LATVIJAS UNIVERSITATE
ANNO 1919

NACIONĀLAIS ATTĪSTĪBAS PLĀNS 2020

EIROPAS SAVIENĪBA
Eiropas Sociālais fonds

IEGULDĪJUMS TAVĀ NĀKOTNĒ

Java programmēšanas pamati

VUMC
VADĪBAS UN UZŅĒMĒJDARBĪBAS MĀCĪBU CENTRS

3. NODARBĪBA

DATU TIPI

KĻŪDAS UN ZĒMŪDENS AKMEŅI

PĀRSKATS

1. Trūkst noslēdzošā simbola ';'
2. Kļūdaina pareizrakstība
    2.1. Klases nosaukumā
    2.2. Pakotnes nosaukumā
    2.3. Mainīgā nosaukumā
3. Kods izvietots ārpus metodes ķermeņa
4. Trūkst vai nepareizi novietotas pēdiņas

(1) TRŪKST NOSLĒDZOŠĀ SIMBOLA

```
public class ForgotSemicolonAgain {

    public static void main(String[] args) {
        System.out.println("Oops.. I did it again")
    }

}
```

(1) TRŪKST NOSLĒDZOŠĀ SIMBOLA

IZLABOTS

```
public class ForgotSemicolonAgain {

    public static void main(String[] args) {
        System.out.println("Oops.. I did it again");
    }

}
```

(2.1) KĻŪDAINS KLASES NOSAUKUMS

```
public class sizeMatters {

    public static void main(String[] args) {
        system.out.println("Sorry, it does");
    }

}
```

(2.1) KĻŪDAINS KLASES NOSAUKUMS

IZLABOTS

```
public class SizeMatters {

   public static void main(String[] args) {
      System.out.println("Sorry, it does");
   }

}
```

(2.2) KĻŪDAINS PAKOTNES NOSAUKUMS

```
package lv.javaguru.lessons.HOMEWork;
```

I

(2.2) KĻŪDAINS PAKOTNES NOSAUKUMS

IZLABOTS

```
package lv.javaguru.lessons.homework;
```

(3) KODS ĀRPUS ĶERMEŅA

```
public class AttentionPlease {

    System.out.println("Hide and seek");
    public static void main(String[] args) {
    }

}
```

(3) KODS ĀRPUS ĶERMEŅA

IZLABOTS

```
public class AttentionPlease {

    public static void main(String[] args) {
        System.out.println("Hide and seek");
    }

}
```

(4) PĒDIŅU IZVIETOJUMS

```
public class NoSleepNoFocus {

    public static void main(String[] args) {
        System.out.println(I wanna coffee);
        System.out.println("So bad);
    }

}
```

(4) PĒDIŅU IZVIETOJUMS

IZLABOTS

```
public class NoSleepNoFocus {

    public static void main(String[] args) {
        System.out.println("I wanna coffee");
        System.out.println("So bad");
    }

}
```

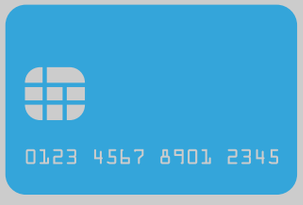
OBJEKT-ORIENTĒTĀ PROGRAMMĒŠANA

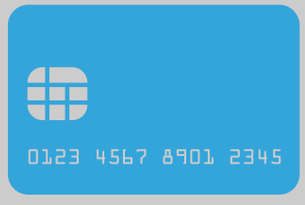
KLASE UN OBJEKTS

- Klase ir apraksts (šablons) “kaut kam” kam piemīt stāvoklis un uzvedība
- Objekts ir konkrēta instance (vienība) no tās klases ar noteiktu stāvokli

BANKAS KARTE (STĀVOKLIS)

Klase		Objekts
A. Bankas nosaukums	→	A. Citadele Banka
B. Maksājumu izpildītajs	→	B. Master Card
C. Vārds uz kartes	→	C. John Doe
D. Kartes numurs	→	D. 5224 9989 7556 2871
E. Derīguma datums	→	E. 12/2022
F. Drošības kods	→	F. 218

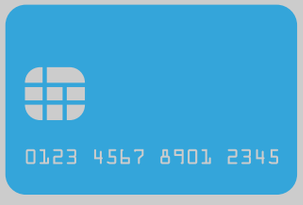




BANKAS KARTE (UZVEDĪBA)

