



# Paveldėjimo savybės

Martynas Mitrulevičius

# Klases laukų paveldėjimas

```
class A {  
    int i = 1;  
  
    String metodas() {  
        return "A";  
    }  
}  
  
class B extends A {  
    int i = 100;  
  
    @Override  
    String metodas() {  
        return "B";  
    }  
}
```

```
B objektas = new B();  
System.out.println(objektas.i);           // 100  
System.out.println(objektas.metodas()); // B
```

```
A kitasObjektas = new B();  
System.out.println(kitasObjektas.i);      // 1  
System.out.println(kitasObjektas.metodas()); // B
```

# Polimorfizmas

- Situacija, kai objekto reakciją į metodo kvietimą nulemia objekto tipas vadinamas **polimorfizmu**.
- Polimorfizmas įgalina:
  - programuoti apibendrintai, realizacijos detales apibrėžiant išvestinėse klasėse
  - specializuoti/modifikuoti klasės funkcionavimą išvestinėje klasėje.
- Java kalboje kiekvienas paveldėtos klasės objektas gali būti naudojamas ten, kur reikalingas tėvinės klasės objektas
- Privalumai:
  - Programinis kodas lengvai rašomas ir skaitomas
  - Klasės sąsaja vienoda, tipų specifika svarbi realizacijoje
- Trūkumai:
  - Jei reikia vaikinės klasės savybių, tada reikia atlikti cast operaciją

# Polimorfizmas

```
abstract class Car {  
    abstract void go();  
}  
  
class Bus extends Car {  
    @Override  
    void go() {  
        System.out.println("Bus goes...");  
    }  
}  
  
class Train extends Car {  
    @Override  
    void go() {  
        System.out.println("Train goes...");  
    }  
}
```

```
public class RunCars {  
  
    public static void main(String[] args) {  
  
        Train a = new Train();  
        Bus b = new Bus();  
        Car c = new Bus();  
  
        makeMove(a);  
        makeMove(b);  
        makeMove(c);  
  
    }  
  
    static void makeMove(Car c) {  
        c.go();  
    }  
}
```

Train goes...  
Bus goes...  
Bus goes...

# Kompozicija vs paveldėjimas

```
class A {  
  
}
```

```
class B {  
  
    private A kintamasisA;  
  
}
```

## Kompozicija.

Viena klasė turi kitos klasės objektą kaip klasės kintamąjį.

Ryšys vadinamas „**has-a**“

```
class A {  
  
}
```

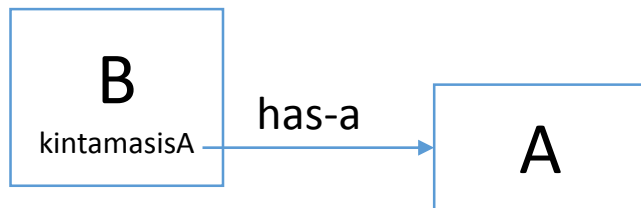
```
class B extends A {  
  
}
```

## Paveldėjimas.

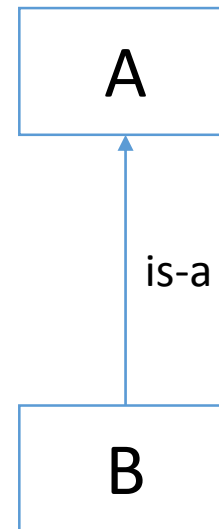
Viena klasė paveldi kitą klasę.

Ryšys vadinamas „**is-a**“

# Kompozīcija vs pārvērtības



Kompozīcija



Pārvērtības

# Perrašymas vs perkrovimas

```
class A {  
    int metodas() {  
        return 5;  
    }  
    // perkrovimas (angl. overloading)  
    int metodas(int k) {  
        return k + 1;  
    }  
}  
  
class B extends A {  
    // perrašymas/užklojimas (angl. overriding)  
    @Override  
    int metodas() {  
        return 6;  
    }  
}
```

# Modifikatorius "protected"

- Leidžia panaudoti **protected** narį tik išvestinėse klasėse
- Išvestinėse klasėse narių matomumas negali būti apribotas.
- Galime tik sumažinti apribojimą:

Tėvinė klasė	Vaikinė klasė
<b>private</b>	- (negalime perrašyti tokio nario)
(default)	(default) <b>protected</b> <b>public</b>
<b>protected</b>	<b>protected</b> <b>public</b>
<b>public</b>	<b>public</b>



# Klasės laukų pasiekiamumas

Modifikatoriai	Tos pačios klasės nariai	To paties paketo klasės	To paties paketo vaikinės klasės nariai	Vaikinės klasės nariai	Visos kitos klasės
<b>private</b>	Taip	Ne	Ne	Ne	Ne
	Taip	Taip	Taip	Ne	Ne
<b>protected</b>	Taip	Taip	Taip	Taip	Ne
<b>public</b>	Taip	Taip	Taip	Taip	Taip

# Modifikatorius „final“

- Jei klasė pažymėta **final**, tokios klasės negali paveldėti jokia kita klasė
- Jei klasės metodas pažymėtas **final**, tokio metodo negalime užkloti/perrašyti

```
final class A {  
    int i = 1;  
  
    String metodos() {  
        return "A";  
    }  
}
```

```
class B extends A {  
    int i = 100;  
  
    @Override  
    String metodos() {  
        return "B";  
    }  
}
```

```
class A {  
    int i = 1;  
  
    final String metodos() {  
        return "A";  
    }  
}
```

```
class B extends A {  
    int i = 100;  
  
    @Override  
    String metodos() {  
        return "B";  
    }  
}
```

# Konstruktoriai vaikinėse klasėse

- Išreikštinai arba neišreikštinai kviečia bazinės klasės konstruktorių.
- **super()** - galimybė nurodyti, kuris bazinės klasės konstruktorius bus kviečiamas. Gali būti tik pirmuoju sakiniu.
- Konstravimo seka (rekursyvus apibrėžimas). Pirma konstruojama objekto tėvo klasės dalis, po to vaiko (vaiko laukų iniciavimas, vaiko konstruktorius).

# Konstruktoriai vaikinėse klasėse

```
class A {
    A () {
        System.out.println("A");
    }
}

class B extends A {
    B () {
        System.out.println("B");
    }
}

class C extends B {
    C () {
        System.out.println("C");
    }
}

new C();

// A
// B
// C
```

Veikia taip lyg kiekvienos klasės konstruktoriuje pirmoji eilutė būtų:  
**super();**

# Cast: primitivieji tipai

- Automatinis tipo konvertavimas
  - `byte` → `short` → `int` → `long` → `float` → `double`
- Išreikštinis tipo konvertavimas
  - `double` → `float` → `long` → `int` → `short` → `byte`

# Automatinis tipo konvertavimas

```
int i = 100;
long l = i; // no explicit type casting required
float f = l; // no explicit type casting required
System.out.println("Int value " + i);
System.out.println("Long value " + l);
System.out.println("Float value " + f);

//Int value 100
//Long value 100
//Float value 100.0

float f = 22.20f;
double d = f; // no explicit type casting required
System.out.println("Float value " + f);
System.out.println("Double value " + d);

//Float value 22.2
//Double value 22.200000762939453
```

# Išreikštinis tipo konvertavimas

```
double d = 100.04;  
long l = (long) d;      //explicit type casting required  
int i = (int) l;        //explicit type casting required
```

```
System.out.println("Double value " + d);  
System.out.println("Long value " + l);  
System.out.println("Int value " + i);
```

```
//Double value 100.04  
//Long value 100  
//Int value 100
```

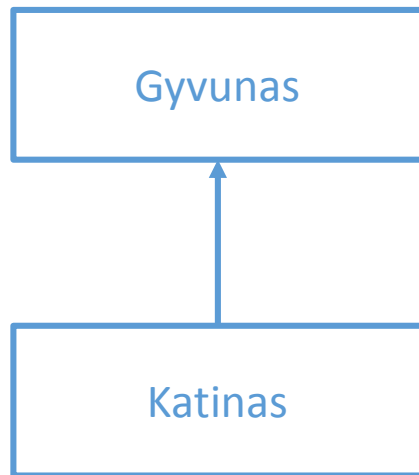
```
double d = 22.99;  
float f = (float) d;    //explicit type casting required
```

```
System.out.println("Double value " + d);  
System.out.println("Float value " + f);
```

```
//Double value 22.99  
//Float value 22.99
```

# Cast: objektai

- Upcasting
- Downcasting



```
class Gyvunas {  
    public void esti() {  
        // ...  
    }  
}  
  
class Katinas extends Gyvunas {  
    @Override  
    public void esti() {  
        // ...  
    }  
    public void miau() {  
        // ...  
    }  
}
```



# Upcasting

```
// Sukuriame katino objekta, kuris paveldi visas gyvuno savybes
Katinas katinas = new Katinas();

// Keiciame tipa i tevini, t.y. Gyvuno tipa.
// cast operacija nereikalinga, nes gyvunas yra abstraktesnis
// katinas turi viska ta pati, ka ir turi gyvunas.
// Java tai suprant ir nereikia isreikstinai nieko nurodyti
Gyvunas gyvunas = katinas;

// Galima naudoto cast operatoriu, kodas kompiliuosis ir veiks,
// taciau upcasting atveju cast operatorius nereikalingas
Gyvunas gyvunas2 = (Gyvunas) katinas;

katinas.miau();
katinas.esti();

gyvunas.esti();
// negalime gyvunui kviesti miau() metodo, nes jis nera katinas pagal tipa
// reikia konvertuoti i katina

gyvunas2.esti();
//negalime gyvunui2 kviesti miau() metodo, nes jis nera katinas pagal tipa
//reikia konvertuoti i katina
```

# Downcasting

```
// Sukuriame katino objekta, kuris paveldi visas gyvuno savybes
Katinas katinas = new Katinas();

// Pakeiciame tipa i gyvuna
Gyvunas gyvunas = katinas;

gyvunas.esti();
// negalime gyvunui kviesti esti() metodo, nes jis nera katinas pagal tipa
// reikia konvertuoti i katina

// konvertuojame gyvuna i katina naudodami cast operatoriu
// tai yra downcasting
Katinas katinas2 = (Katinas) gyvunas;

//dar geriau yra patikrinti, ar gyvunas tikrai gali buti konvertuojamas i
//katina
if (gyvunas instanceof Katinas) {
    katinas2 = (Katinas) gyvunas;
}

// katinui2 jau galime kviesti ir gyvuno, ir katino metodus
katinas2.esti();
katinas2.miau();
```

# Downcasting: trumpas būdas

```
Katinas katinas = new Katinas();
```

```
Gyvunas gyvunas = katinas;
```

```
// Jei mums po cast operacijos reikes iskviesti vos viena katino metoda,  
// tada galime naudoti trumpesni cast uzrasa  
((Katinas) gyvunas).miau();
```

# Downcasting: dar vienas būdas

```
Katinas katinas = new Katinas();
```

```
Gyvunas gyvunas = katinas;
```

```
Katinas katinas2 = Katinas.class.cast(gyvunas);
```

# Downcasting: ko nedaryti

```
// Sukuriame gyvuno objekta, kuris nieko nezino apie katina
// nes katino klase yra hierarchiskai zemiau pagal paveldejima
Gyvunas gyvunas = new Gyvunas();

// jei bandome konvertuoti i Katina, zinoma tai mums nepavyks, nes
// gyvunas nera katinas. Kodas kompiliuojasi, bet paleidus programa
// gauname klaida ClassCastException
Katinas katinas = (Katinas) gyvunas;

// siuo atveju nuo klaidos gelbsti pries konvertavima atliekamas
// patikrinimas, ar gyvunas yra katinas
if (gyvunas instanceof Katinas) {
    katinas = (Katinas) gyvunas;
}
```

```
Exception in thread "main" java.lang.ClassCastException: Gyvunas cannot be cast to Katinas
    at CastTest.main(CastTest.java:13)
```

# Cast operacijos sintaksė

```
Gyvunas gyvunas = new Katinas();
```

```
Katinas katinas = (Katinas) gyvunas;
```

Katinas tipo objektas

Gyvunas tipo objektas

Tipas, į kurį konvertuojame

Turi būti ta pati klasė

# Nuorodos

Upcasting/Downcasting:

<http://www.cs.utexas.edu/~cannata/cs345/Class%20Notes/14%20Java%20Upcasting%20Downcasting.htm>



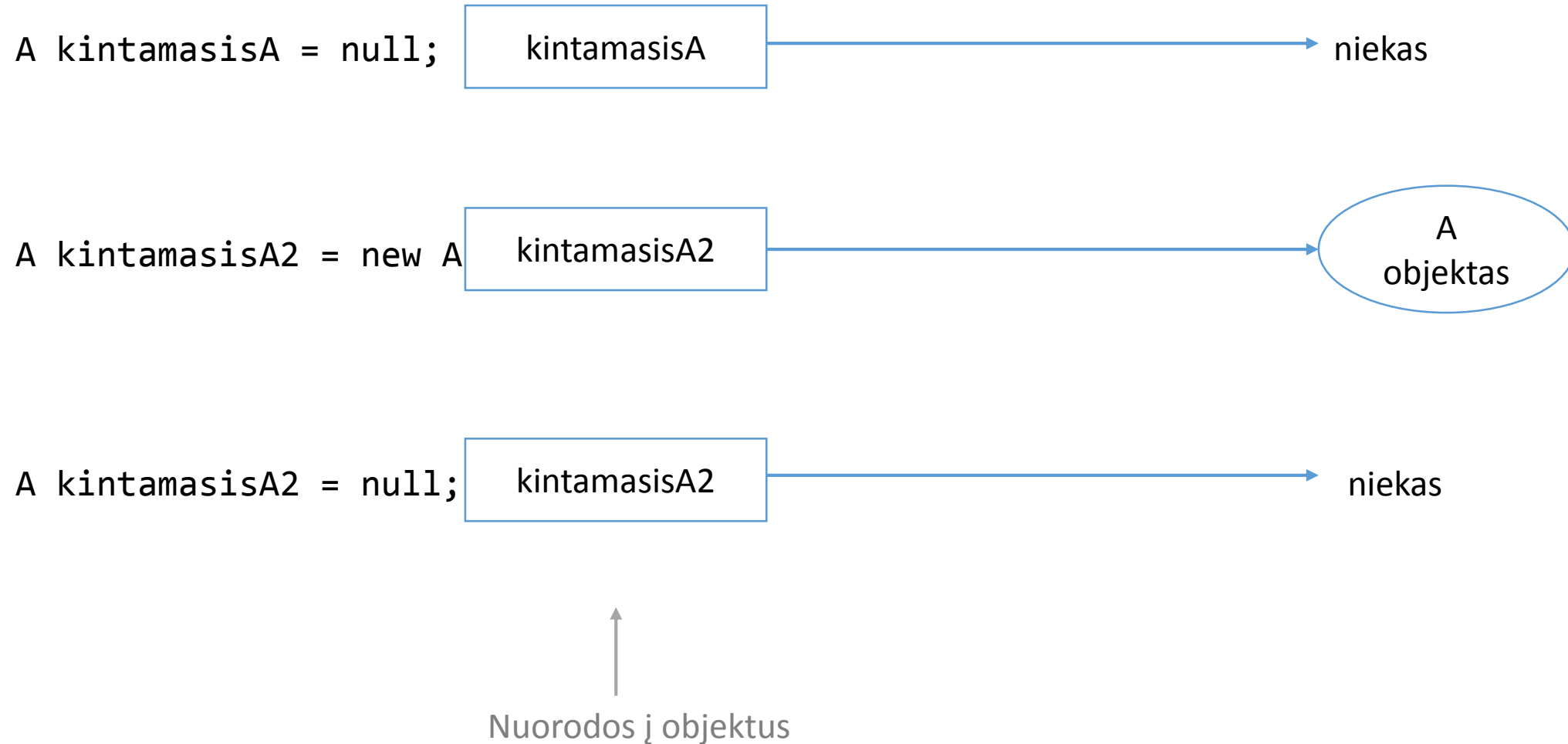
Null



# null

- ***null*** žymi neegzistuojantį objektą
- ***null*** objektas neturi jokių kintamųjų ir metodų, todėl su juo negalime atlikti jokių veiksmų
- Jei bandysime panaudoti neegzistuojantį objektą ***null*** (pvz. kviesime jo metodą), tai gausime ***NullPointerException*** klaidą

# null



# null

```
public class NullTestIncorrect {  
    public static void main(String[] args) throws ArithmeticException {  
        A a = null;  
        a.metodasA();  
    }  
}  
  
class A {  
    public void metodasA() {  
        System.out.println("iskviestas metodas A");  
    }  
}
```

Exception in thread "main" [java.lang.NullPointerException](#)  
at CastTest.main([CastTest.java:8](#))

```
public class NullTestCorrect {  
    public static void main(String[] args) throws ArithmeticException {  
        A a = new A();  
        a.metodasA();  
    }  
}  
  
class A {  
    public void metodasA() {  
        System.out.println("iskviestas metodas A");  
    }  
}
```

// iskviestas metodas A

# Patikrinimas ar ne null

```
public class NullTestCheck {  
    public static void main(String[] args) throws ArithmeticException {  
        A a = null;  
  
        if (a != null) {  
            a.metodasA();  
        }  
  
        a = new A();  
  
        if (a != null) {  
            a.metodasA();  
        }  
    }  
}  
  
class A {  
    public void metodasA() {  
        System.out.println("iskviestas metodas A");  
    }  
}
```

# Nuorodos

Upcasting/Downcasting:

<http://www.cs.utexas.edu/~cannata/cs345/Class%20Notes/14%20Java%20Upcasting%20Downcasting.htm>



# Išimtys (*exceptions*)

# Keletas garsių klaidų



**Therac-25, 1985-87m.**  
Įrenginys apšvitino žmones,  
kelis iš jų mirtinai. Klaida  
pasireiškėdavo operatoriui per  
greitai surinkus tam tikrą  
komandą.

**Ariane 5** skrydis, **1996**, 370 mln. \$  
nuostolis. Sukėlė reikšmės  
perpildymas priskiriant 64 bit. slank.  
kabl. į 16 bit. integer reikšmę



**Betty Heidler –  
Londonas, 2012**  
Programa neužskaitė  
metimo, nes rezultatas  
centimentro tikslumu  
sutapo su ankstesniu



# Problema

- Programa gali "sulūžti" daugeliu atvejų:
  - Netinkamai konvertuojant tipus
  - Reikšmės nebuvimas
  - Pasibaigia atmintis
  - Begalinė rekursija
  - ...



