objekto naudotojo (kito programuotojo)



Paketas

- Paketas - tai vardų erdvė (angl. namespace), kurioje laikoma aibė susijusių klasių ir interfeisų. Paketai leidžia organizuoti programinius komponentus, panašiai, kaip katalogai leidžia organizuoti failus asmeniniame kompiuteryje.

- Papildomai, paketas leidžia slėpti tam tikras detales nuo kituose paketuose esančių klasių (inkapsuliacija), tokias kaip:
  - Klasės
  - Interfeisai
  - Kintamieji
  - Metodai

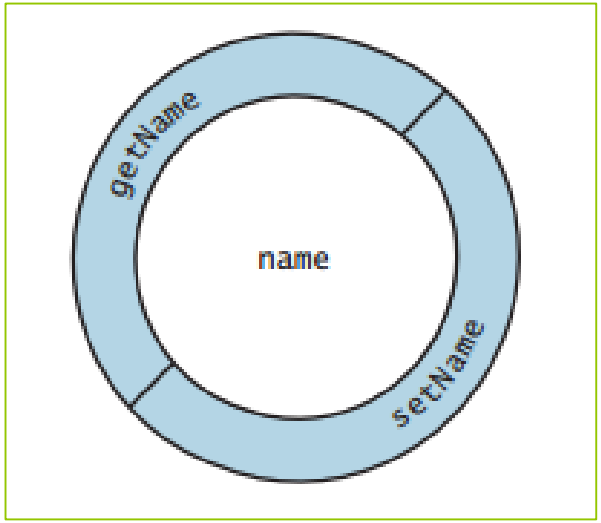
Modifikatoriai / access modifiers

- Kintamųjų ir metodų galiojimo zoną papildomai reguliuoja modifikatoriai.

Modifier	Same class or nested class	Other class inside the same package	Extended Class inside another package	Non-extended inside another package
private	yes	no	no	no
default (package private)	yes	yes	no	no
protected	yes	yes	yes	no
public	yes	yes	yes	yes

Set ir Get metodai

- Klasės metodai, prasidedantys priešdėliais ***get*** ir ***set*** (***is*** ir ***set*** boolean tipo kintamiesiems), skirti objekto savybėms nuskaityti ir pakeisti.




```
class Person{
    private String name;
    private int age;

    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public int getAge() {
        return age;
    }
    public void setAge(int age) {
        this.age = age;
    }
}
```

Set ir Get metodai

```
public class Example3 {

    public static void main(String[] args) {

        Person p1 = new Person();

        p1.setName("William");
        p1.setAge(31);

        System.out.println(p1.getName() + " is " + p1.getAge() + " years old.");
    }
}
```

```
William is 31 years old.
```

Metodas toString()

```
class Person{

}

public class ExampletoString {

    public static void main(String[] args) {
        Person p1 = new Person();
        System.out.println(p1);
    }
}
```

```
Person@15db9742
```

Metodas toString()

```
class Person{

    private String name;
    private int age;

    public Person(String name, int age){
        this.name = name;
        this.age = age;
    }

    @Override
    public String toString() {
        return name + " is " + age + " years old.";
    }

}

public class ExampletoString {

    public static void main(String[] args) {
        Person p1 = new Person("Wiliam", 31);
        System.out.println(p1);
    }
}
```

```
Wiliam is 31 years old.
```