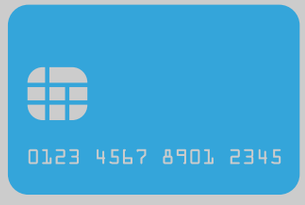
Klase

A. Nosakidrot atlikumu
B. Noguldīt līdzekļus
C. Izņemt līdzekļus



Objekts

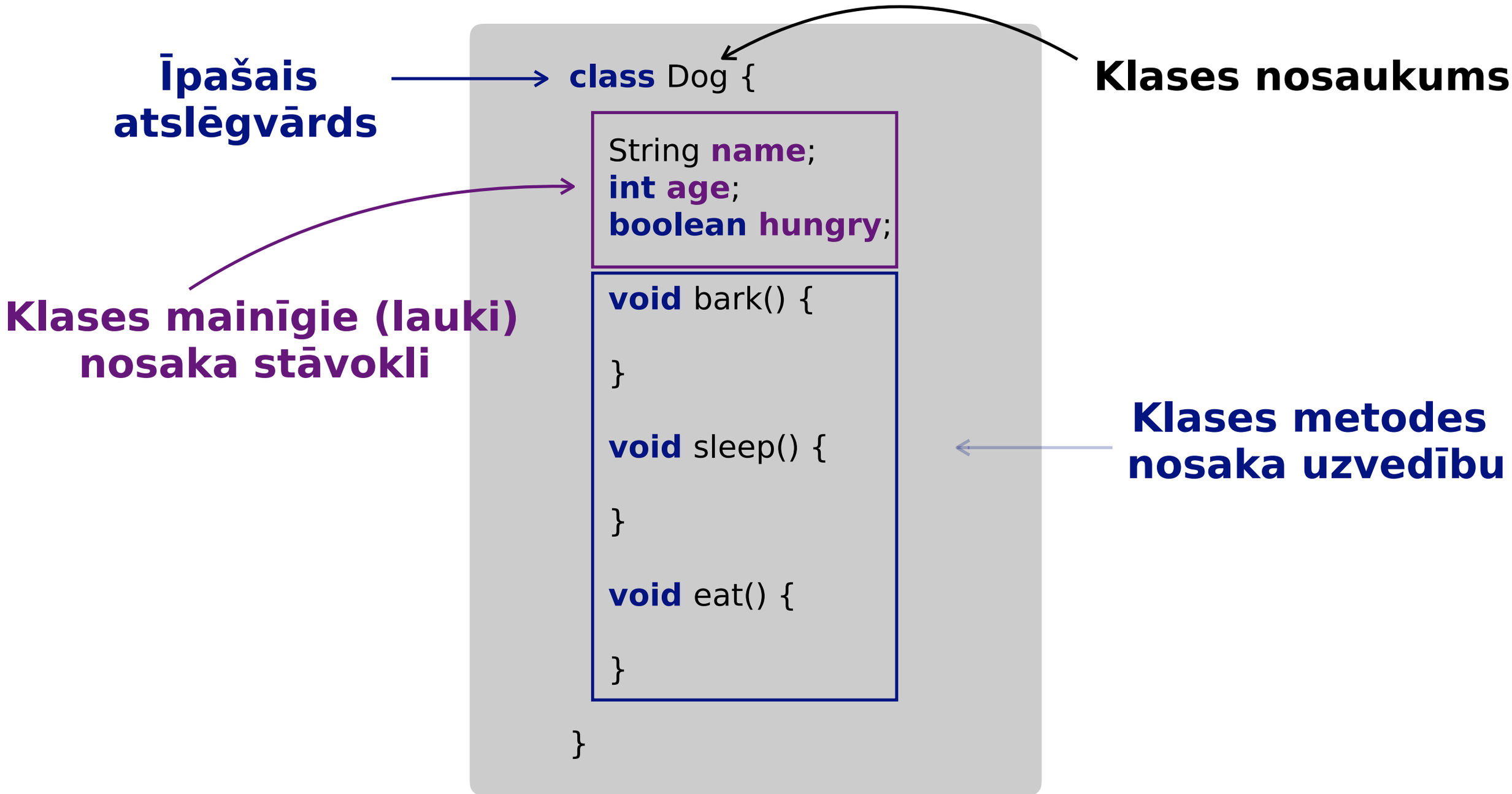
A. Nosakidrot atlikumu
B. Noguldīt līdzekļus
C. Izņemt līdzekļus