# Galimas sprendimas

- Galime parašyti daug if'ų, kad po kiekvieno žingsnio tikrintų ar yra reikšmė, ar neatliekama dalyba iš nulio, ar tipas konvertuojamas tinkamai ir t.t.
- Bet tokiu atveju kodas paskės klaidų apdorojime ir taps labai sunkiai skaitomas.
- Vis tiek gali likti nenumatytų situacijų
- Klaidų apdorojimą bus sunku modifikuoti

# Java sprendimas

- Naudoti išimtinių situacijų (*exception* - išimtis) mechanizmą
- Programos įprastas kodas bus atskiriamas nuo klaidų apdorojimo kodo
- Java kalbos sakiniai *try-catch-finally, throw*

# Išimtinės situacijos atsiradimas

- Kai vykdoma operacija ar metodo kvietimas, kurio sėkminga baigtis neįmanoma
- Išreikštiniu būdu iššaukiant situaciją Java sakiniu:
  - *throw new Exception();*

# Išimties vystymasis

- Programos normalus vykdymas nutraukiamas ir sukuriama išimtį apibūdinantis objektas (*exception*)
- Jei nesiimama priemonių, programos darbas užbaigiamas išvedant diagnostinį pranešimą

# Kaip sukelti išimtį

```
int z = 5 / 0;
```



```
Exception in thread "main" java.lang.ArithmeticException: / by  
zero  
    at Programa.main(Programa.java:7)
```

# Kaip sukelti išimtį

```
throw new Exception("labai didele klaida");
```



```
Exception in thread "main" java.lang.Exception: labai didele  
klaida  
    at Programa.main(Programa.java:7)
```

# try-catch-finally

```
try {  
    ...  
} catch (...) {  
    ...  
} finally {  
    ...  
}
```

Galimai nesaugus kodas

Klaidos tipas

Veiksmi, jei įvyko klaida

Veiksmi atliekami tiek klaidos atveju, tiek ir jei klaidos nėra

```
Scanner sc = new Scanner(System.in);  
int i = sc.nextInt();  
try {  
    int z = 5 / i;  
    System.out.println("z: " + z);  
} catch (ArithmeticException e) {  
    System.out.println("Dalyba is nulio");  
} finally {  
    sc.close();  
    System.out.println("scanneris uzdarytas");  
}
```

Input: 5 → z: 1  
scanneris uzdarytas

Input: 0 → Dalyba is nulio  
scanneris uzdarytas

# Kelių tipų klaidų apdorojimas

```
try {  
    ...  
} catch (...) {  
    ...  
} catch (...) {  
    ...  
} catch (...) {  
    ...  
} finally {  
    ...  
}
```

```
try {  
    int z = 5 / 0;  
    String tekstas = null;  
    tekstas.indexOf('A');  
} catch (NullPointerException e) {  
    System.out.println("Nera reikšmes");  
} catch (ArithmeticException e) {  
    System.out.println("Dalyba is nulio");  
} catch (Exception e) {  
    System.out.println("Nezinoma klaida");  
}
```

Skirtingų tipų klaidos.  
Anksčiau turi būti gaudoma  
vaikinė klasė, vėliau tėvinė.



# Kelių tipų klaidų apdorojimas

```
try {  
    ...  
} catch (... | ... | ...) {  
    ...  
} finally {  
    ...  
}
```

```
try {  
    int z = 5 / 0;  
    String tekstas = null;  
    tekstas.indexOf('A');  
} catch (NumberFormatException | NullPointerException e) {  
    System.out.println("klaida");  
}
```

# Vidinis try-catch-finally blokas

```
try {  
    ...  
    try {  
        ...  
    } catch (...) {  
        ...  
    }  
    ...  
} catch (...) {  
    ...  
}
```

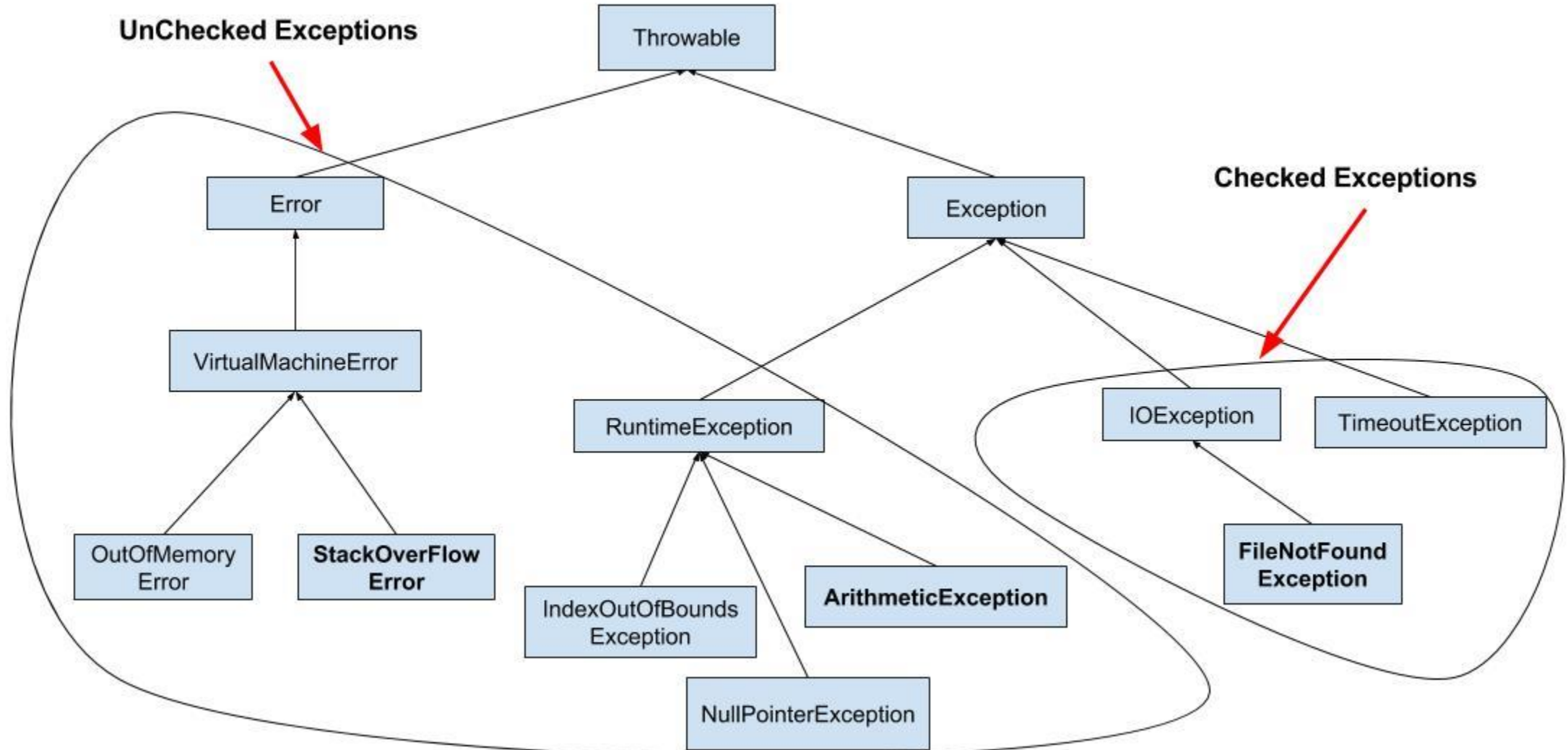
```
try {  
    try {  
        int z = 5 / 0;  
    } catch (ArithmeticException e) {  
        System.out.println("Dalyba is nulio");  
    }  
} catch (Exception e) {  
    System.out.println("Nezinoma klaida");  
}
```

# Išimties objektas

- Saugo informaciją apie klaidą
- Paveldi klasę ***java.lang.Throwable***
- Turi konstruktorius, metodus išgauti klaidos vietą, pranešimo tekstą, priežastį ir pan.
- Išvestinės klasės specializuoja klaidas ir gali suteikti papildomą informaciją
- Patiems iššaukiant situacijas naudinga naudoti savo apibrėžtas išimčių klases

# Svarbiausios išimčių klasės

- **Throwable** – išimčių superklasė
- **Error** – išreiškia paprastai neapdorojamas klaidas
- **Exception** – klaidos, kurios privalo būti deklaruojamos ir apdorojamos (checked)
- **RuntimeException** – kompiliatoriaus nekontroliuojamos klaidos – gali įvykti bet kada. Apdorojamos esant reikalui



# Catch blokų tvarka

- Nusako išimčių tinkamumo tikrinimo eiliškumą
- Blokas pagauna išimtį, jei išimties objektas gali būti priskirtas catch dalyje nurodytam tipui
- Kompiliatorius traktuoja kaip klaidingą **catch** blokų išdėstyme, jei superklasės klaida gaudoma anksčiau, nei subklasės

# Veiksmai apdorojant pagautą išimtį

- Naivusis – išvesti diagnostinį pranešimą ir nutraukti programos vykdymą
- Optimistinis – ištaisyti situaciją ir toliau testuoti programą
- Realistinis – jei klaida nepataisoma, catch bloke sakiniu throw pakartotinai iššaukti išimtį, naudojant tą patį situacijos objektą arba sukurti naują, tikintis apdorojimo kitame kodo lygmenyje

# Išimties iššaukimas

```
public static void main(String[] args) throws ArithmeticException {  
    int i = 0;  
    try {  
        int z = 5 / 0;  
    } catch (ArithmeticException e) {  
        System.out.println("Dalyba is nulio, pataisyti nepavyks");  
        throw e;  
    }  
}
```



# Savo apsibrėžta išimties klasė

- Reikalingas, kai norime specializuoti išimtį, kad klientinė dalis gautų išskirti ją iš kitų situacijų ir atitinkamai reaguotų
- Kai norime papildyti *Exception* klasę specifine informacija

# Apsibrėžiame išimties klasę

```
class AccountException extends Exception {  
    private int balance = 0;  
  
    public AccountException() {  
    }  
  
    public AccountException(String message) {  
        super(message);  
    }  
  
    public AccountException(String message, int balance) {  
        super(message);  
        this.balance = balance;  
    }  
  
    public int getBalance() {  
        return balance;  
    }  
}
```

# Panaudojame išimties klasę kode

```
class Account {  
    private int balance = 100;  
  
    public void withdraw(int amount) throws AccountException {  
        if (balance - amount >= 0) {  
            balance -= amount;  
        } else {  
            throw new AccountException("Nepakanka pinigų", balance - amount);  
        }  
    }  
}
```

# Panaudojame išimties klasę kode

```
public static void main(String[] args) {  
    Account a = new Account();  
    try {  
        a.withdraw(90);  
        System.out.println("Pinigai nuskaityti");  
        a.withdraw(50);  
        System.out.println("Pinigai nuskaityti");  
    } catch (AccountException e) {  
        System.out.println("Klaida. Po nuskaitymo balansas butu: " + e.getBalance());  
    }  
}
```

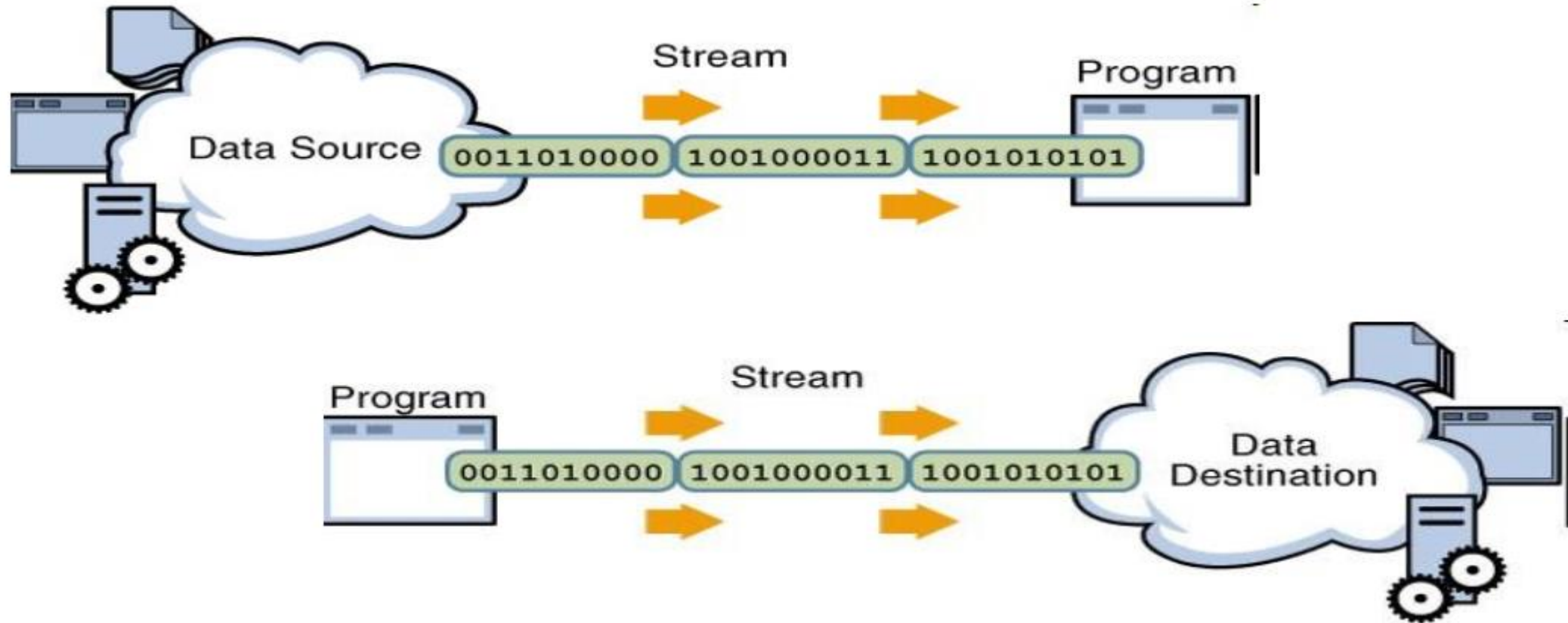
Pinigai nuskaityti

Klaida. Po nuskaitymo balansas butu: -40



# Darbas su failais

# Srauto abstrakcija



# Java srautas

- Suprantamas kaip duomenų (baitų, simbolių) seka.
- Įgalina operuoti srauto skaitymo / rašymo operacijomis nepriklausomai nuo konkretaus įrenginio.
- Kiekvieną konkretaus tipo srautą atitinka konkrečios Java klasės objektas.
- **java.io** – pagrindinės srautų klasės.
- java klasės paruoštos plėtimui.

# Srautų tipai

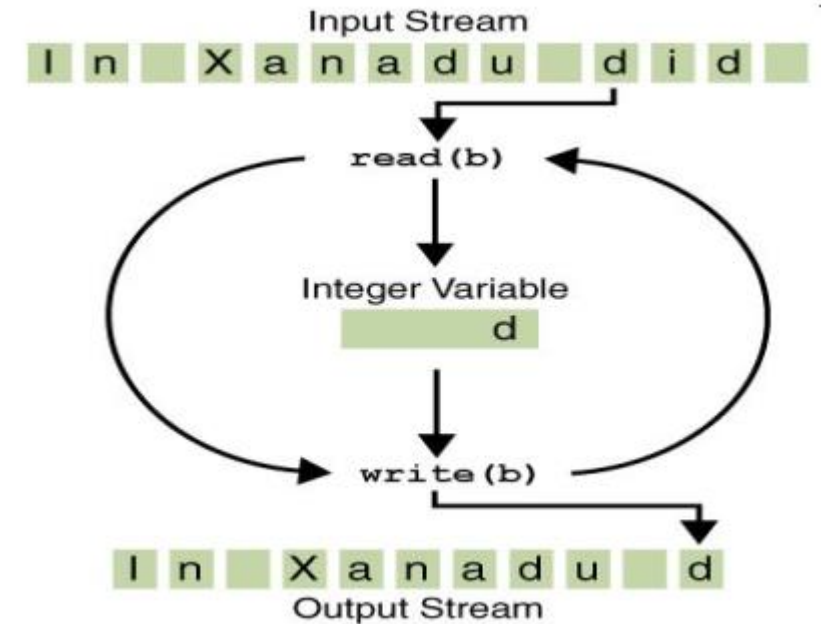
- Baitų / simbolių srautai
- Skaitymo / rašymo srautai
- 4 bazinės klasės:
  - *InputStream, OutputStream* – operuoja baitais (byte tipas)
  - *Reader, Writer* – operuoja simboliais (char tipas), leidžia išvengti simbolių kodavimo / dekodavimo problemų
  - Galima operuoti simboliais virš baitų srauto



# Baitų kopijavimo pavyzdys

```
import java.io.*;

public class CopyBytesPP {
    public static void main(String[] args) throws IOException {
        InputStream in = new FileInputStream("orig.txt");
        OutputStream out = new FileOutputStream("copy.txt");
        for (int c; (c = in.read()) != -1;) {
            out.write(c);
        }
        in.close();
        out.close();
    }
}
```



# Specializuosios išvedimo klasės

- Įgalina atlikti specializuotas išvestis, parūpina papildomą funkcionalumą
- `ByteArrayOutputStream`, `FileOutputStream`, `FilterOutputStream`, `ObjectOutputStream`, `PipedOutputStream`
- `BufferedWriter`, `CharArrayWriter`, `FilterWriter`, `OutputStreamWriter`, `PipedWriter`, `PrintWriter`, `String`

# Specializuotos įvedimo klasės

- Įgalina skaityti iš specializuotų šaltinių, parūpina papildomą funkcionalumą
- ByteArrayInputStream, FileInputStream, FilterInputStream, InputStream, ObjectInputStream, PipedInputStream, SequenceInputStream, StringBufferInputStream
- BufferedReader, CharArrayReader, FilterReader, InputStreamReader, PipedReader, StringReader

# Objekto serializacija: įrašymas

```
import java.io.*;

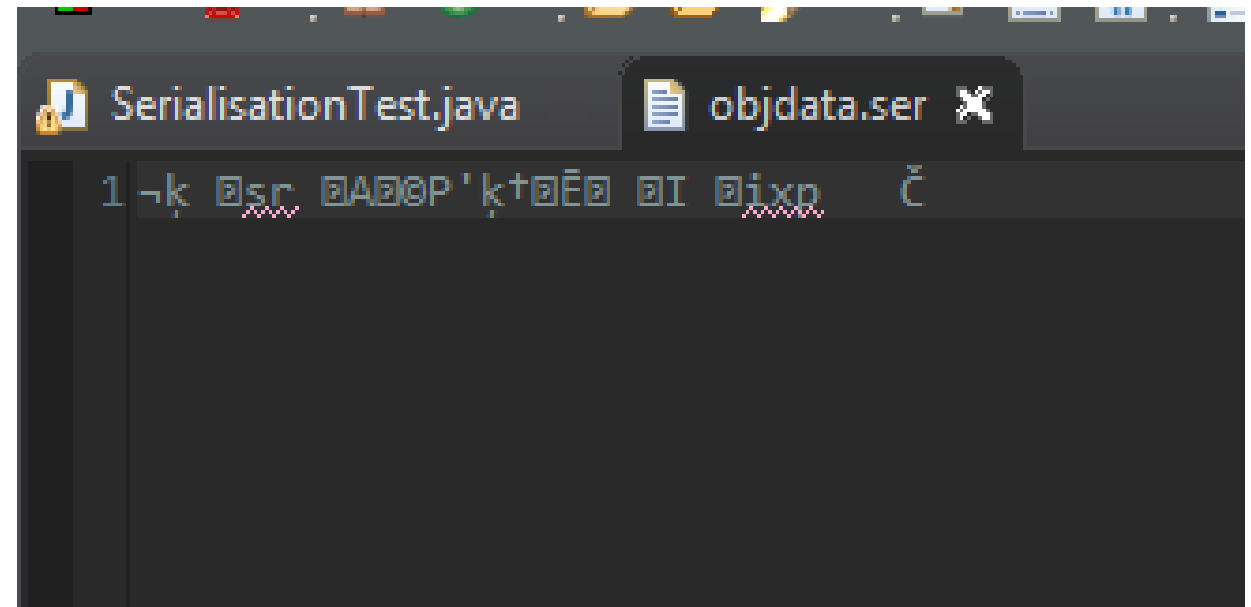
public class SerialisationTest {
    public static void main(String[] args) throws IOException {
        FileOutputStream out = new FileOutputStream("objdata.ser");
        ObjectOutputStream s = new ObjectOutputStream(out);
        s.writeObject(new A(200));
        s.flush();
        s.close();
    }
}

class A implements Serializable {
    private int i;

    public A(int i) {
        this.i = i;
    }

    public int getI() {
        return i;
    }

    public void setI(int i) {
        this.i = i;
    }
}
```



# Objekto serializacija: nuskaitymas

```
import java.io.*;

public class SerialisationTest {
    public static void main(String[] args) throws IOException, ClassNotFoundException {
        FileInputStream in = new FileInputStream("objdata.ser");
        ObjectInputStream s = new ObjectInputStream(in);
        A a = (A) s.readObject();
        System.out.println(a.getI());
        s.close();
    }
}

class A implements Serializable {
    private int i;

    public A(int i) {
        this.i = i;
    }

    public int getI() {
        return i;
    }

    public void setI(int i) {
        this.i = i;
    }
}
```

200

# Serializacijos pritaikymas

- Serializuojamas objektas turi implementuoti **Serializable** interfeisą
- Daugelis standartinių klasių serializuojamos, bet kai kurios - ne (pvz., **Thread**)
- Klasės neserializuojamus laukus galima pažymėti raktažodžiu **transient**
- Klasė gali papildyti standartinius serializacijos veiksmus parūpindama metodus **writeObject**, **readObject**

# Kitos klasės

- **java.io.File** – “apgaubia” failo vardą. Turi failų atributų tikrinimo, sukūrimo / pašalinimo, katalogų peržiūros operacijas.
- **java.io.RandomAccessFile** – tiesioginės skaitymo / rašymo kreipties klasė. Leidžia skaitymo / rašymo / pozicionavimo operacijas.

