Vidinės klasės

Mindaugas Karpinskas
2018

Parengta pagal www.cloudbus.org/~raj/254/Lectures/RajLec7.ppt

Ne-vidine klasė

NonInner.java

```
package lt.codeacademy.week7.innerclass;

public class NonInner {
    public static void main(String[] args) {
        System.out.println("First class main method");
    }
}
class NonInner2 {
    public static void main(String[] args) {
        System.out.println("Second class main method");
    }
}
```

Vidinės (inner) klasės

Sintaksė:

```
[modifiers] class OuterClassName {
    code...
    [modifiers] class InnerClassName {
        code....
    }
}
```

- Išorinė klasė (kurioje yra vidinė) gali sukurtu daug vidinės klasės egzempliorių
- Jei vidinė ir išorinė yra public, kitos klasės taip pat gali kurti vidinės klasės tipo egzempliorius. Tai daroma tokiu būdu:

*<OuterClassName> outerObj = **new** <OuterClassName>(arguments);*
*<OuterClassName>.<InnerClassName> innerObj = outerObj.**new** <InnerClassName>(arguments);*

- Kai sukuriamas išorinės klasės egzempliorius, automatikai joks vidinės klasės egzepliorius neatsiranda.
- Jei vidinė klasė yra statinė, jos egzempliorius gali būti sukurtas nekuriant išorinės klasės egzemplioriaus.
- Vidinė klasė turi prieigą prie visų ją sukūrusios išorinės klasės elementų (!netgi prie *private*). Jei vidinė klasė turi to paties pavadinimo elementą, pasiekti išorinį elementą galima tokiu būdu:
  <OuterClassName>.this.<variableName>

PVZ

```
package lt.codeacademy.week7.innerclass.examples;

public class Isorine {
    public class Vidine {

    }
}
```

Uzd 1 (vidinės klasės)

- Sukurit dvi klasės:
  - OuterClass - public klasę
  - InnerClass - OuterClass klasės bloke sukurkite dar vieną public klasę

Main metode pabandyti sukurti abejų klasių egzempliorių

```
[modifiers] class OuterClassName {
    code...
    [modifiers] class InnerClassName {
        code....
    }
}
```

OuterClass

```
public class OuterClass {
    int a;

    public class InnerClass {
        int a;
    }
    public static void main(String[] args) {
        OuterClass outerClass = new OuterClass();
        InnerClass innerClass = outerClass.new InnerClass();
    }
}
```

Uzd 2 (vidinės klasės) ND

- Sukurit dvi klasės:
  - IsorineKlase2 - public klasę
    - Turi kintamuosius, pvz.:
      - private String pavadinimas;
      - protected String nr;
      - int sk;
  - VidineKlase2 - IsorineKlase2 klasės bloke sukurkite dar vieną public klasę
    - int sk;
    - Turi metoda spausdinti()

Main metode pabandyti sukurti
abejų klasių egzempliorių
IsorineKlase2 klasės kintamiesiams priskirti reikšmes
ir jas atspausdinti VidineKlase2 klasės *spausdinti*() metodo pagalba

Statinė vidinė klasė

Sintaksė:

```
<access-specifier> class OuterClassName {
    public static class <StaticInnerClassName> {
        . . .
    }
    . . .
}
```

- **Papildomos *static* vidinės klasės savybės:**
  - Statiniai išorinės klasės elementai vidiniai statiniai prieinami **visada**.
  - Ne statiniai neprieinami, nes statinė vidinė klasė nepriklauso egzemplioriui
  - Jei vidinė klasė nepažymėta *static,* ji negali turėti statinių elementų.
  - Vienintelis skirtumas statinės vidinės klasės nuo kitų vidinių klasių, kad ji neturi nuorodos į išorinę klasę, kurioje ji patalpinta.

Statinės vidinės klasės

- Dar vadinamos įterptinėmis (nested) klasėmis
- Norint sukurti vidinės klasės objektą nereikia turėti išorinės klasės objekto
- Iš jos negalima pasiekti nestatinio išorinės klasės objekto
- Įterptinę klasę galima aprašyti ir intefeiso viduje

Uzd 3 (static vidinės klasės)

- Sukurit dvi klasės:
  - **OuterClassStaticTest** - public klasę
    - Turi kintamuosius, pvz.:
      - private String pavadinimas;
      - protected String nr;
      - int sk;
  - **InnerStaticClass** - OuterClassStaticTest klasės bloke sukurkite dar vieną public klasę
    - Pabandyti pasiekti **OuterClassStaticTest** klasės kintamuosius

```
<access-specifier> class OuterClassName {
    public static class <StaticInnerClassName> {
        . . .
    }
    . . .
}
```