KLASES DEKLARĀCIJA: SINTAKSE

```
class ClassName {

    type variable1;
    type variable2;
    ...
    type variableN;

    method1() {}
    method2() {}
    ...
    methodN() {}

}
```

KLASES DEKLARĒŠANA: DETAĻĀS



OBJEKTA IZVEIDE

- Objekta izveide bez piešķiršanas

```
new Class();
```

- Objekta izveide ar piešķiršanas

```
Class var = new Class();
```

OBJEKTA IZVEIDE: SINTAKSE

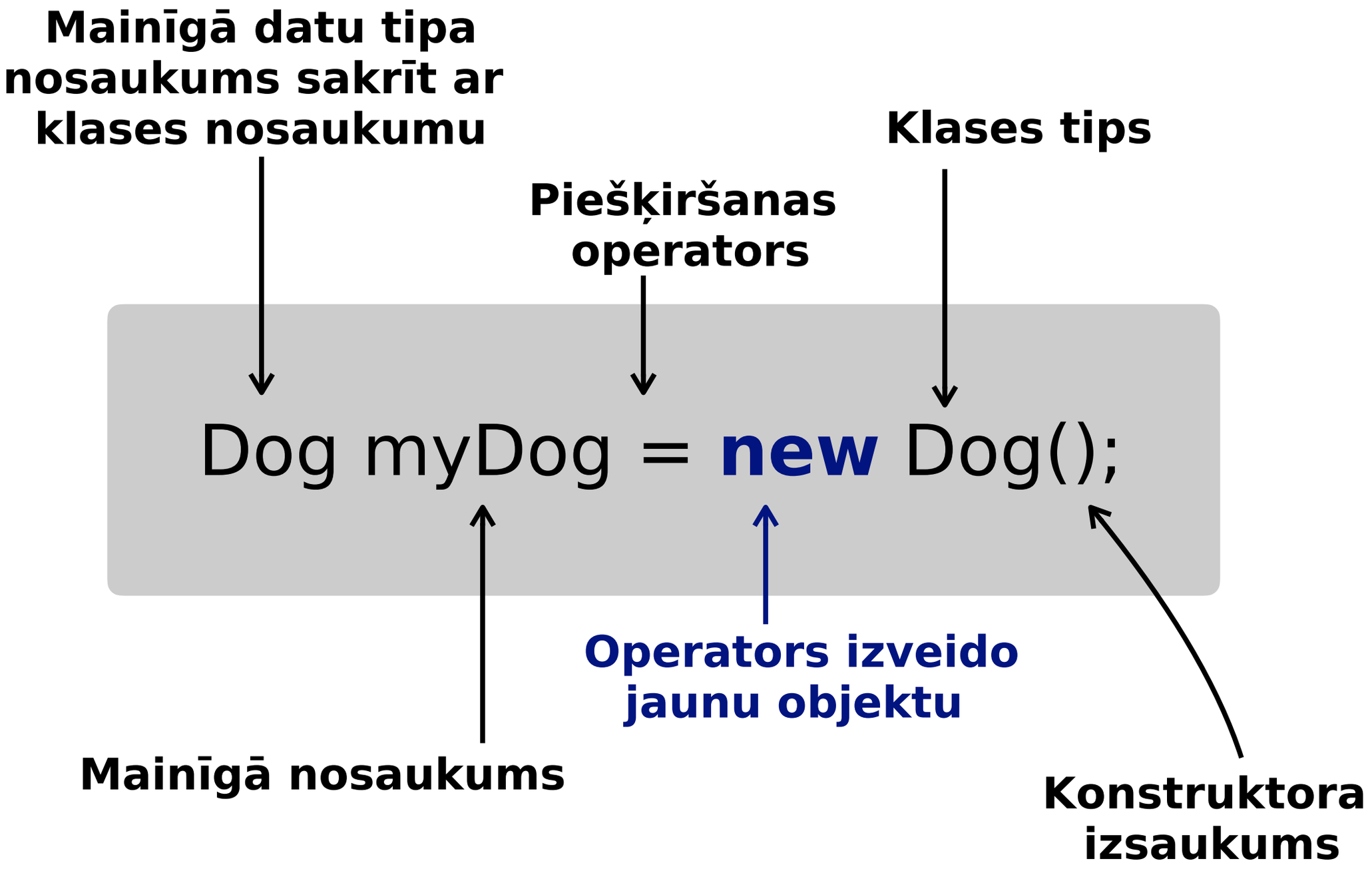
- Objekta izveide bez piešķiršanas

```
2 new Dog();
```