## InputStream Methods

---

- Reading
  - read() methods will block until data is available to be read
  - two of the three read() methods return the number of bytes read
    - -1 is returned if the Stream has ended
  - throws IOException if an I/O error occurs. This is a checked exception

. There are 3 main read methods:

```
int read()
```

- Reads a single character. Returns it as integer

```
int read(byte[] buffer)
```

- Reads bytes and places them into buffer (max = size of buffer)
- returns the number of bytes read

```
int read(byte[] buffer, int offset, int length)
```

- Reads up to length bytes and places them into buffer
- First byte read is stored in buffer[offset]
- returns the number of bytes read



## InputStream Methods

---

- `available()` method returns the number of bytes which can be read without blocking
- `skip()` method skips over a number of bytes in the input stream
- `close()` method will close the input stream and release any system resources
- input streams optionally support repositioning the stream
  - can mark the stream at a certain point and 'rewind' the stream to that point later.
- methods that support repositioning are:
  - `markSupported()` returns true if repositioning is supported
  - `mark()` places a mark in the stream
  - `reset()` 'rewinds' the stream to a previously set mark

## Creating an InputStream

---

- InputStream is an abstract class
  - Programmers can only instantiate subclasses.
- ByteArrayInputStream:
  - Constructor is provided with a byte array.
  - This byte array contains all the bytes provided by this stream
  - Useful if the programmer wishes to provide access to a byte array using the stream interface.
- FileInputStream:
  - Constructor takes a filename, File object or FileDescriptor Object.
  - Opens a stream to a file.
- FilterInputStream:
  - Provides a basis for filtered input streams
  - Filtered streams are covered later in the chapter.

## OutputStream Methods

---

- Writing:
  - `write()` methods write data to the stream. Written data is buffered.
  - Use `flush()` to flush any buffered data from the stream.
  - throws `IOException` if an I/O error occurs. This is a checked exception
- There are 3 main write methods:

```
void write(int data)
```

- Writes a single character
- Note: even though data is an integer, data must be set such that:
  - $0 \leq \text{data} \leq 255$

```
void write(byte[] buffer)
```

- Writes all the bytes contained in buffer to the stream

```
void write(byte[] buffer, int offset, int length)
```

- Writes length bytes to stream starting from `buffer[offset]`

## OutputStream Methods

---

- `flush()`
  - To improve performance, almost all output protocols buffer output.
  - Data written to a stream is not actually sent until buffering thresholds are met.
  - Invoking `flush()` causes the `OutputStream` to clear its internal buffers.
- `close()`
  - Closes stream and releases any system resources.

## Creating an OutputStream

---

- OutputStream is an abstract class.
  - Programmers instantiate one of its subclasses
- ByteArrayOutputStream:
  - Any bytes written to this stream will be stored in a byte array
  - The resulting byte array can be retrieved using toByteArray() method.
- FileOutputStream:
  - Constructor takes a filename, File object, or FileDescriptor object.
  - Any bytes written to this stream will be written to the underlying file.
  - Has one constructor which allows for appending to file:  
`FileOutputStream(String filename, boolean append)`
- FilterOutputStream:
  - Provides a basis for Output Filter Streams.
  - Will be covered later in chapter.

## Creating an OutputStream

---

- **ObjectOutputStream**
  - Created from another output stream (such as `FileOutputStream`)
  - Programmers serialize objects to the stream using the `writeObject()` method
  - More on Serialization later in the Chapter.
- **PipedOutputStream:**
  - Connects to an Instance of `PipedInputStream`
  - A pipe represents a one-way stream through which 2 threads may communicate
    - Thread1 writes to a `PipedOutputStream`
    - Thread2 reads from the `PipedInputStream`

# Skaitymas: *FileInputStream*

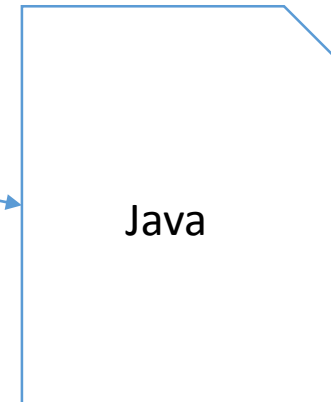
```
InputStream in = null;
try {
    in = new FileInputStream("src/failas.txt");
    int raidesKodas;
    while ((raidesKodas = in.read()) > 0) {
        char raide = (char) raidesKodas;
        System.out.print(raide);
    }
} catch (Exception e) {
    e.printStackTrace();
} finally {
    if (in != null)
        in.close();
}
```

A blue arrow points from the line of code "System.out.print(raide);", specifically from the word "print", to the word "Java".

Java

# Rašymas: *FileOutputStream*

```
char[] tekstas = { 'J', 'a', 'v', 'a' };
OutputStream os = null;
try {
    os = new FileOutputStream("src/failas.txt");
    for (int i = 0; i < tekstas.length; i++) {
        os.write(tekstas[i]);
    }
    os.flush();
} catch (IOException e) {
    e.printStackTrace();
} finally {
    if (os != null)
        os.close();
}
```





# Rašymas: *FileWriter* + *BufferedWriter*

```
import java.io.BufferedWriter;
import java.io.FileWriter;
import java.io.IOException;

public class WriteToFile {
    private static final String FILENAME = "src/failas.txt";

    public static void main(String[] args) {
        try (BufferedWriter bw = new BufferedWriter(new FileWriter(FILENAME))) {
            String content = "This is the content to write into file\n";
            bw.write(content);
            System.out.println("Done");
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

# Rašymas: *FileWriter* + *BufferedWriter*

```
import java.io.BufferedWriter;
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;

public class WriteToFile {
    public static void main(String[] args) {
        try {
            String content = "This is the content to write into file";
            File file = new File("src/failas.txt");
            // if file doesnt exists, then create it
            if (!file.exists()) {
                file.createNewFile();
            }
            FileWriter fw = new FileWriter(file.getAbsolutePath());
            BufferedWriter bw = new BufferedWriter(fw);
            bw.write(content);
            bw.close();
            System.out.println("Done");
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

# Skaitymas: *File*

```
import java.io.BufferedReader;
import java.io.File;
import java.io.FileInputStream;
import java.io.IOException;
import java.io.InputStreamReader;

public class FileTest {

    public static void main(String[] args) throws IOException {

        File failas = new File("src/failas.txt");

        readFile1(failas);

    }

    private static void readFile1(File fin) throws IOException {
        FileInputStream fis = new FileInputStream(fin);
        // Construct BufferedReader from InputStreamReader
        BufferedReader br = new BufferedReader(new InputStreamReader(fis));
        String line = null;
        while ((line = br.readLine()) != null) {
            System.out.println(line);
        }
        br.close();
    }

}
```

# Rašymas: *PrintWriter*

```
import java.io.BufferedWriter;
import java.io.FileWriter;
import java.io.IOException;
import java.io.PrintWriter;

public class FileTest {

    public static void main(String[] args) {
        try {
            FileWriter fw = new FileWriter("src/failas.txt", true);
            BufferedWriter bw = new BufferedWriter(fw);
            PrintWriter out = new PrintWriter(bw);
            out.println("the text");
            out.close();
        } catch (IOException e) {
            // ...
        }
    }
}
```

# Nuorodos

- <http://www.slideshare.net/martyhall/file-io-in-java-8-applying-the-power-of-streams>
- <http://howtodoinjava.com/java-8/read-file-line-by-line-in-java-8-streams-of-lines-example/>
- <http://www.mkyong.com/java/how-to-write-to-file-in-java-bufferedwriter-example/>
- <https://www.mkyong.com/java/how-to-write-to-file-in-java-fileoutputstream-example/>
- <https://examples.javacodegeeks.com/core-java/writeread-csv-files-in-java-example/>
- <https://www.mkyong.com/java/how-to-export-data-to-csv-file-java/>



# Vartotojo sąsaja: Swing

# Vartotojo sąsaja

WindowTitle

MenuWidget1 MenuWidget2

ToolBarButton ☒ ToolBarCheckBox

PanelCaption

Panel

Item 1 ☒ RadioButton1

Item 2 ☐ RadioButton2

Item 3 ☐ RadioButton3

Item 4 ☐ InactiveRadio

Item 5

Button

SelectedTab OtherTab

☐ UncheckedCheckBox

☒ CheckedCheckBox

☐ InactiveCheckBox

Forms Company Misc Options Help

Home Master Data - 97

Company Details

Capital Details

Director Details

Shareholder Details

Charge Details

Registration of Resolutions

Interested Contracts

Financial Details

Minutes Details

Misc. Registers

Master Data Print

TextField

TextArea

Item 1

Firstna...	Lastname	Company	Street	City	Country	E-Mail	Status	Limits Code
Mike	Beck	ACME Corp	Elizabeth Street 124	Chandler	United States	mike.beck@acmecorp.com	Active and Used	DDEGEDE
Jeff	Fowler	ACME Limited	Hanover Court 124	St. Louis	United Kingdom	jeff.fowler@acmelimited.com	Inactive or Deleted	
Amy	Levenson	ACME & Sons	8th Street 126	Fort Worth	Sweden	amy.levenson@acme&sons.com	Active and Standby	DFBBDDEB
Mark	Smith	ACME Brothers	Sycamore Lane 137	Stockton	Australia	mark.smith@acmebrothers.com	Active Spare	DCBEABEA
Brian	Muller	ACME Industry	Cardinal Drive 83	Montgomery	Canada	brian.muller@acmeindustry.com	Active and Used	
Lara	Wilson	ACME Corp	Jefferson Court 50	Chandler	United States	lara.wilson@acmecorp.com	Inactive or Deleted	EGDFECCA
Roger	Tucker	ACME Limited	Elizabeth Street 140	St. Louis	United Kingdom	roger.tucker@acmelimited.com	Active and Standby	BFBFAEGA
Mike	Beck	ACME & Sons	Hanover Court 69	Fort Worth	Sweden	mike.beck@acme&sons.com	Active Spare	
Jeff	Fowler	ACME Brothers	8th Street 87	Stockton	Australia	jeff.fowler@acmebrothers.com	Active and Used	AGAGHCGE
Amy	Levenson	ACME Industry	Sycamore Lane 91	Montgomery	Canada	amy.levenson@acmeindustry.com	Inactive or Deleted	DAEGEFEE
Mark	Smith	ACME Corp	Cardinal Drive 118	Chandler	United States	mark.smith@acmecorp.com	Active and Standby	
Brian	Muller	ACME Limited	Jefferson Court 130	St. Louis	United Kingdom	brian.muller@acmelimited.com	Active Spare	HEEGCGAB
Lara	Wilson	ACME & Sons	Elizabeth Street 123	Fort Worth	Sweden	lara.wilson@acme&sons.com	Active and Used	GAFGEBAE
Roger	Tucker	ACME Brothers	Hanover Court 122	Stockton	Australia	roger.tucker@acmebrothers.com	Inactive or Deleted	
Mike	Beck	ACME Industry	8th Street 80	Montgomery	Canada	mike.beck@acmeindustry.com	Active and Standby	BFCFGFGE
Jeff	Fowler	ACME Corp	Sycamore Lane 103	Chandler	United States	jeff.fowler@acmecorp.com	Active Spare	BAGFEBCB
Amy	Levenson	ACME Limited	Cardinal Drive 108	St. Louis	United Kingdom	amy.levenson@acmelimited.com	Active and Used	
Mark	Smith	ACME & Sons	Jefferson Court 54	Fort Worth	Sweden	mark.smith@acme&sons.com	Inactive or Deleted	EDBGABDC
Brian	Muller	ACME Brothers	Elizabeth Street 61	Stockton	Australia	brian.muller@acmebrothers.com	Active and Standby	ACFEBAEB
Lara	Wilson	ACME Industry	Hanover Court 98	Montgomery	Canada	lara.wilson@acmeindustry.com	Active Spare	
Roger	Tucker	ACME Corp	8th Street 129	Chandler	United States	roger.tucker@acmecorp.com	Active and Used	EGEFCEEB

Licensee Details

First Name Mike

Last Name Beck

Company ACME Corp

Street Elizabeth Street 124

City Chandler

Country United States

E-Mail mike.beck@acmecorp.com

Users

☐ No User

☐ 1 User

☒ 2-4 Users

☐ 5-9 Users

☐ 10-19 Users

☐ 20-49 Users

☐ 50-99 Users

☐ 100-499 Users

☐ Unlimited

Installations

☐ Single

☐ Team

☒ Workgroup

☐ Division

☐ Company

☐ Country

☐ Unlimited

Bandwidth

☐ Offline

☐ 64 KBit

☐ 1 MBit

☒ 16 MBit

☐ 100 MBit

☐ 1 GBit

☐ Unlimited

Duration

☐ 1 Hour

☐ 8 Hours

☐ 1 Day

☐ 1 Week

☐ 1 Month

☐ 1 Year

☒ Unlimited

Storage

☐ None

☐ 128 KBytes

☐ 1 MByte

☒ 256 MBytes

☐ 1 GBytes

☐ 4 GBytes

☐ Unlimited

Transfers

☐ None

☐ 1 or 2

☐ 3-9

☐ 10-99

☐ 100-299

☐ 300-999

☐ Unlimited

Invocations

☐ 0-100

☐ 100-1K

☐ 1K-10K

☒ 10K-100K

☐ 100K-1M

☐ 1M-10M

☐ 10M-1G

☐ Unlimited

CPU Cores

☐ 1 Core

☐ 2 Cores

☐ 3-4 Cores

☒ 5-8 Cores

☐ Unlimited

Forms Company Misc Options Help

Home Master Data - 97

Company Details

Capital Details

Director Details

Shareholder Details

Charge Details

Registration of Resolutions

Interested Contracts

Financial Details

Minutes Details

Misc. Registers

Master Data Print

Director/Manager/Secretary Details

DIN or PAN	Date of Event	Nature of Event	Designation	Name	Address	Date of Birth	Gender
00286316	27/03/2003	Appointment	Director	RAMASUBRAMANIAN KUMBAKONAM SE...	1910, 8TH A CROSS, H.A.L ...	12/06/1952	Male
00286316	09/03/2012	Cessation	Director	RAMASUBRAMANIAN KUMBAKONAM SE...	1910, 8TH A CROSS, H.A.L ...	12/06/1952	Male
02008267	27/03/2003	Appointment	Director	CHENNABASAVESWAR GADIGEPPA KAUJ...	NO.201, LANGFORD HOU...	05/08/1967	Male
02008267	17/06/2010	Cessation	Director	CHENNABASAVESWAR GADIGEPPA KAUJ...	NO.201, LANGFORD HOU...	05/08/1967	Male
02742162	08/03/2012	Appointment	Director/Chairman/Executive ...	NARAYANI RAMASUBRAMANIAN	1910, 8TH A CROSS (B.D.A...	20/12/1959	
02905730	17/06/2010	Appointment	Director/Executive Director	ASHWINI SETHU RAMAN	1910, 8TH A CROSS H. A. L...	14/09/1982	Female
02906011	17/06/2010	Appointment	Director	SETHURAMAN RAMASUBRAMANIAN	1910, 8TH A CROSS H.A.L 3...	18/07/1982	Male
02906011	09/03/2012	Cessation	Director	SETHURAMAN RAMASUBRAMANIAN	1910, 8TH A CROSS H.A.L 3...	18/07/1982	Male

Show Only Current Directors Open Director Register Generate Director Register Save

Digital Signature Certificate Details

DIN/PAN/ Membership No	Name	DSC Registered	DSC Expiry Date	Default Signatory	Resolution No to sign E-form	Date of passing Resolution
02742162	NARAYANI RAMASUBRAMANIAN	No		No		
02905730	ASHWINI SETHU RAMAN	Yes	14/03/2014	Yes	3	17/06/2010