Main metode pabandyti sukurti abejų klasių egzempliorių

Vidinė klasė & vidinė statinė klasė

```
public class InnerClassTest {

    public class Inner1 {

    }

    static class Inner2 {

    }
}
```

```
class MainClient {
public static void main(String[] args) {
Inner1 i1 = new Inner1();
Inner2 i2 = new Inner2();

InnerClassTest container = new InnerClassTest();
Inner1 i1ok = container.new Inner1();

}
}
```

Vidinė klasė & vidinė statinė klasė

No enclosing instance of type InnerClassTest is accessible. Must qualify the allocation with an enclosing instance of type InnerClassTest (e.g. x.new A() where x is an instance of InnerClassTest).

at lt.codeacademy.sdudy.inner.MainClient.main(MainClient.java:8)

```
class MainClient {
public static void main(String[] args) {
Inner1 i1 = new Inner1();
Inner2 i2 = new Inner2();

InnerClassTest container = new
InnerClassTest();
Inner1 i1ok = container.new Inner1();

}
}
```

Lokali vidinė klasė

Sintaksė:

```
<access-specifier> class <OuterClassName> {
    code...
    <access-specifier> <return-type> <MethodName>(<argumer
        class <LocalInnerClassName>{
            code...
        }
        code...
    }
    code...
}
```

- Lokalios vidinės klasės savybės:
  - Lokaliai vidinei klasei nenurodomas prieigos tipas (t.į. public, protected ar private). Ji visada galioja tik tame bloke, kuriame yra apibrėžta.
  - Didelis lokalių klasių privalumas: jos visiškai nematomos išorinėms klasėms.
  - Lokalios klasės gali naudoti visus egzemplioriaus elemantus ir tuos **lokalius** metodo kintamuosius, kurie yra paskelbti **final**.

PVZ

```
package lt.codeacademy.sdudy.inner;
public class City {
	interface Tikslas {
		String readLabel();
	}
	public Tikslas dest(String s) {
		class MyLocalCity implements Tikslas {
			private String label;
			private MyLocalCity(String whereTo) {
				label = whereTo;
			}
			public String readLabel() {
				return "*" + label + "*";
			}
		}
		return new MyLocalCity(s);
	}
	public static void main(String[] args) {
		City p = new City();
		Tikslas v = p.dest("Vilnius");
		MyLocalCity v = p.dest("Kaunas");
	}}
```

Uzd 4 (lokali klasės)

- Public klasė Labas turi
  - kintamuosisu
    - private int sk = 10;
    - private String zodis = “žodis”
  - metodus:
    - labaDiena(); - metode apsirašykime lokalią klasę Diena, su metodu diena() - išspausdina: “diena, ” +sk, +zodis;. Sukurkime egzempliorių…. patestuokime
    - labasVakaras();- metode apsirašykime lokalią klasę Vakaras, su metodu diena(), - sukurkime egzempliorių…. patestuokime
- Main metode patestuokime

```
class Labas {
	private int sk = 10;
	private String zodis = "žodis";
	void labaDiena() {
		class Diena {
			void diena() {
				System.out.println("labaDiena() -> Diena: " + zodis);
			}
		}
		Diena d = new Diena();
		d.diena();
	}
	// ...
}
```

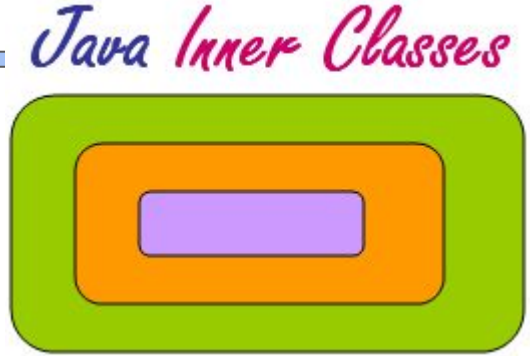
Uzd 5 (lokali klasės)

- Atsiranda public interfeisas IDiena su metodu void diena();
- Public klasė Labas turi
  - kintamuosisu
    - private int sk = 10;
    - private String zodis = “žodis”
  - Metodus, kurie grąžina IDiena interfeiso objektus:
    - labaDiena(); - metode apsirašykime lokalią klasę Diena, su metodu diena(), sukurkime egzempliorių ir grąžinti
    - labasVakaras();- metode apsirašykime lokalią klasę Vakaras, su metodu diena(), sukurkime egzempliorių ir grąžinti
- Main metode patestuoti

Labas

```
class Labas {
        private int sk = 10;
        private String zodis = "žodis";
        IDiena labaDiena() {
                class Diena implements IDiena {
                        public void diena() {
                          System.out.println("labaDiena() -> Diena: " + zodis);
                        }
                }
                Diena d = new Diena();
                return d;
        }
        // ...
}
interface IDiena {
        void diena();
}
```