- Objekta izveide ar piešķiršanu

```
Dog myDog = new Dog();
```

OBJEKTA IZVEIDE



OBJEKTA IZVEIDE

1. Deklarēšana (declaration) – objekta mainīgā deklarēšana ar klases tipu
2. Izveide (instantiation) – objekta izveidošana izmantojot operatoru new
3. Inicializācija (initialisation) - objekta uzbūvēšana, iestatot tā sākotnējo stāvokli

KONSTRUKTORS

- Katrai klasei ir savs konstruktors
- Ja atsevišķi kostruktors(i) nav definēts kodā, tad Java kompilators izveidos noklusēto konstruktoru netieši
- Lai izveidotu jaunu objektu, tiks izsaukts vismaz viens konstruktors
- Katram konstruktoram ir jābūt unikālam parakstam jeb “signature”(sakārtots argumentu skaits un tips)

KONSTRUKTORA DEKLARĒŠANA

Nolkusētais konstruktors bez argumentiem

```
public class Dog {

    private String name;

    public Dog() {

    }

    public Dog(String name) {
        this.name = name;
    }

}
```

Atsevišķs konstruktors ar argumentiem un inicializāciju

ATMIŅA PĀRSKATS

ATMIŅAS VEIDI

- Java Heap Atmiņa
  - Izveidotie objekti tiek glabāti “Java Heap atmiņā”
  - Objekts “dzīvo” atmiņā no programmas darbības sākuma līdz beigām
  - “Java Heap Atmiņā” saglabātie objekti ir globāli pieejami
- Java Stack Atmiņa
  - Satur lokālos primitīvos mainīgos un atsauces mainīgos objektiem, kas atrodas “Java Heap Atmiņā”
  - Dzīvo tikai metodes izpildes laikā, īslaicīgi
  - Piesaistīts tekošajai izpildes plūsmai (current execution thread)

METODES PĀRSKATS

METODES DEFINĒŠANA

- Metode ir instrukciju kopums, kas ir sagrupētas noteikta uzdevuma veikšanai
  - Teksta attēlošanai konsolē izsauc metodi System.out.println(), kas patiesībā izpilda vairākas instrukcijas
- Apraksta klases uzvedību jeb darbības, kuras objekts spēj veikt
- Metode var dot vai nedot rezultātu

SINTAKSE

Metodes piekļuves tips

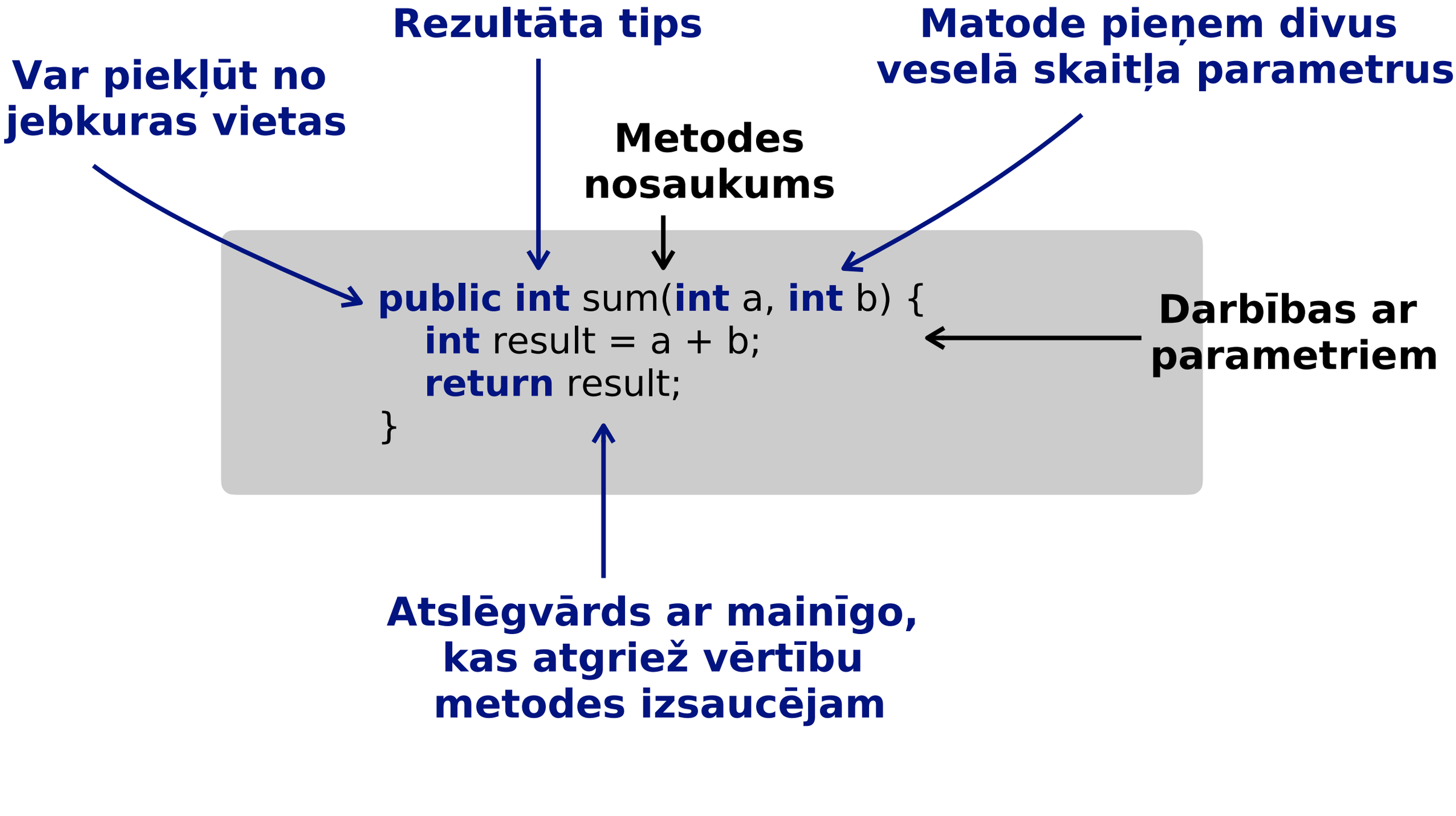
Metodes nosaukums

```
modifier returnType methodName (arg1, arg2, ..., argN) {

    //body

}
```