# Tuščias rėmelis JFrame

```
import javax.swing.JFrame;

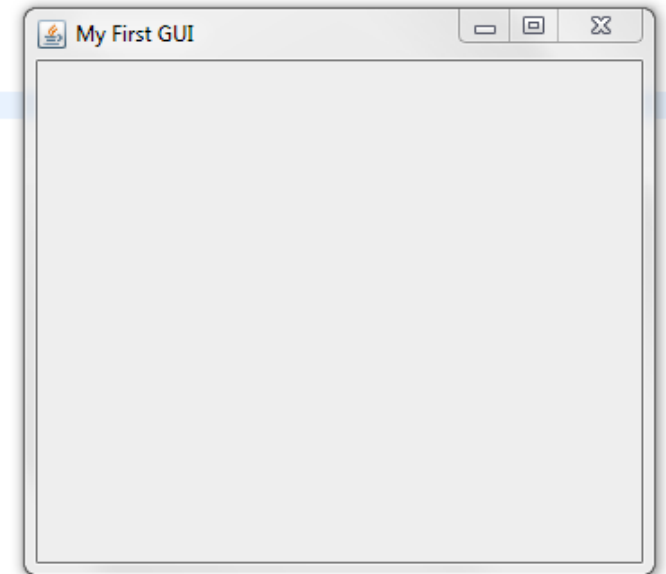
public class SimpleGUI {

    public static void main(String[] args) {

        JFrame frame = new JFrame("My First GUI"); // remelio objektas
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(600, 600); // remelio dydis pikseliais
        frame.setVisible(true);

    }

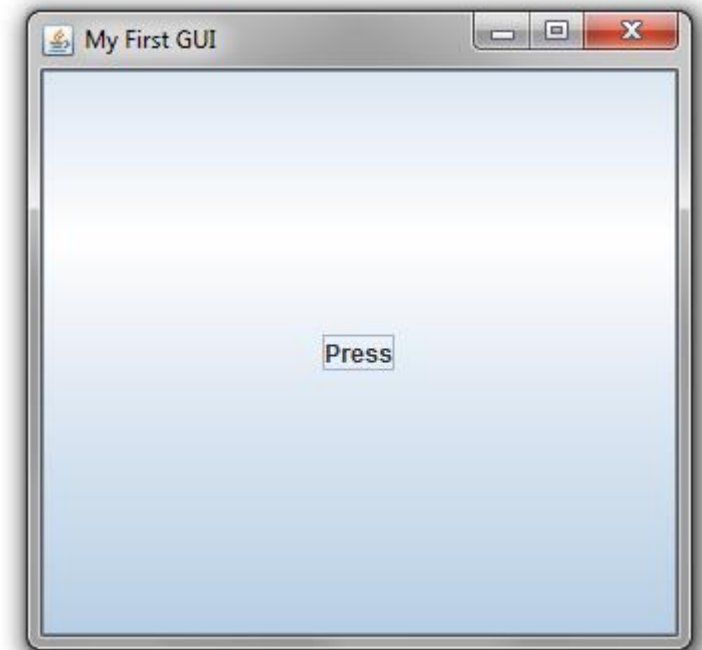
}
```





# Mygtukas rėmelyje JButton

```
import javax.swing.JButton;  
import javax.swing.JFrame;  
  
public class SimpleGUI {  
  
    public static void main(String[] args) {  
  
        JFrame frame = new JFrame("My First GUI");  
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);  
        frame.setSize(600, 600);  
  
        JButton button = new JButton("Press"); // mygtuko objektas  
        frame.getContentPane().add(button); // i remeli idedamas mygtukas  
  
        frame.setVisible(true);  
    }  
}
```



# Dar vienas mygtukas rėmelyje

```
import javax.swing.JButton;  
import javax.swing.JFrame;  
  
public class SimpleGUI {  
  
    public static void main(String[] args) {  
  
        JFrame frame = new JFrame("My First GUI");  
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);  
        frame.setSize(600, 600);  
  
        JButton button1 = new JButton("Mygtukas 1");  
        JButton button2 = new JButton("Mygtukas 2");  
        frame.getContentPane().add(button1);  
        frame.getContentPane().add(button2);  
  
        frame.setVisible(true);  
    }  
}
```

Bet matomas tik vienas mygtukas...



# Išdėstymas BorderLayout

```
import java.awt.BorderLayout;
import java.awt.Color;

import javax.swing.JFrame;
import javax.swing.JPanel;

public class SimpleGUI {

    public static void main(String[] args) {

        JFrame frame = new JFrame("My First GUI");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(600, 600);

        JPanel apatinisPanel = new JPanel(); // sukuriame panel
        apatinisPanel.setBackground(Color.GRAY); // suteikiame panel spalva
        // idedame panel i remeli
        frame.getContentPane().add(BorderLayout.SOUTH, apatinisPanel);

        JPanel virsutinisPanel = new JPanel();
        virsutinisPanel.setBackground(Color.GRAY);
        frame.getContentPane().add(BorderLayout.NORTH, virsutinisPanel);

        JPanel kairysisPanel = new JPanel();
        kairysisPanel.setBackground(Color.BLUE);
        frame.getContentPane().add(BorderLayout.WEST, kairysisPanel);

        JPanel desinysisPanel = new JPanel();
        desinysisPanel.setBackground(Color.BLUE);
        frame.getContentPane().add(BorderLayout.EAST, desinysisPanel);

        JPanel centrinisPanel = new JPanel();
        centrinisPanel.setBackground(Color.ORANGE);
        frame.getContentPane().add(BorderLayout.CENTER, centrinisPanel);

        frame.setVisible(true);
    }
}
```

<https://docs.oracle.com/javase/tutorial/uiswing/layout/border.html>



# Mygtukai panelyje su BorderLayout

```
import java.awt.BorderLayout;
import java.awt.Color;

import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JPanel;

public class SimpleGUI {

    public static void main(String[] args) {

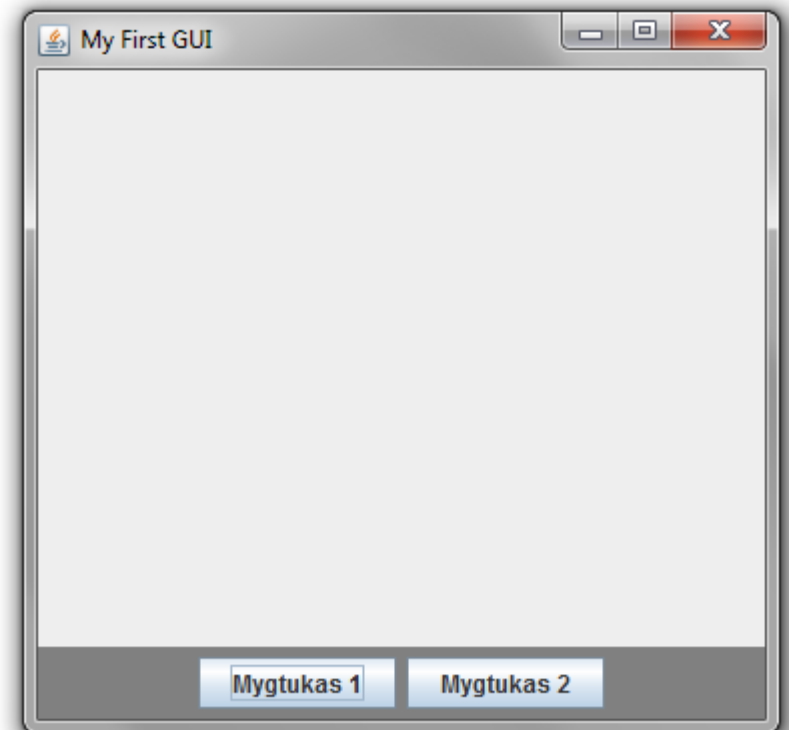
        JFrame frame = new JFrame("My First GUI");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(600, 600);

        JPanel apatinisPanel = new JPanel();
        apatinisPanel.setBackground(Color.GRAY);
        frame.getContentPane().add(BorderLayout.SOUTH, apatinisPanel);

        JButton mygtukas1 = new JButton("Mygtukas 1");
        JButton mygtukas2 = new JButton("Mygtukas 2");

        apatinisPanel.add(mygtukas1);
        apatinisPanel.add(mygtukas2);

        frame.setVisible(true);
    }
}
```



# Išdėstymas FlowLayout

```
import java.awt.Color;

import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JPanel;

public class SimpleGUI {

    public static void main(String[] args) {

        JFrame frame = new JFrame("My First GUI");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(600, 600);

        JPanel panel = new JPanel();
        panel.setLayout(new FlowLayout());
        panel.setBackground(Color.GRAY);
        frame.getContentPane().add(panel);

        JButton mygtukas1 = new JButton("Mygtukas 1");
        JButton mygtukas2 = new JButton("Mygtukas 2");

        panel.add(mygtukas1);
        panel.add(mygtukas2);

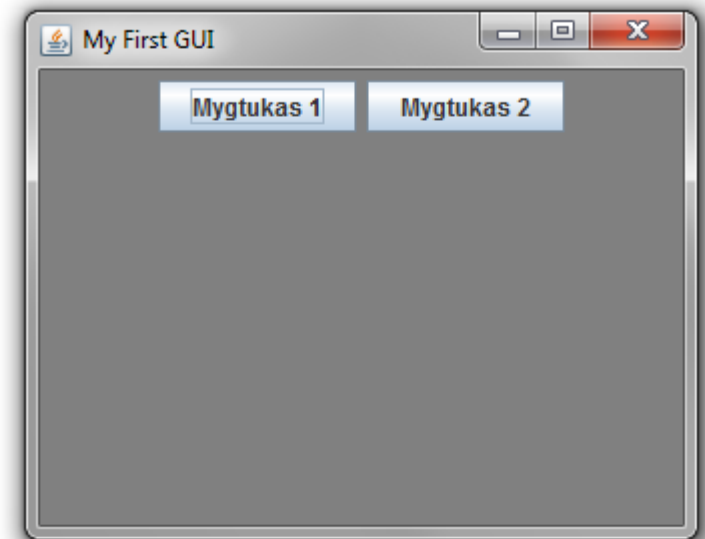
        frame.setVisible(true);

    }
}
```

<https://docs.oracle.com/javase/tutorial/uiswing/layout/flow.html>

Kiti išdėstymai:

<https://docs.oracle.com/javase/tutorial/uiswing/layout/>



# Pasirinkimai JCheckBox

```
import java.awt.GridLayout;

import javax.swing.JCheckBox;
import javax.swing.JFrame;
import javax.swing.JPanel;

public class SimpleGUI {

    public static void main(String[] args) throws Exception {

        JFrame frame = new JFrame("My First GUI");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(600, 600);

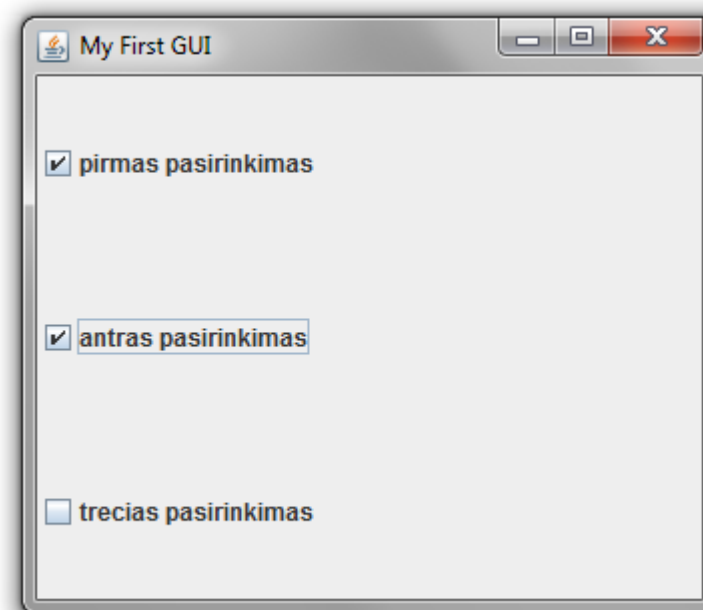
        JCheckBox checkBox1 = new JCheckBox("pirmas pasirinkimas");
        JCheckBox checkBox2 = new JCheckBox("antras pasirinkimas");
        JCheckBox checkBox3 = new JCheckBox("trecias pasirinkimas");

        JPanel checkPanel = new JPanel(new GridLayout(0, 1));
        checkPanel.add(checkBox1);
        checkPanel.add(checkBox2);
        checkPanel.add(checkBox3);
        frame.getContentPane().add(checkPanel);

        frame.setVisible(true);

    }

}
```



# Pasirinkimai JRadioButton

```
import java.awt.GridLayout;

import javax.swing.ButtonGroup;
import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.JRadioButton;

public class SimpleGUI {

    public static void main(String[] args) throws Exception {

        JFrame frame = new JFrame("My First GUI");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(600, 600);

        JRadioButton radioButton1 = new JRadioButton("pirmas pasirinkimas");
        JRadioButton radioButton2 = new JRadioButton("antras pasirinkimas");
        JRadioButton radioButton3 = new JRadioButton("trecias pasirinkimas");

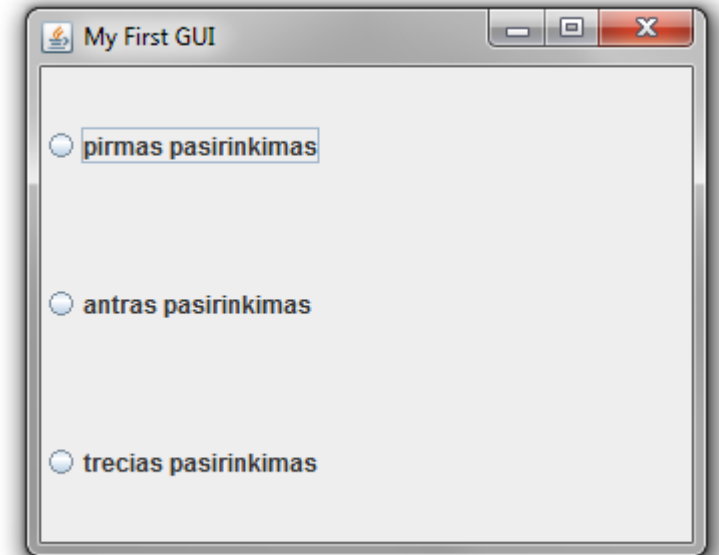
        ButtonGroup group = new ButtonGroup();
        group.add(radioButton1);
        group.add(radioButton2);
        group.add(radioButton3);

        JPanel radioPanel = new JPanel(new GridLayout(0, 1));
        radioPanel.add(radioButton1);
        radioPanel.add(radioButton2);
        radioPanel.add(radioButton3);

        frame.getContentPane().add(radioPanel);

        frame.setVisible(true);

    }
}
```



# Išsiskleidžiantys sąrašai JComboBox

```
import java.awt.FlowLayout;

import javax.swing.JComboBox;
import javax.swing.JFrame;
import javax.swing.JPanel;

public class SimpleGUI {

    public static void main(String[] args) throws Exception {

        JFrame frame = new JFrame("My First GUI");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(600, 600);

        String[] metuLaikai = { "Vasara", "Ruduo", "Žiema", "Pavasaris" };

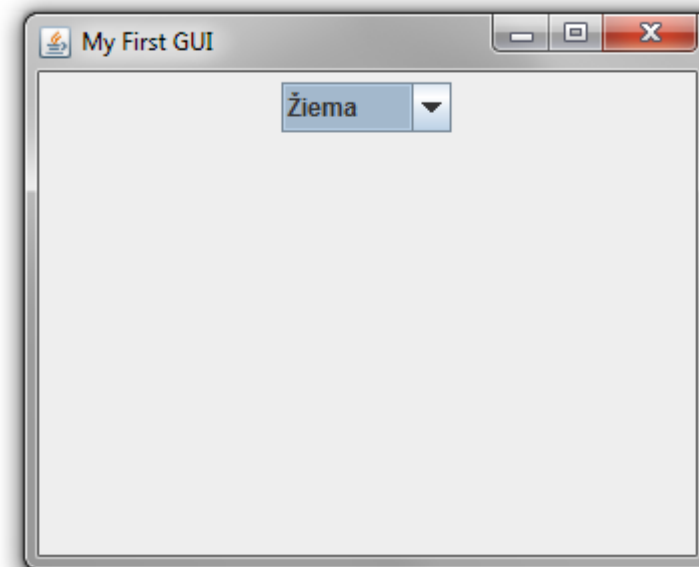
        JComboBox<String> metuLaikaiComboBox = new JComboBox<>(metuLaikai);
        metuLaikaiComboBox.setSelectedIndex(2);

        JPanel panel = new JPanel(new FlowLayout());
        panel.add(metuLaikaiComboBox);

        frame.getContentPane().add(panel);

        frame.setVisible(true);

    }
}
```





# Javedimo laukai JTextField ir JPasswordField

```
import java.awt.FlowLayout;

import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.JPasswordField;
import javax.swing.JTextField;

public class SimpleGUI {

    public static void main(String[] args) throws Exception {

        JFrame frame = new JFrame("My First GUI");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(600, 600);

        JTextField tekstas = new JTextField(20);
        JPasswordField slaptazodis = new JPasswordField(20);

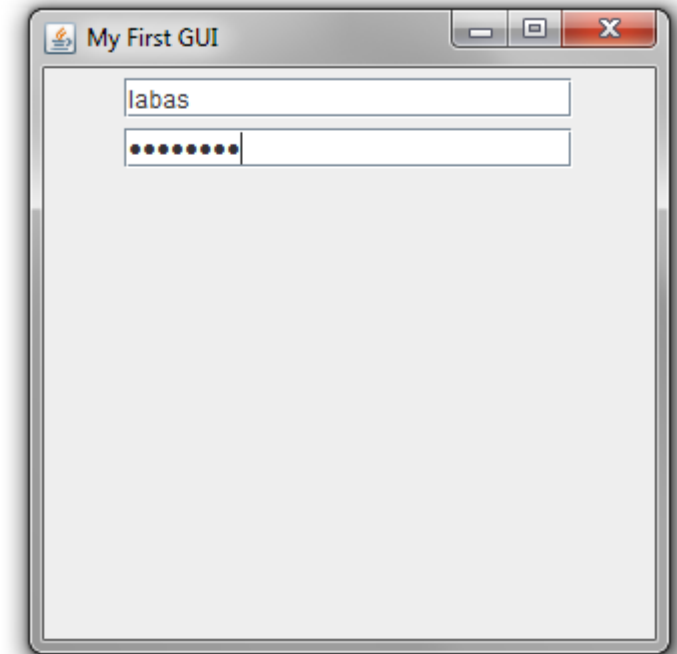
        JPanel panel = new JPanel(new FlowLayout());
        panel.add(tekstas);
        panel.add(slaptazodis);

        frame.getContentPane().add(panel);

        frame.setVisible(true);

    }

}
```



# Etiketè JLabel

```
import java.awt.FlowLayout;

import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;

public class SimpleGUI {

    public static void main(String[] args) throws Exception {

        JFrame frame = new JFrame("My First GUI");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(600, 600);

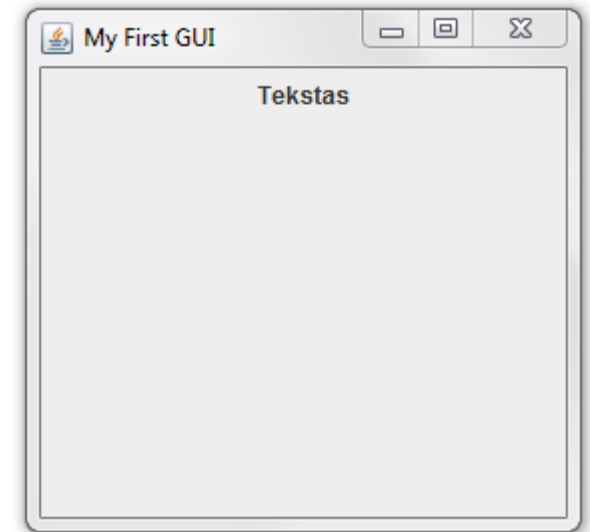
        JLabel label = new JLabel("Tekstas");

        JPanel panel = new JPanel(new FlowLayout());
        panel.add(label);

        frame.getContentPane().add(panel);

        frame.setVisible(true);

    }
}
```