Vidinės klasės



Vidines klases sukuria kompiliatorius, JVM vidinės klasės nieko nesiskiria nuo kitų klasių

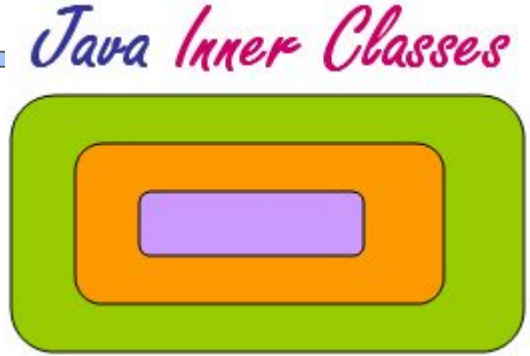
```
// VidinesDemo.java: vidinių klasių kūrimo pvz
public class Vidines{

   // Vidine klasė Test1
   class Test1 {}

   // Vidine klasė Test2
   class Test2 {}

   public static void main(String [] args) {
      // Anonymous vidinė klasė 1
      new Object() {};
      // Anonymous vidinė klasė 2
      new Object() {};
      System.out.println("Sveikas, Pasauli");
   }
}
```

Vidinės klasės



Vidines.java klasės
javac Vidines.java
Kompiliavimo rezultatas:

Vidines.class
Vidines$Test1.class
Vidines$Test2.class
Vidines$1.class
Vidines$2.class

Bevardė vidinė klasė

Sintaksė:

```
new SuperType(construction parameters) {
    inner class methods and data
}
```

Bevardės (*anonymous*) vidinės klasės savybės:

- SuperType gali būti interfeisas, pvz ActionListener. Jei taip, vidinė klasė realizuoją tą interfeisą. Jei SuperType yra klasė, bevardė vidinė klasė praplečia tą klasę.
- Bevardė klasė negali turėti konstruktorių, nes jos vardas yra tas pats kaip ir išorinės, kurioje ji sukurta, vardas.
- Rekomenduojama bevardes vidines klases naudoti saikingai, nes jos apsunkina kodo skaitymą.

Bevardė vidinė klasė

1. Object
2. Ne Object
3. Ne Object

```
class MainClient2 {
    public static void main(String[] args) {

        Object obj1 = new Object();
        Object obj2 = new Object() {
        };
        Object obj3 = new Object() {
            @Override
            public String toString() {
                // TODO Auto-generated method stub
                return super.toString();
            }
        };
        System.out.println(obj1.getClass());
        System.out.println(obj2.getClass());
        System.out.println(obj3.getClass());
    }
}
```

```
class java.lang.Object
class lt.codeacademy.sdudy.inner.MainClient2$1
class lt.codeacademy.sdudy.inner.MainClient2$2
```

Uzd 6 (bevardės klasės)

- Public interfeisas IDiena su metodu void diena();
- Public klasė Labas turi
  - kintamuosisu
    - private int sk = 10;
    - private String zodis = “žodis”
  - Metodus, kurie grąžina IDiena interfeiso objektus:
    - labaDiena(); - metode sukurti bevardę IDiena tipo klasė ir grąžinti
    - labasVakaras();- metode sukurti bevardę IDiena tipo klasė ir grąžinti
- Main metode patestuoti

Bevardė

```
class Labas {
        private int sk = 10;
        private String zodis = "žodis";
        IDiena labaDiena() {
                return new IDiena() {
                        @Override
                        public void diena() {
                                System.out.println("labaDiena() -> Diena: " + zodis);
                        }
                };
        }
        // ...
}
interface IDiena {
        void diena();
}
```

lembda...

Vidinės klasės

Vidines.java klasės
javac Vidines.java
Kompiliavimo rezultatas:

Vidines.class
Vidines$Test1.class
Vidines$Test2.class
Vidines$1.class
Vidines$2.class

Pakartokime:

- Vidinė klasė
- Statinė vidinė klasė
- Lokali vidinė klasė
- Bevardė vidinė klasė

Uzduotis 7

1. Susikurkite klasę A, kurioje yra vidinė klasė B, o klasėje B – vidinė klasė C.
   a. Be to, klasė A turi sveiką privatų kintamąjį i (jį inicializuoti deklaravimo metu), privatų metodą getI() be parametrų, kuris grąžina sveiką reikšmę, ir metodą main.
   b. Klasė C turi sveiką privatų kintamąjį k (jį inicializuoti deklaravimo metu), privatų metodą getK() be parametrų, kuris grąžina sveiką reikšmę, ir viešą metodą s be parametrų, kuris nieko negrąžina, tačiau atspausdina klasės A kintamojo i reikšmę ir metodo getI() grąžintą reikšmę.
   c. Metode main sukurkite klasės C objektą. Iškviečiant jo metodą s įsitikinkite, kad vidinei klasei yra pasiekiami išorinės klasės privatūs laukai ir metodai.
   d. Patikrinkite, ar iš metodo main galima atspausdinti klasės C kintamojo k reikšmę ir metodo getK() grąžintą reikšmę.