Metodes rezultāta tips

Metodes parametru saraksts (tips un nosaukms)

PIEMĒRS (AR REZULTĀTU)



PIEMĒRS (BEZ REZULTĀTA)

Metodes nosaukums

Atslēgvārds – bez rezultāta

Matode pieņem divus veselā skaitļa parametrus

```
public void printSum(int a, int b) {
    int result = a + b;
    System.out.println("Sum of numbers is" + result);
}
```

Darbības ar parametriem

ATGRIEŽAMAIS REZULTĀTS

- Pēc izpildes, metode atgriežas pie tās izejas koda rindiņas, kura to ir izsaukusi
- Tas, vai metode atgriež vērtību, tiek deklarēts metodes parakstā (method signature)
  - Tips void - return paziņojumu nav nepieciešams, taču to var norādīt
  - Citi tipi - return paziņojums ir obligāts

“GETERI” UN “SETERI”

- OOP nevienam nevajadzētu būt iespējai tieši piekļūt objekta stāvoklim
- Lai viss būtu drošībā, var
  - Izgūt objekta stāvokli izmantojot “get” metodes (geterus)
  - Mainīt objekta stāvokli izmantojot “set” metodes (seterus)

GETERU UN SETERU PIEMĒRS

```
public 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;
    }
}
```