# Progreso atvaizdavimas JProgressBar

```
import java.awt.FlowLayout;

import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.JProgressBar;

public class SimpleGUI {

    public static void main(String[] args) throws Exception {

        JFrame frame = new JFrame("My First GUI");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(600, 600);

        JProgressBar pBar = new JProgressBar();

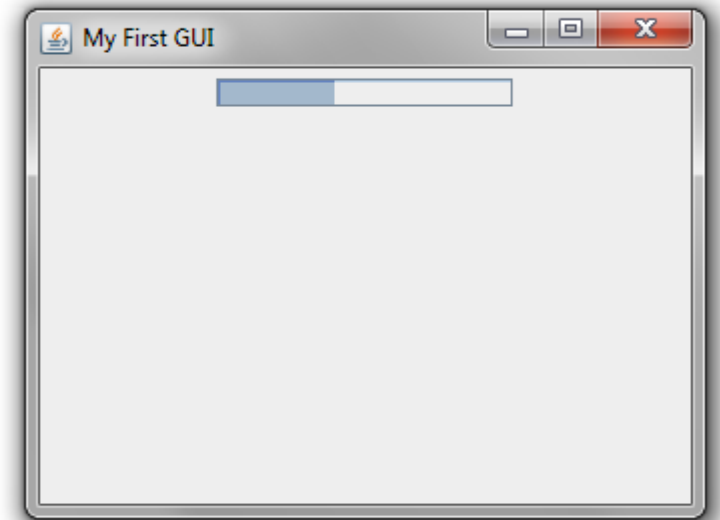
        JPanel panel = new JPanel(new FlowLayout());
        panel.add(pBar);

        frame.getContentPane().add(panel);

        frame.setVisible(true);

        for (int i = 0; i < 100; i++) {
            Thread.sleep(100);
            pBar.setValue(i);
        }

    }
}
```



# Sąrašai JList

```
import java.awt.FlowLayout;

import javax.swing.JFrame;
import javax.swing.JList;
import javax.swing.JPanel;

public class SimpleGUI {

    public static void main(String[] args) throws Exception {

        JFrame frame = new JFrame("My First GUI");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(600, 600);

        String[] savaiteDienos = { "Pirmadieni", "Antradienis",
                                   "Treciadienis", "Ketvirtadienis", "Penktadienis",
                                   "Sestadienis", "Sekmadinis" };

        JList<String> sarasas = new JList<>(savaiteDienos);

        JPanel panel = new JPanel(new FlowLayout());
        panel.add(sarasas);

        frame.getContentPane().add(panel);

        frame.setVisible(true);

    }
}
```



# Pavyzdys: įtraukimas į sąrašą

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class SimpleGUI {

    public static void main(String[] args) throws Exception {

        JFrame frame = new JFrame("My First GUI");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(300, 600);

        DefaultListModel<String> model = new DefaultListModel<>();
        JList<String> sarasas = new JList<>(model);

        JLabel labelVardas = new JLabel("Vardas");
        JTextField vardas = new JTextField(20);
        JLabel labelPavarde = new JLabel("Pavarde");
        JTextField pavarde = new JTextField(20);
        JButton itrauktiMygtukas = new JButton("Įtraukti į sąrašą");
        JButton isvalymoMygtukas = new JButton("Išvalyti sąrašą");

        itrauktiMygtukas.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                String ivestasVardas = vardas.getText();
                String ivestaPavarde = pavarde.getText();
                model.addElement(ivestasVardas + " " + ivestaPavarde);
                vardas.setText(null);
                pavarde.setText(null);
            }
        });

        isvalymoMygtukas.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                model.clear();
            }
        });

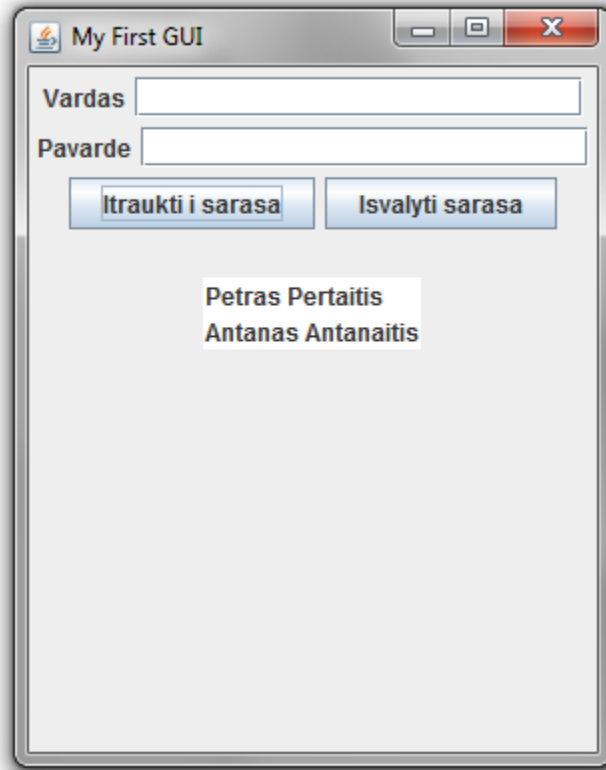
        JPanel panelVirusus = new JPanel();
        panelVirusus.setPreferredSize(new Dimension(300, 100));
        JPanel panelCentras = new JPanel();
        panelVirusus.add(labelVardas);
        panelVirusus.add(vardas);
        panelVirusus.add(labelPavarde);
        panelVirusus.add(pavarde);
        panelVirusus.add(itrauktiMygtukas);
        panelVirusus.add(isvalymoMygtukas);
        panelCentras.add(sarasas);

        frame.getContentPane().add(BorderLayout.NORTH, panelVirusus);
        frame.getContentPane().add(BorderLayout.CENTER, panelCentras);

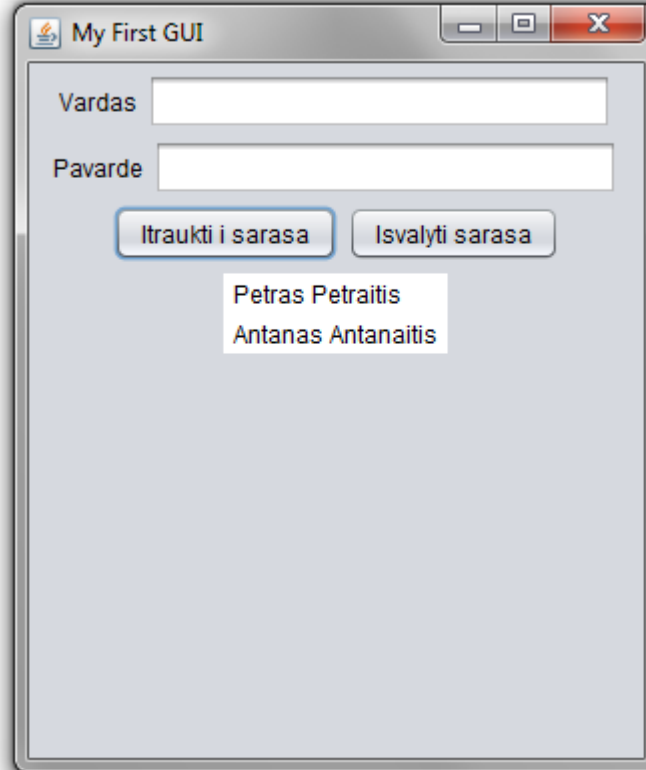
        frame.setVisible(true);
    }
}
```



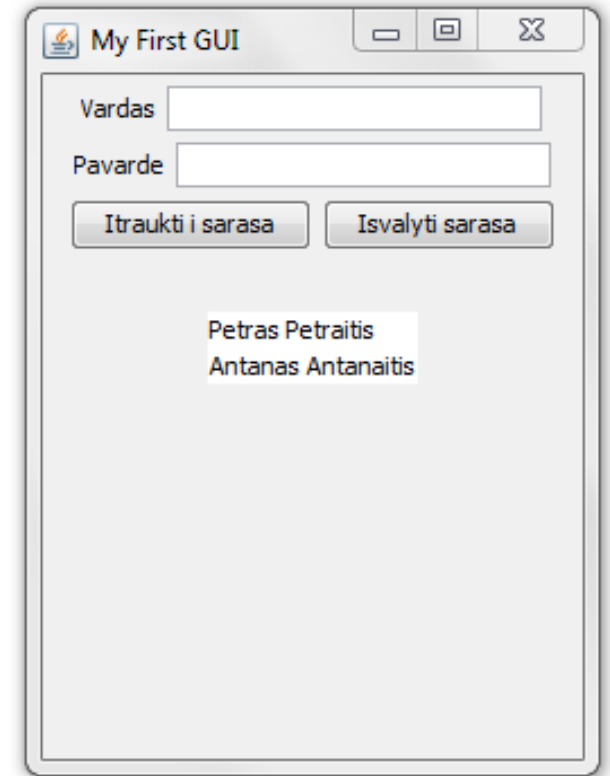
# Išvaizdos pakeitimas



Default



Nimbus



Windows

```
UIManager.setLookAndFeel("javax.swing.plaf.nimbus.NimbusLookAndFeel");  
UIManager.setLookAndFeel("com.sun.java.swing.plaf.windows.WindowsLookAndFeel");
```

# Nuorodos

- Swing dokumentacija: <https://docs.oracle.com/javase/tutorial/uiswing/>
- Swing komponentai:  
<https://docs.oracle.com/javase/tutorial/uiswing/components/index.html>
- Look & Feel Nimbus:  
<https://docs.oracle.com/javase/tutorial/uiswing/lookandfeel/nimbus.html>
- Simple example: <https://beginnersbook.com/2015/07/java-swing-tutorial/>
- Layouts:  
<https://docs.oracle.com/javase/tutorial/uiswing/layout/visual.html>
- Swing tutorial: <https://www.guru99.com/java-swing-gui.html>



# varargs

Martynas Mitrulevičius



# Problema

```
public class VarargTest {  
  
    public static void main(String[] args) {  
  
        int suma1 = sudetiSkaicius(1, 2);  
        int suma2 = sudetiSkaicius(1, 2, 5);  
        int suma3 = sudetiSkaicius(1, 2, 4, 6);  
  
        System.out.println(suma1);  
        System.out.println(suma2);  
        System.out.println(suma3);  
  
    }  
  
    private static int sudetiSkaicius(int a, int b) {  
        return a + b;  
    }  
  
    private static int sudetiSkaicius(int a, int b, int c) {  
        return a + b + c;  
    }  
  
    private static int sudetiSkaicius(int a, int b, int c, int d) {  
        return a + b + c + d;  
    }  
  
}
```

Metodas kviečiamas perduodant parametrus. Skirtingais atvejais gali reikėti metodų su skirtingu skaičiumi parametų. Tam turime parašyti daug beveik vienodų metodų su skirtingu parametų skaičiumi.

# Problemos sprendimas

```
public class VarargTest {  
  
    public static void main(String[] args) {  
  
        int suma1 = sudetiSkaicius(1, 2);  
        int suma2 = sudetiSkaicius(1, 2, 5);  
        int suma3 = sudetiSkaicius(1, 2, 4, 6);  
  
        System.out.println(suma1);  
        System.out.println(suma2);  
        System.out.println(suma3);  
  
    }  
  
    private static int sudetiSkaicius(int... skaiciai) {  
        int rez = 0;  
        for (int i : skaiciai) {  
            rez += i;  
        }  
        return rez;  
    }  
  