A B C

```
class A {
    class B {
        class C {

        }
    }
}
```

A B C

Susikurkite klasę A, kurioje yra vidinė klasė B, o klasėje B – vidinė klasė C.

a. Be to, klasė A turi sveiką privatų kintamąjį i (jį inicializuoti deklaravimo metu), privatų metodą getI() be parametrų, kuris grąžina sveiką reikšmę, ir metodą main.

A B C

Susikurkite klasę A, kurioje yra vidinė klasė B, o klasėje B – vidinė klasė C.

a. Be to, klasė A turi sveiką privatų kintamąjį i (jį inicializuoti deklaravimo metu), privatų metodą getI() be parametrų, kuris grąžina sveiką reikšmę, ir metodą main.

```
class A {
    private int i = 10;

    private int getI() {
        return i;
    }

    class B {

        class C {

        }
    }
```

A B C

b. Klasė C turi sveiką privatų kintamąjį k (jį inicializuoti deklaravimo metu), privatų metodą getK() be parametrų, kuris grąžina sveiką reikšmę, ir viešą metodą s be parametrų, kuris nieko negrąžina, tačiau atspausdina klasės A kintamojo i reikšmę ir metodo getI() grąžintą reikšmę

A B C

b. Klasė C turi sveiką privatų kintamąjį k (jį inicializuoti deklaravimo metu), privatų metodą getK() be parametrų, kuris grąžina sveiką reikšmę, ir viešą metodą s be parametrų, kuris nieko negrąžina, tačiau atspausdina klasės A kintamojo i reikšmę ir metodo getI() grąžintą reikšmę

```
class A {
    private int i;

    private int getI() {
        return i;
    }
    class B {

        class C {
          private int k = 100;

          private int getK() {
            return k;
          }

          public void s() {
          System.out.println("A.i=" + i);
          System.out.println("A.getI()=" + getI());
          }
...
```

A B C

b. Klasė C turi sveiką privatų kintamąjį k (jį inicializuoti deklaravimo metu), privatų metodą getK() be parametrų, kuris grąžina sveiką reikšmę, ir viešą metodą s be parametrų, kuris nieko negrąžina, tačiau atspausdina klasės A kintamojo i reikšmę ir metodo getI() grąžintą reikšmę

pakeisti

```
class A {
    private int i = 5;

    private int getI() {
        return i;
    }
    class B {

        class C {
          private int i = 100;

          private int getK() {
            return k;
          }

          public void s() {
          System.out.println("A.i=" + ???);
          System.out.println("C.i=" + ???);
          System.out.println("C.getK=" + getK());
          }
...
```

A B C

c. Metode main sukurkite klasės C objektą. Iškviečiant jo metodą s įsitikinkite, kad vidinei klasei yra pasiekiami išorinės klasės privatūs laukai ir metodai.

```
public static void main(String[] args) {
        A a = ...
}
```

A B C

c. Metode main sukurkite klasės C objektą. Iškviečiant jo metodą s įsitikinkite, kad vidinei klasei yra pasiekiami išorinės klasės privatūs laukai ir metodai.

```
public static void main(String[] args) {
        A a = new A();
        B b = a.new B();
        B.C c = b.new C();
        c.s();
}
```

A B C

c. Metode main sukurkite klasės C objektą. Iškviečiant jo metodą s įsitikinkite, kad vidinei klasei yra pasiekiami išorinės klasės privatūs laukai ir metodai.

```
public static void main(String[] args) {
        A a = new A();
        B b = a.new B();
        B.C c = b.new C();
        c.s();
}
```

A.i=10
A.getI()=10

A B C

d. Patikrinkite, ar iš metodo main galima atspausdinti klasės C kintamojo k reikšmę ir metodo getK() grąžintą reikšmę.

```
class A {
        private int i = 10;

        private int getI() {
                return i;
        }
        class B {
                class C {
                        private int k = 100;

                        private int getK() {
                                return k;
                        }
                        public void s() {
                                System.out.println("A.i=" + i);
                                System.out.println("A.getI()=" + getI());
                        }

                }
        }
        public static void main(String[] args) {
                A a = new A();
                B b = a.new B();
                B.C c = b.new C();
                c.s();
        }
}
```

TODO

“lembda”