Geteri

Seteri

GETERU UN SETERU LIETOJUMS

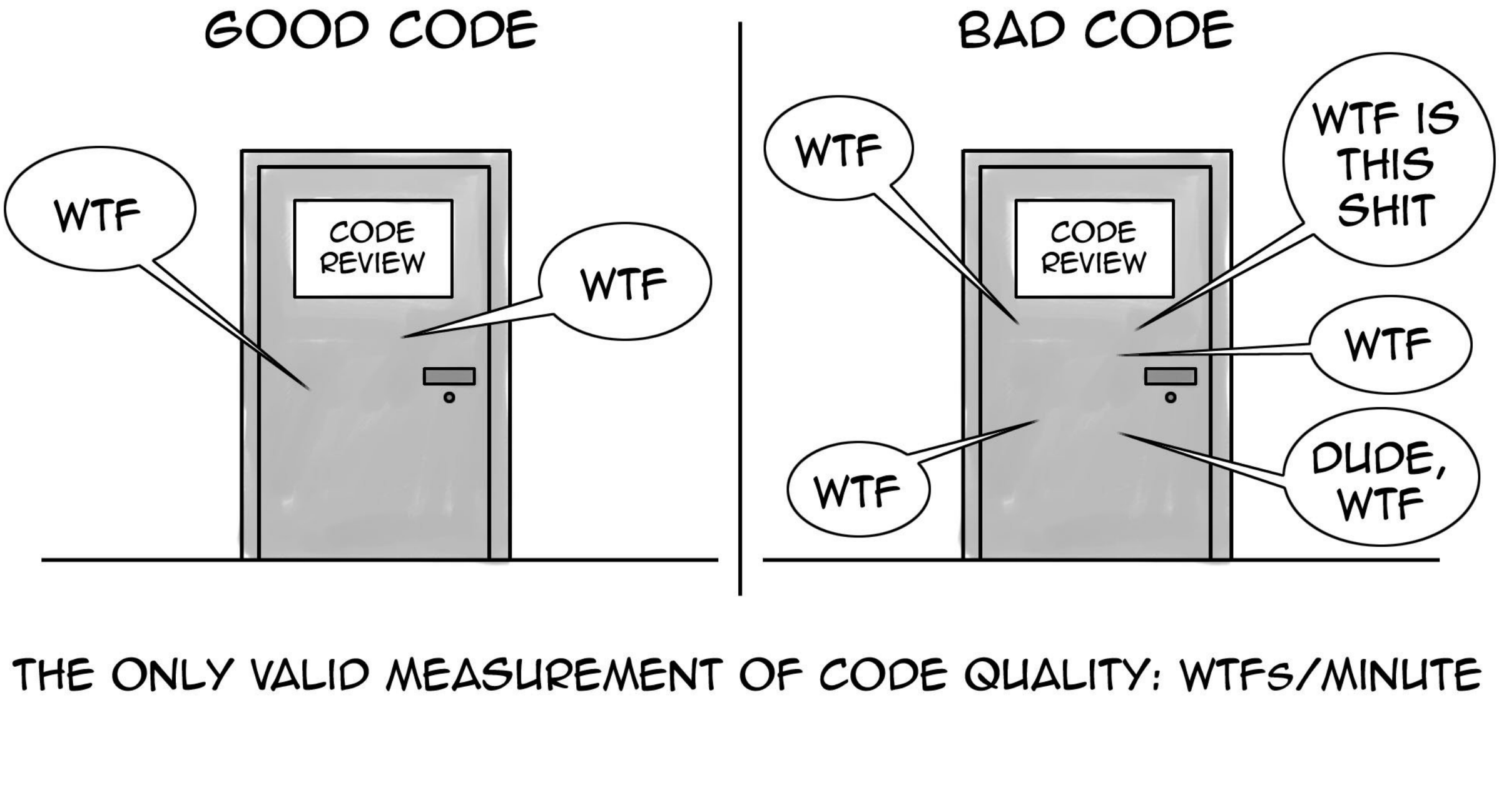
```
public class PersonTest {

    public static void main(String[] args) {

        Person person = new Person();
        person.setName("John Doe");
        person.setAge(32);

        String personName = person.getName();
        int personAge = person.getAge();

        System.out.println("His name is " + personName);
        System.out.println("He is " + personAge + " years old");

    }

}
```

“CLEAN CODE“

PRAKSE

JEBKURŠ MULĶIS VAR UZRAKSTĪT KODU, KURU DATORS SAPRATĪS. LABS PROGRAMMĒTĀJS RAKSTA KODU, KURU CILVĒKI SAPROT.

Martin Fowler


GOOD CODE
WTF
CODE REVIEW
WTF
BAD CODE
WTF
CODE REVIEW
WTF IS THIS SHIT
WTF
WTF
DUDE, WTF
THE ONLY VALID MEASUREMENT OF CODE QUALITY: WTFS/MINUTE

SLIKT KODS UN LABS KODS

Slikts

```
public class Cat {

    private String n;

    public String getN() {
        return n;
    }

    public void setN(String n) {
        this.n = n;
    }

    public void v() {
        System.out.println("Meow");
    }

}
```

Labs

```
public class Cat {

    private String name;

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public void voice() {
        System.out.println("Meow");
    }

}
```

ATSAUCES

- https://docs.oracle.com/javase/tutorial/java/javaOO/methods.html
- https://www.tutorialspoint.com/java/java_methods.htm

LATVIJAS UNIVERSITATE
ANNO 1919

VADĪBAS UN UZŅĒMĒJDARBĪBAS MĀCĪBU CENTRS