}
```

# Varags sintaksė

- ```
void metodoPavadinimas(String x, String y, int... z) {  
    // z naudojamas taip pat kaip masyvas, pvz:  
    int suma = z[0] + z[1];  
}
```
- Reikia nepamiršti, kad `z` gali būti tuščias
- Varargs – visada paskutinis parametras metodo parametrų sąrašė

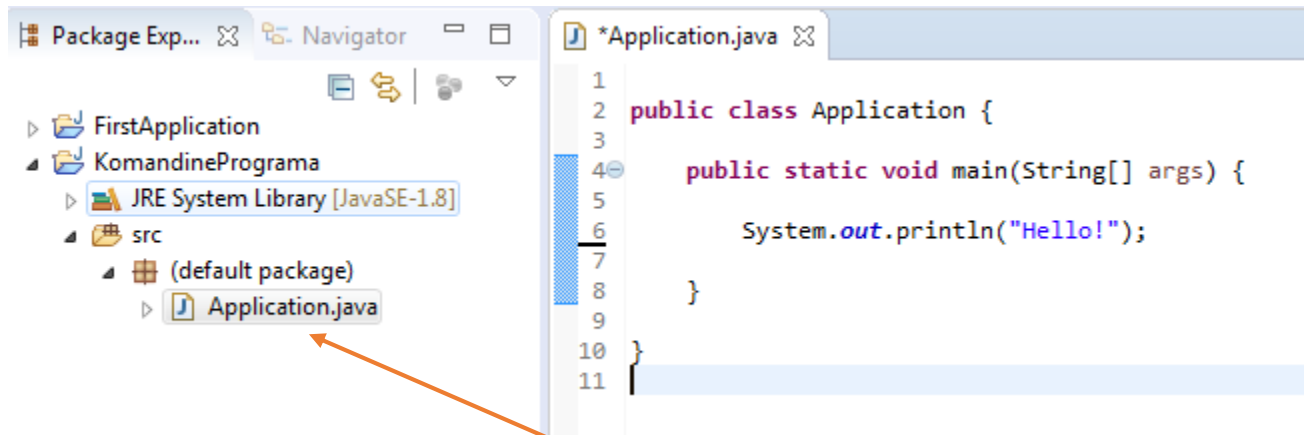
# Jau naudojome varags

- `String.format("This is an integer: %d", myInt);`
- `String.format("This is an integer: %d and a string: %s", myInt, myString);`



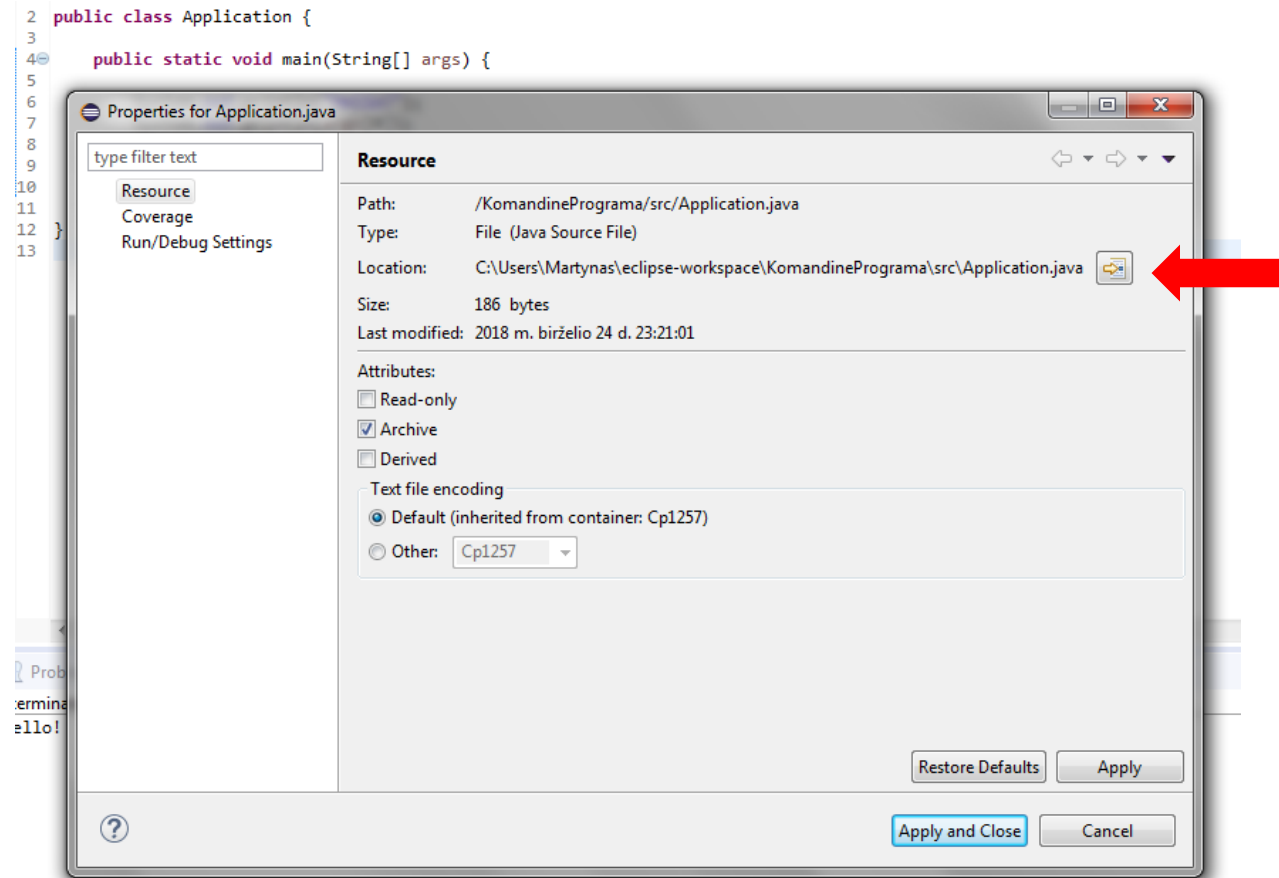
# Komandinė eilutė

# Turime Java programą



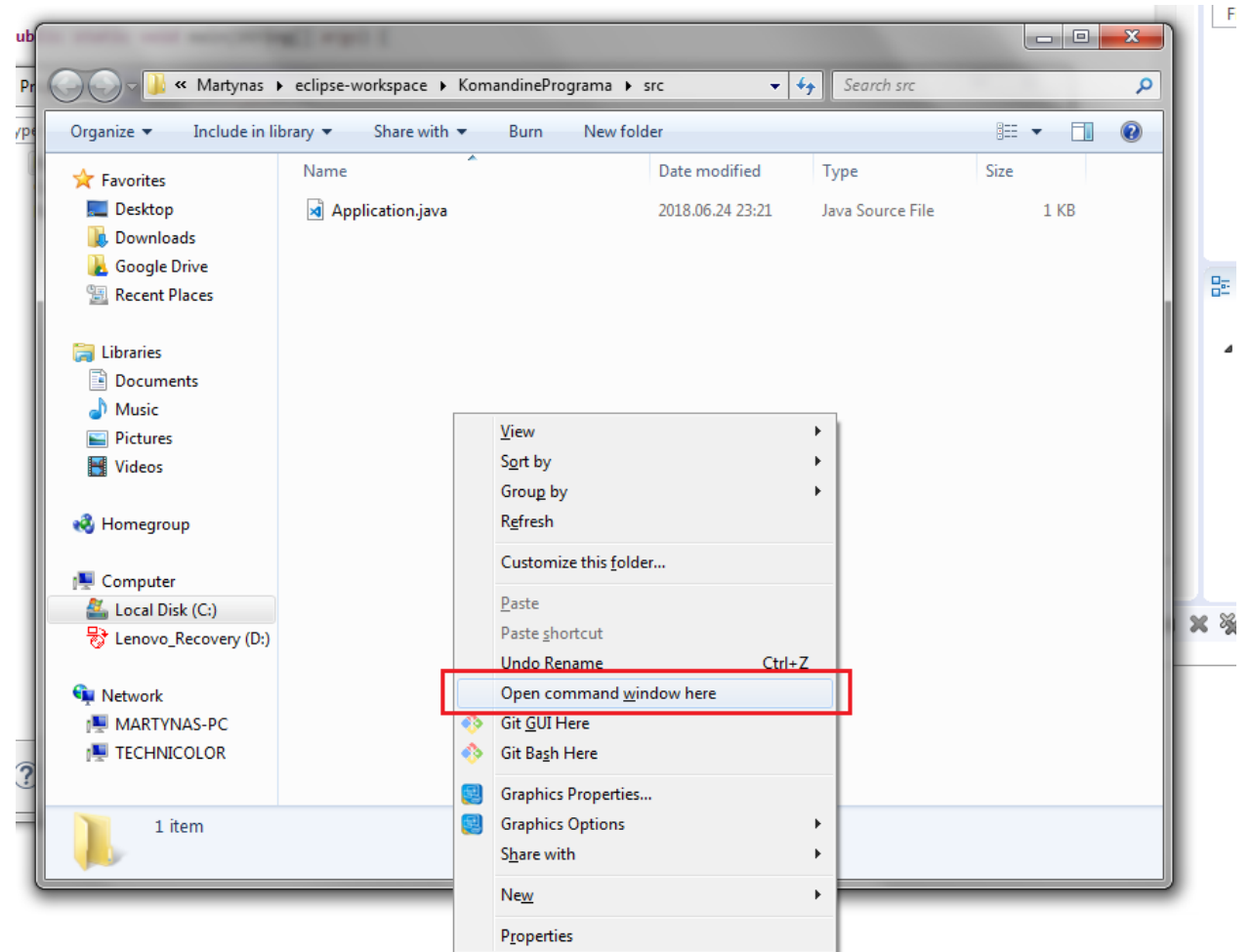
Kad sužinotume, kur yra saugoma mūsų programa, and **Application.java** spaudžiame dešinį pelės klavišą ir pasirenkame **Properties**

# Einame į vietą, kur saugoma Java programa



# Atidarome CMD

Atsidarius failų langui laikydami  
nuspaukę *Shift* spaudžiame dešinį  
pelės klavišą ir pasirenkame  
*Open command window here*



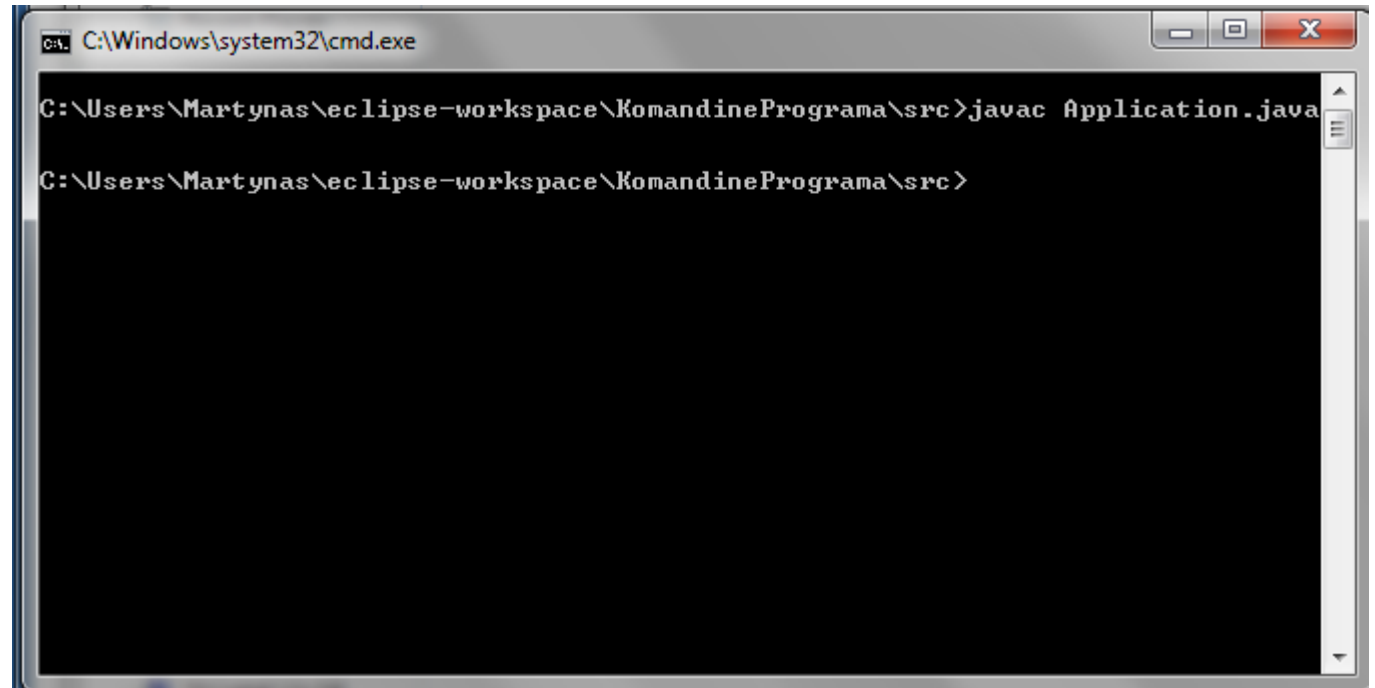


# Kompiluojame

Kompiluojame komanda:

`javac Application.java`

Kompilatorius sukurs `Application.class` failą, kuris yra ta pati mūsų parašyta programa tik paversta į binarinį kodą, Kurį supranta JVM.

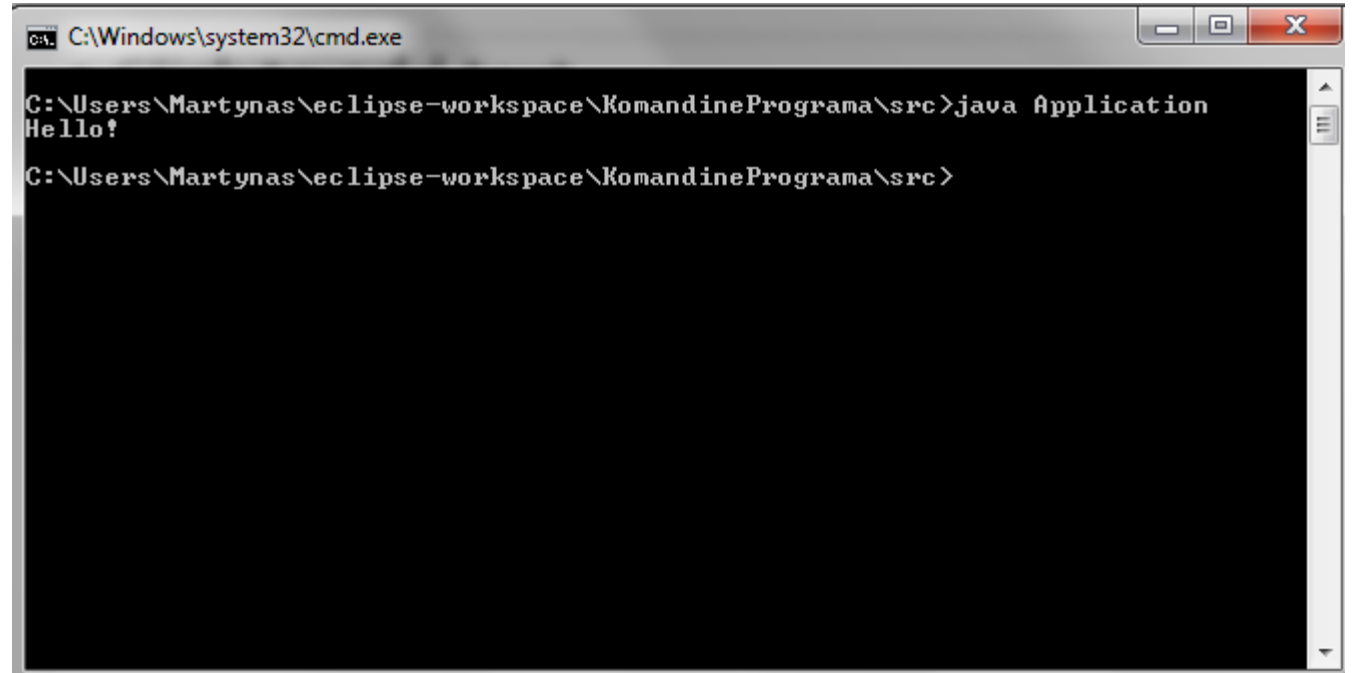


```
C:\Windows\system32\cmd.exe
C:\Users\Martynas\eclipse-workspace\KomandinePrograma\src>javac Application.java
C:\Users\Martynas\eclipse-workspace\KomandinePrograma\src>
```

# Paleidžiame programą

Paleidžiame komanda:

`java Application`



```
C:\Windows\system32\cmd.exe

C:\Users\Martynas\eclipse-workspace\KomandinePrograma\src>java Application
Hello!

C:\Users\Martynas\eclipse-workspace\KomandinePrograma\src>
```

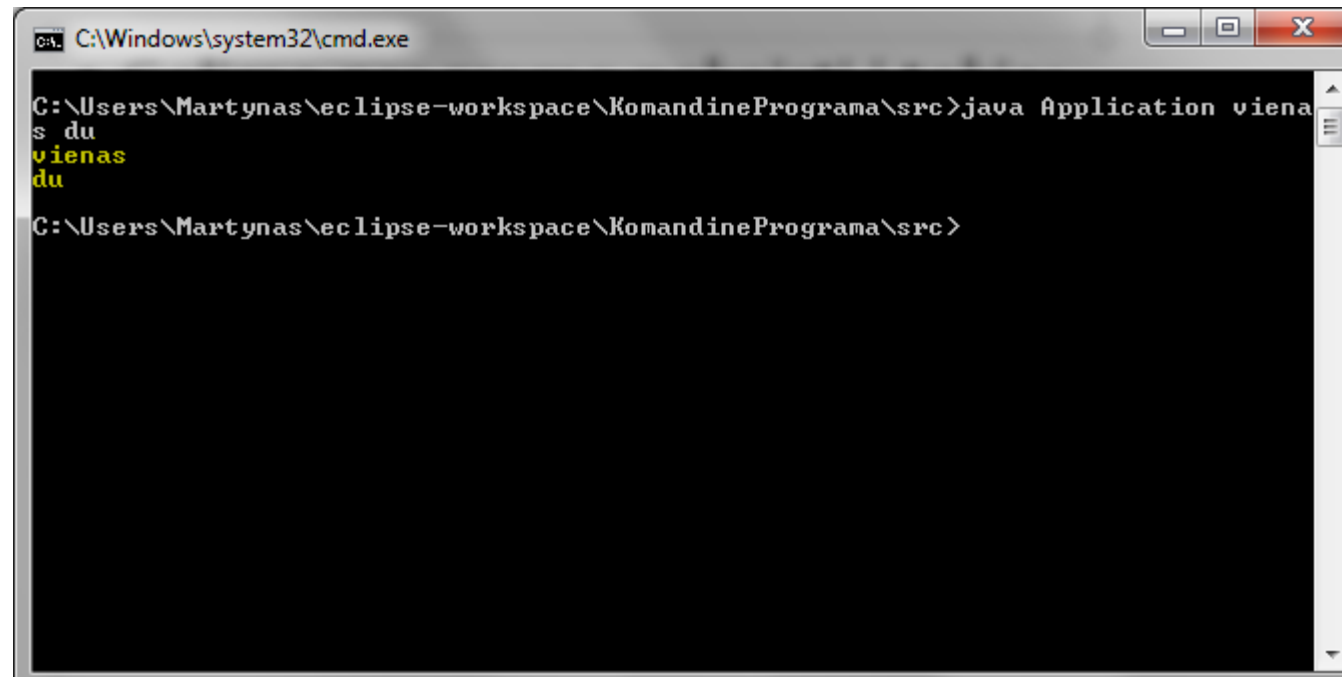
# Argumentu perdavimas

- Galime programą pakeisti į tokią:

```
public class Application {  
    public static void main(String[] args) {  
        System.out.println(args[0]);  
        System.out.println(args[1]);  
    }  
}
```

- Tada turėsime programai perduoti argumentus, kuriuos ji atspausdins, pvz:

`java Application vienas du`

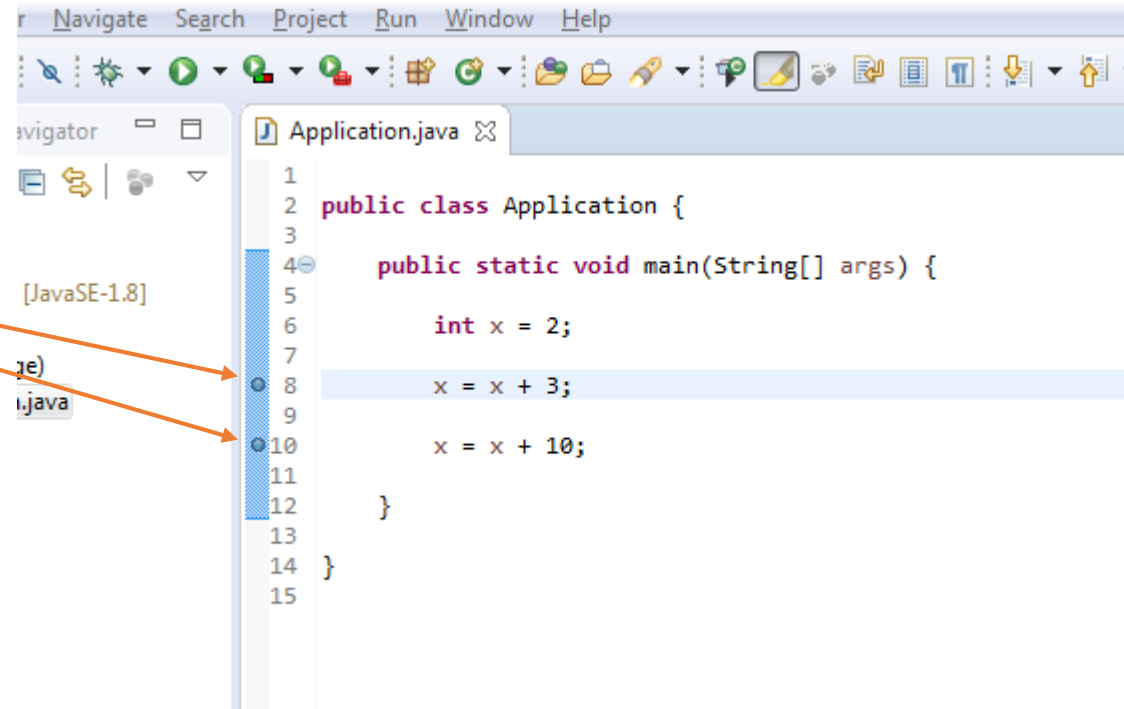


```
C:\Windows\system32\cmd.exe  
C:\Users\Martynas\eclipse-workspace\KomandinePrograma\src>java Application vienas  
s du  
vienas  
du  
C:\Users\Martynas\eclipse-workspace\KomandinePrograma\src>
```



Derinimas (*eclipse*)

Turime tokią Java programą.  
Joje pasižymime eilutes, ties kuriomis  
norėsime sustabdyti programą.  
Toje eilutėje pele paspaudžiame  
du kartus.

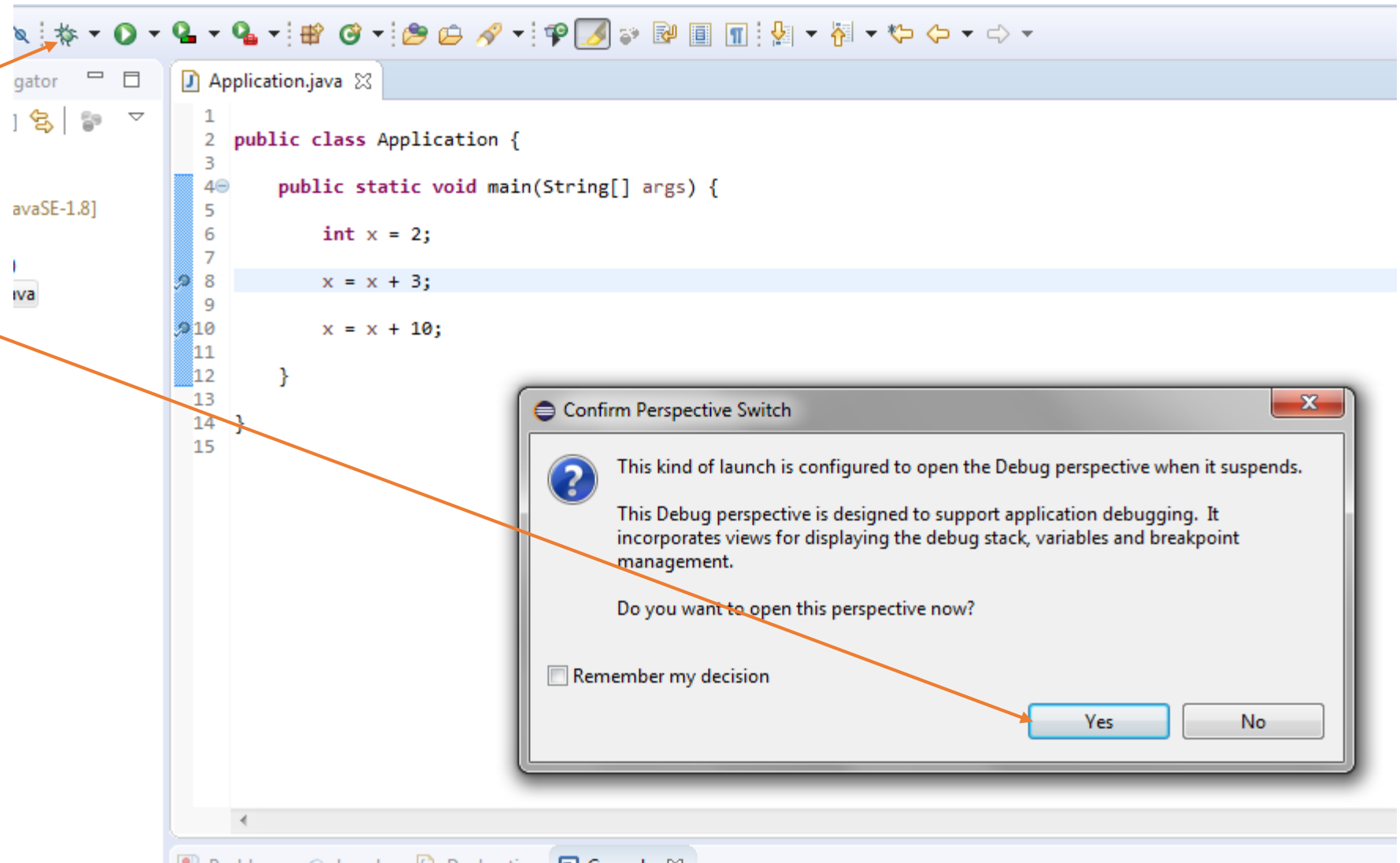


The screenshot shows an IDE window titled "Application.java". The code is as follows:

```
1  
2 public class Application {  
3  
4     public static void main(String[] args) {  
5  
6         int x = 2;  
7  
8         x = x + 3;  
9  
10        x = x + 10;  
11  
12    }  
13  
14 }  
15
```

Two orange arrows originate from the text on the left. The first arrow points to line 8, and the second arrow points to line 10. In the IDE, lines 8 and 10 are highlighted with a blue background, and the line numbers 8 and 10 in the left margin are also highlighted.

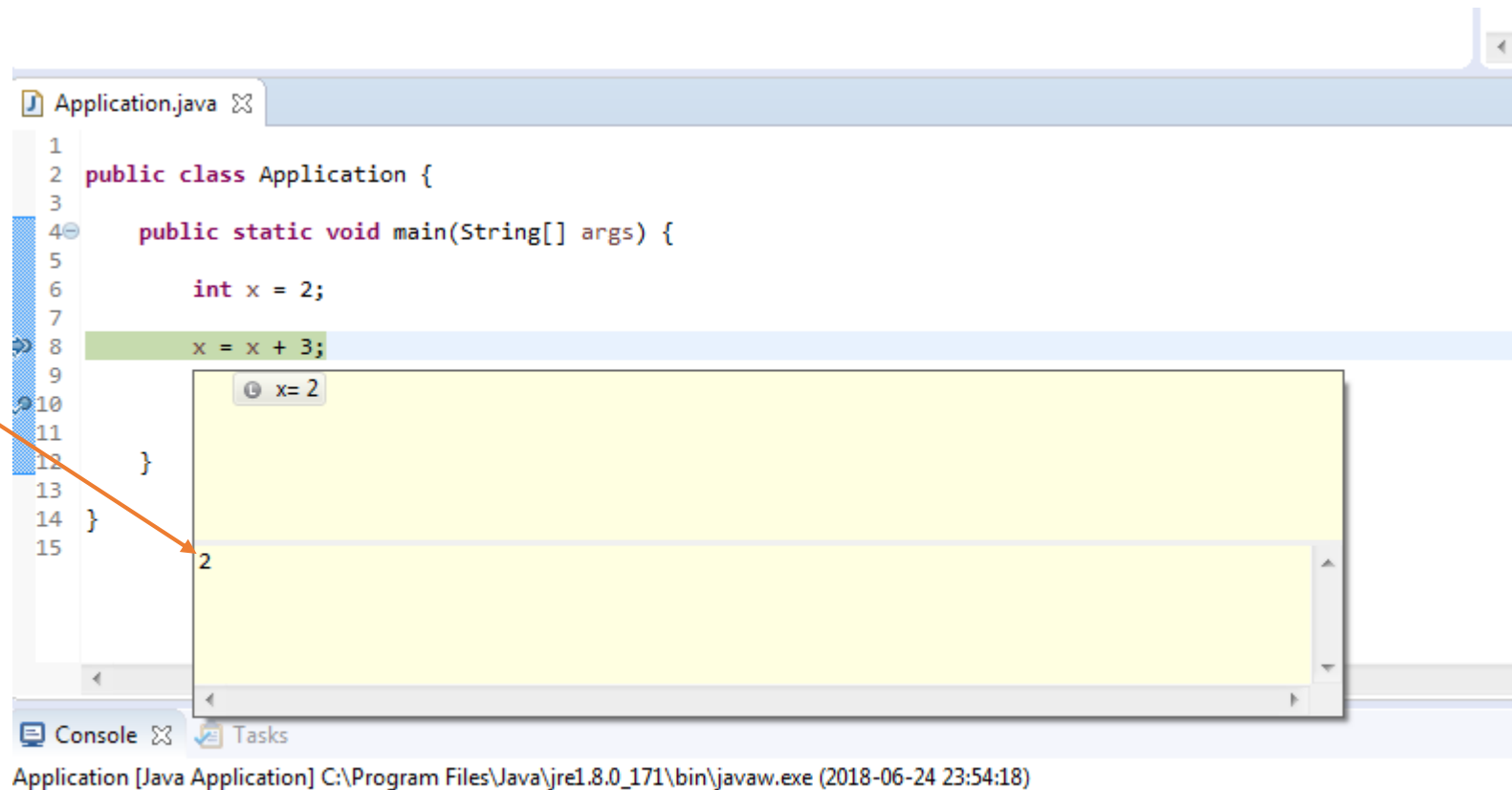
Spaudžiame *debug* mugtuką. Gali iššokti langas, informuojantis, kad perjungs į debug langą. Spaudžiame Yes.



Užvedame pele ties  
kintamuoju ir matome  
jo reikšmę.

Kad pereiti prie kitos  
pažymėtos eilutės,  
Spaudžiame F8.

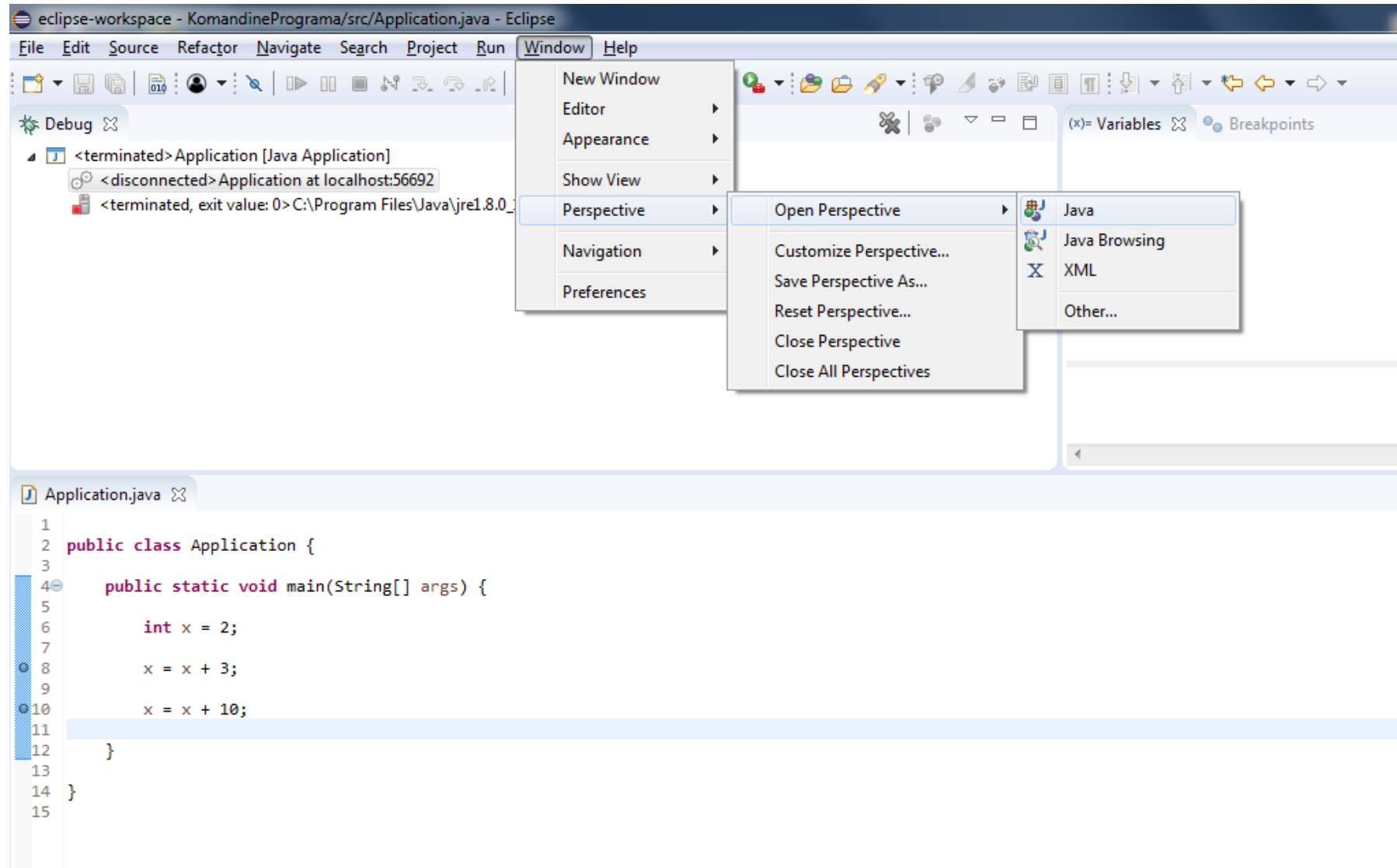
Tokios pažymėtos  
Eilutės vadinamos *breakpoints*.



The screenshot shows an IDE window titled "Application.java". The code is as follows:

```
1  
2 public class Application {  
3  
4     public static void main(String[] args) {  
5  
6         int x = 2;  
7  
8         x = x + 3;  
9  
10    }  
11  
12 }  
13  
14 }  
15
```

A breakpoint is set on line 8, indicated by a blue dot in the left margin. A yellow tooltip is visible over the breakpoint, showing the variable `x = 2`. Below the code editor, there are tabs for "Console" and "Tasks". At the bottom, the status bar shows "Application [Java Application] C:\Program Files\Java\jre1.8.0\_171\bin\javaw.exe (2018-06-24 23:54:18)".



Grįžimas į prieš tai buvusį (ne *debug*) langą: *Window -> Perspective -> Open Perspective -> Java*





# JAR failo sukūrimas (eclipse)

# Kas yra JAR failas?

- Tai failo formatas į kurį galime supakuoti Java programą
- JAR failas remiasi ZIP archyvo formatu. JAR failą galime atidaryti bet kuria failų archyvavimo programa (WinZip, 7-Zip, ...)
- JAR failo viduje sudėtos visos Java programos sukompiliuotos klasės (\*.class) ir kiti programme naudojami resursai (tekstai, paveikslėliai, ...)

# SumService klasé

```
package lt.codeacademy.service;

public class SumService {

    public static int getSumOf(int firstNumber, int secondNumber) {
        return firstNumber + secondNumber;
    }

}
```

# SumApp klasé

```
package lt.codeacademy;

import lt.codeacademy.service.SumService;

public class SumApp {

    public static void main(String[] args) {

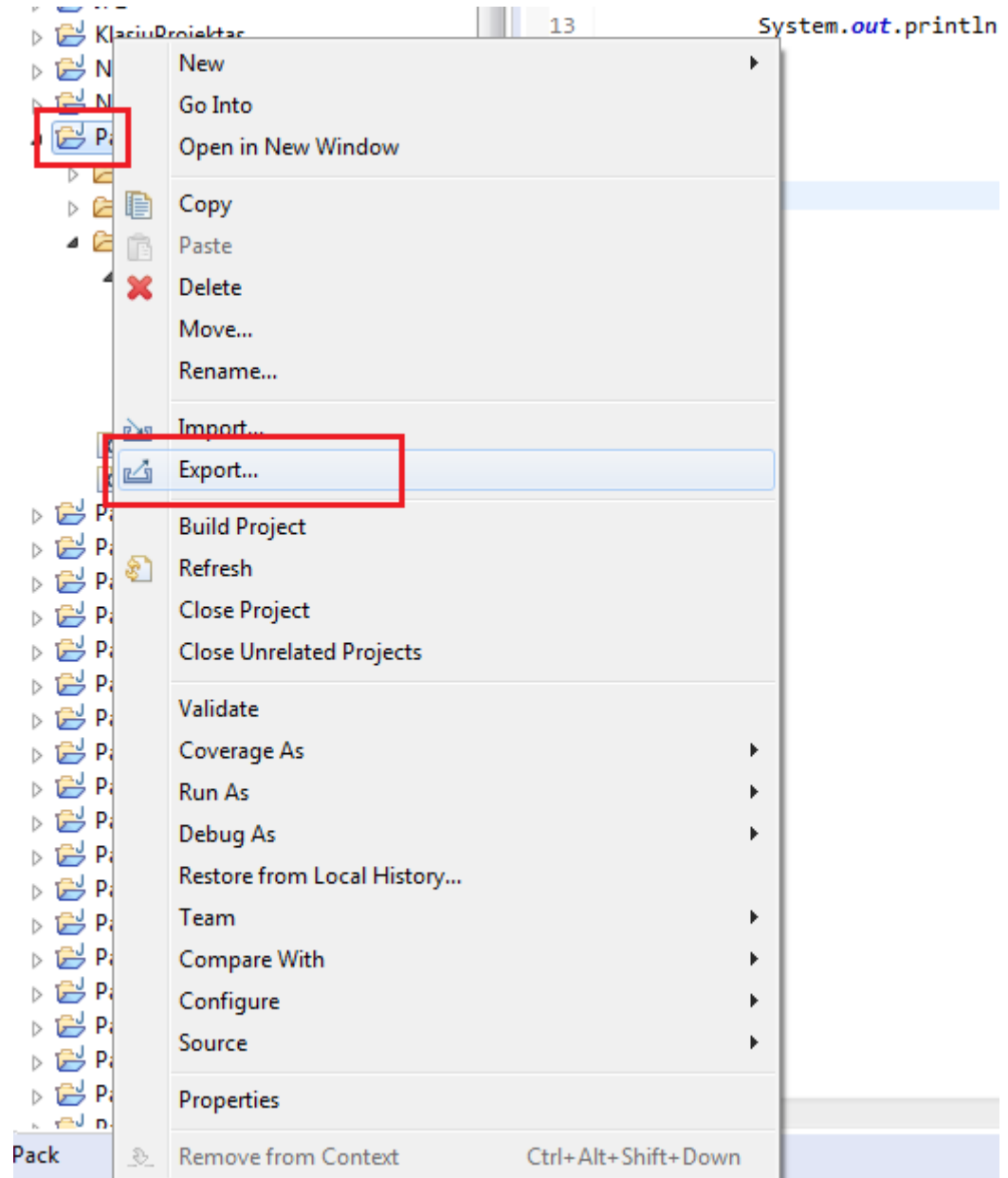
        if (args.length >= 2) {
            int number1 = Integer.parseInt(args[0]);
            int number2 = Integer.parseInt(args[1]);
            int result = SumService.getSumOf(number1, number2);
            System.out.println("Sum: " + result);
        }

    }

}
```

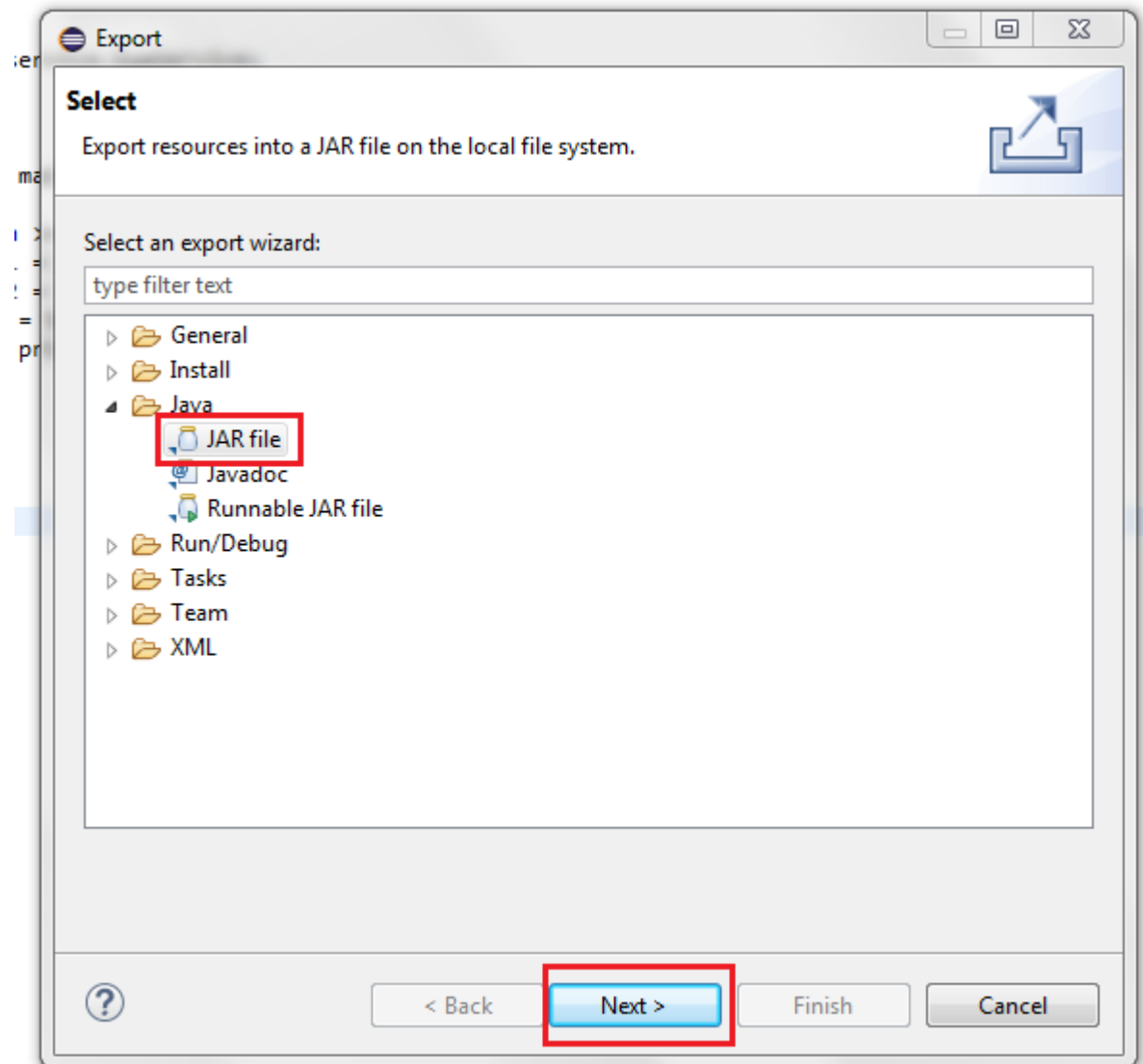
# Eksportavimas

- Pasirinkti Java projektą ir spausti dešinį pelės klavišą
- Atsidarius pasirinkimų meniu pasirinkti **Export**



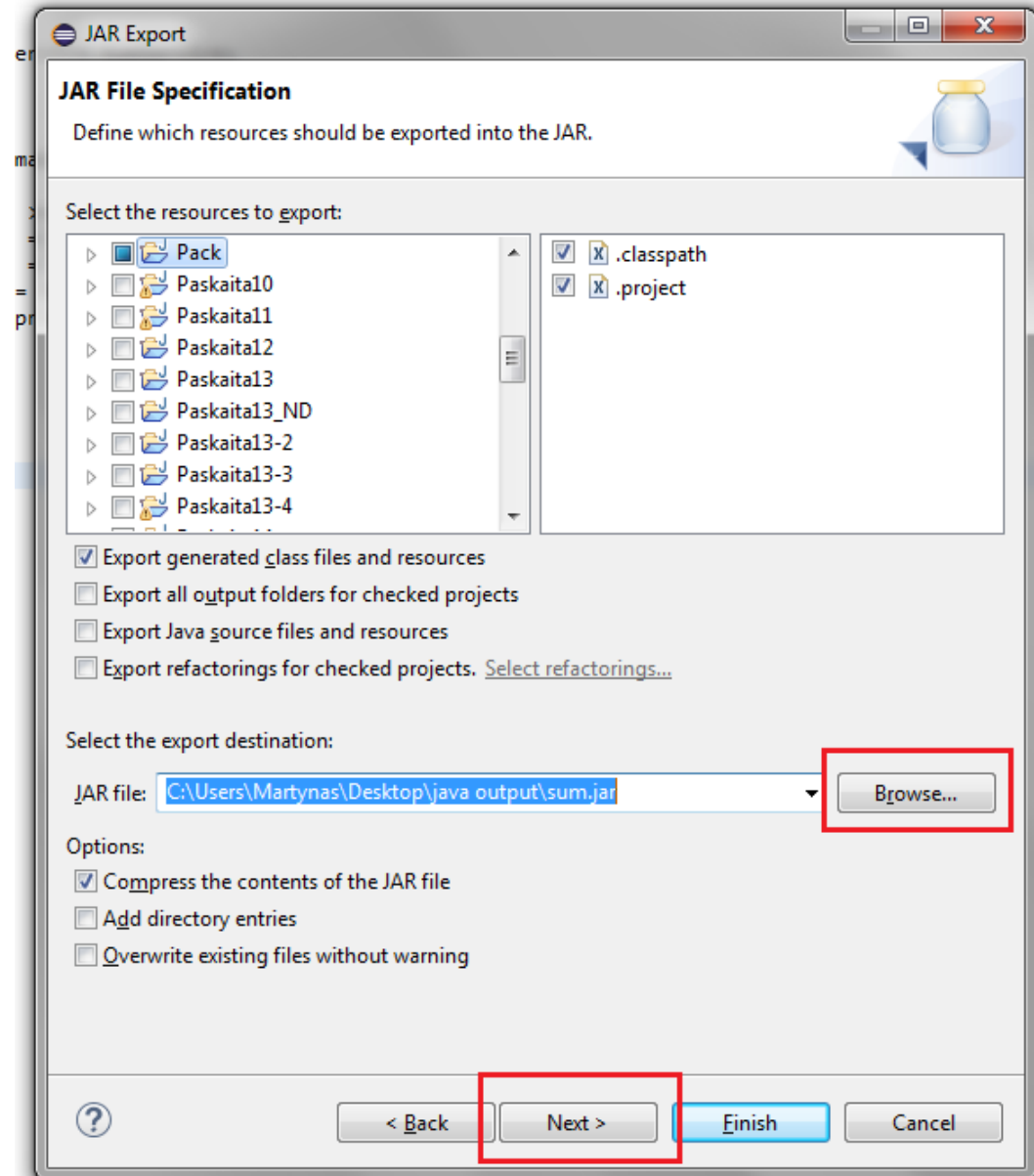
# Eksportavimas

- Atsidariusiame lange išsiskleisti Java grupę ir pasirinkti **JAR file**
- Spausti **Next**



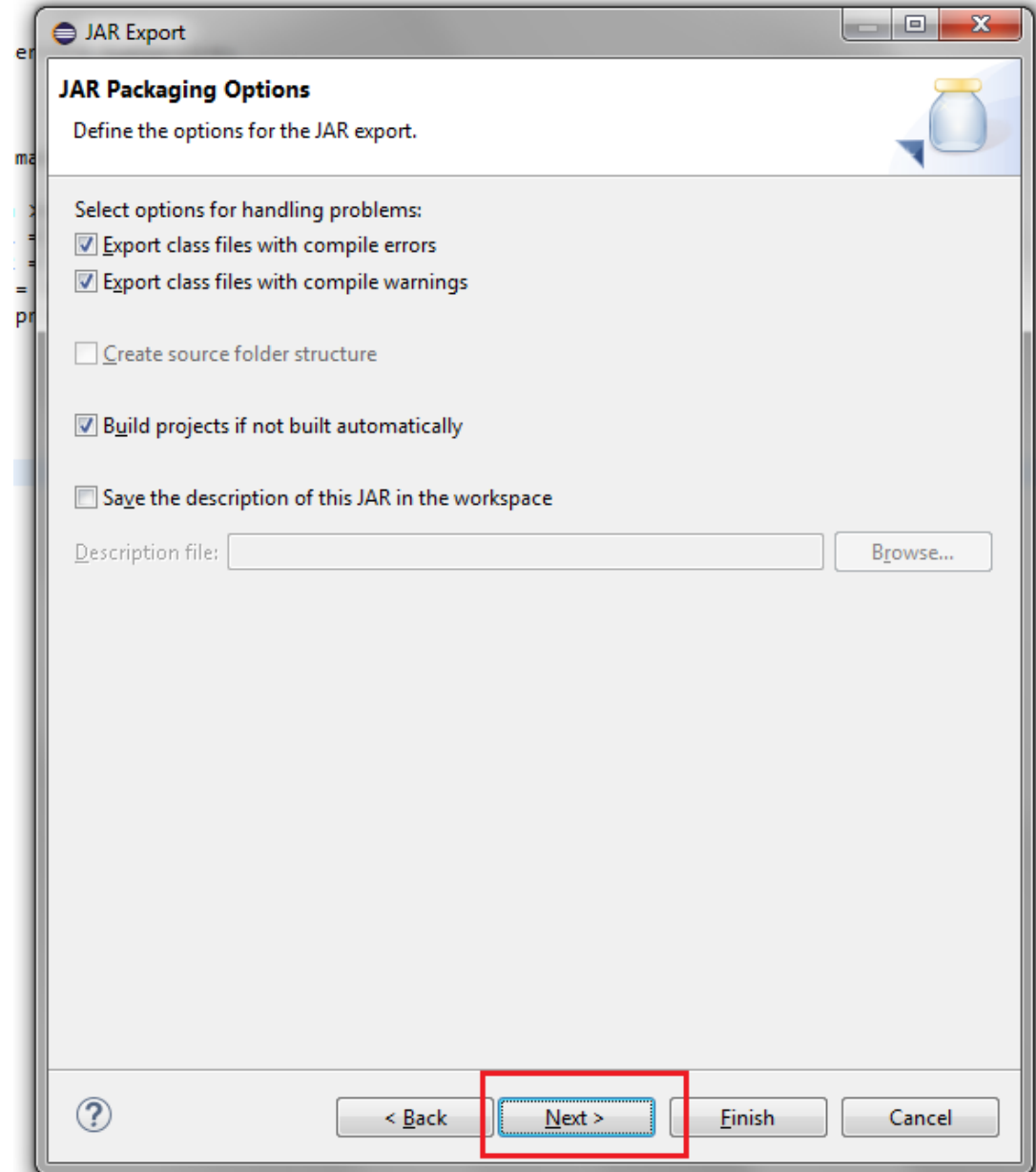
# Eksportavimas

- Kitame lange spaudžiame **Browse** ir pasirenkame, kur saugoti JAR failą
- Po to spaudžiame **Next**



# Eksportavimas

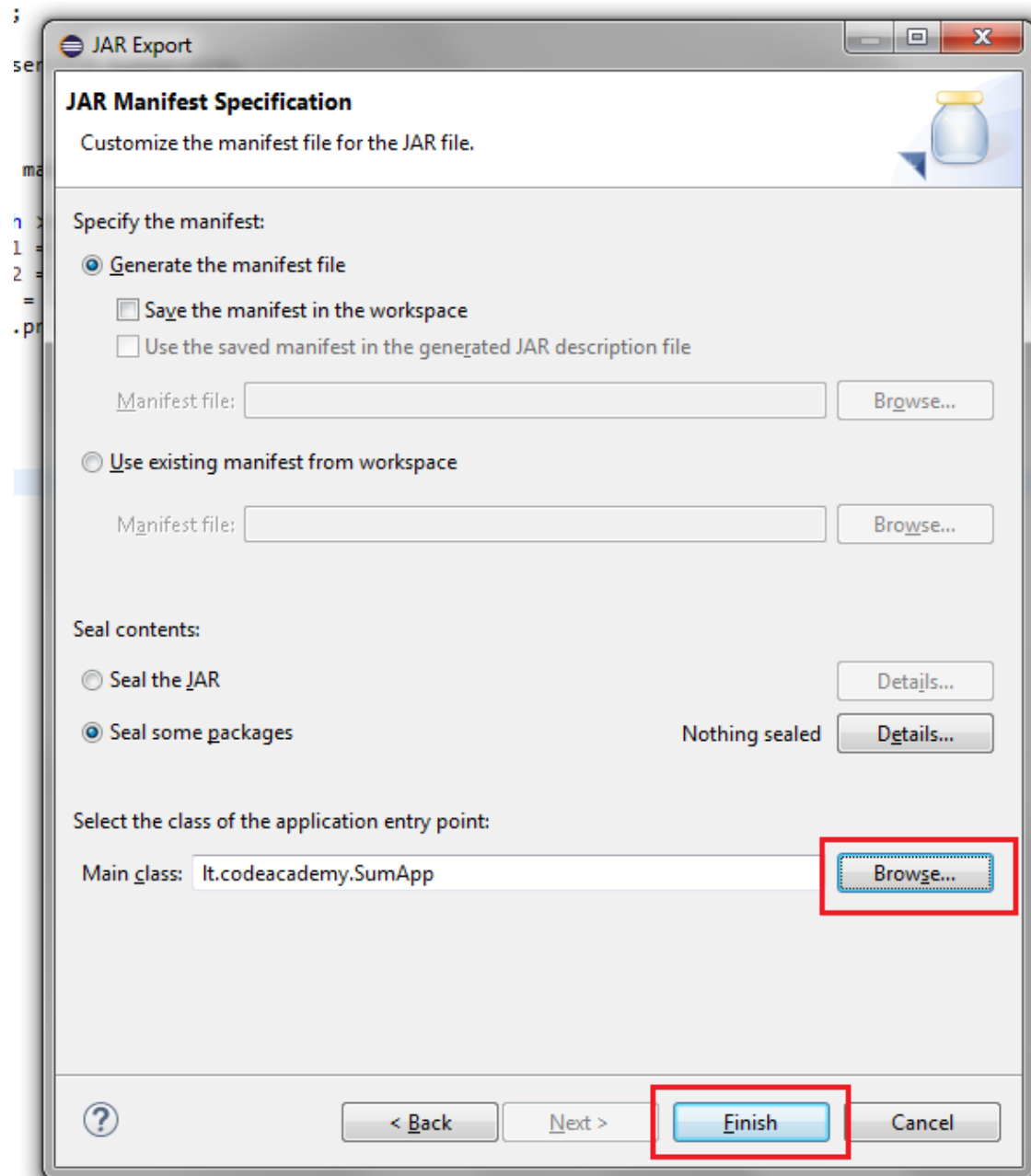
- Dar kitame lange nieko nekeičiame ir spaudžiame **Next**



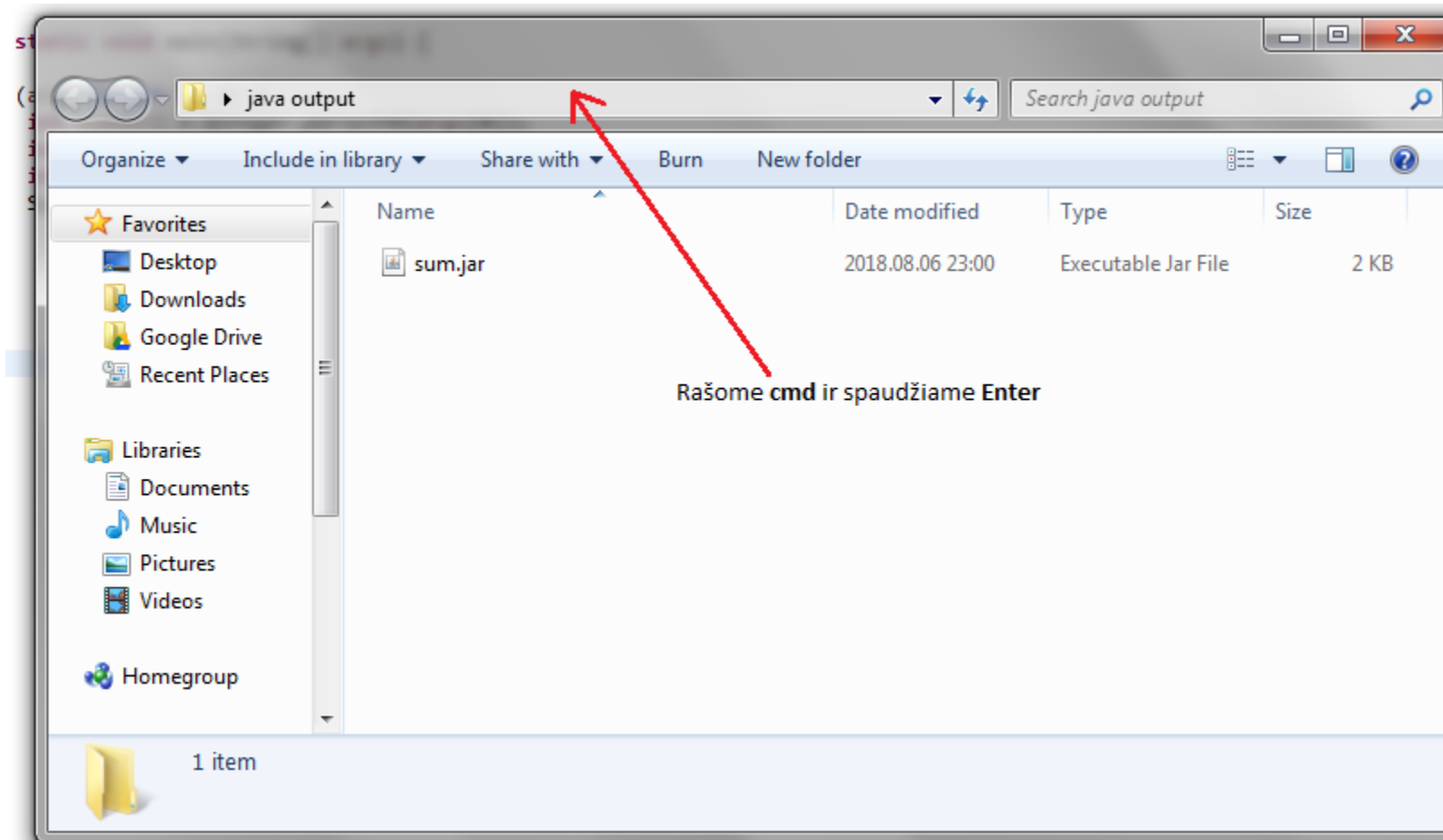


# Eksportavimas

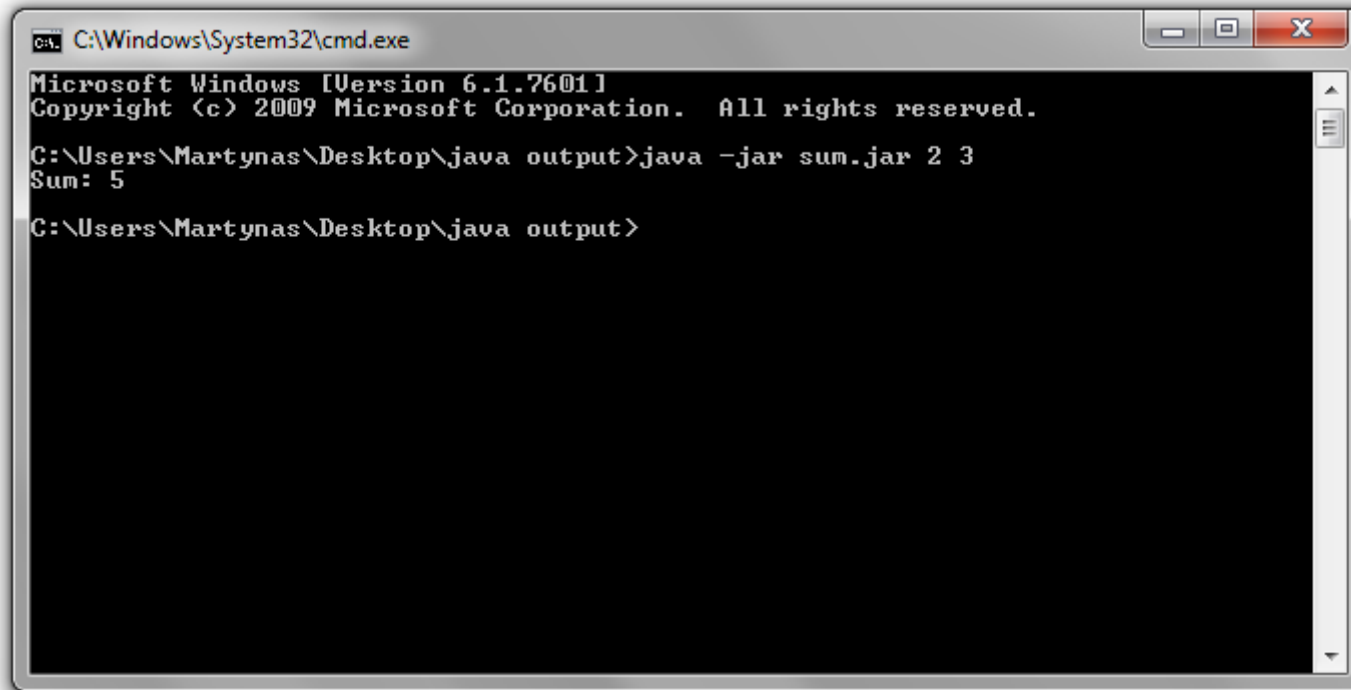
- Spaudžiame **Browse** ir pasirenkame klasę su **main** metodu
- Spaudžiame **Finish**



# Programos paleidimas



# Programos paleidimas



```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\Martynas\Desktop\java output>java -jar sum.jar 2 3
Sum: 5

C:\Users\Martynas\Desktop\java output>
```

- Programą paleidžiame komanda: **java -jar <jarFailas.jar> <programos argumentai>**
- Pvz.: **java -jar sum.jar 2 3**