Klasės narių pasiekiamumas (panaudojimas) intro...

Mindaugas Karpinskas
2017

Trumai

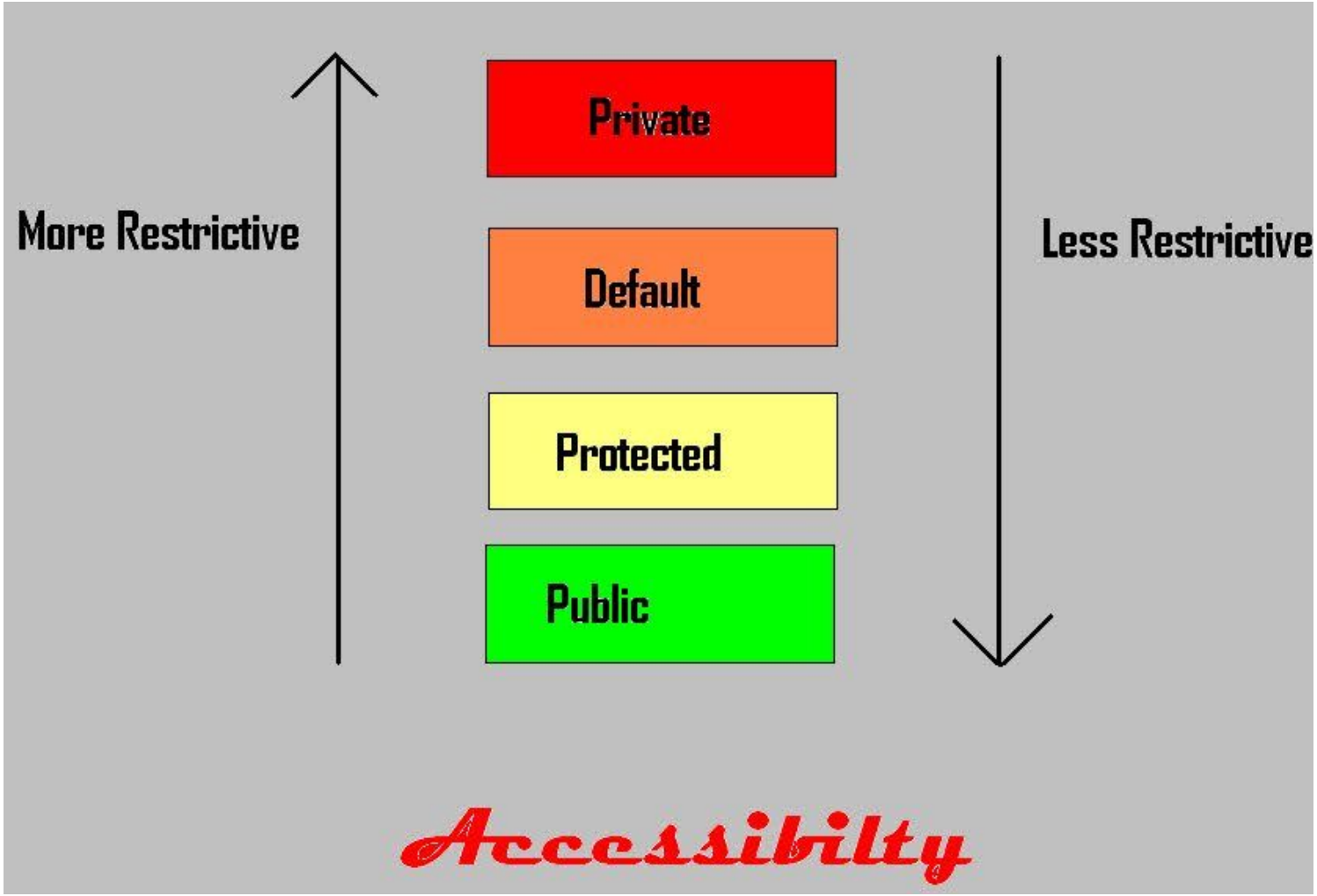
- Klasės narių panaudojimas
  a. **private**,
  b. (nutylimas),
  c. **protected**,
  d. public
- Links:
  a. http://www.java-made-easy.com/java-access-modifiers.html
  b. https://beginnersbook.com/2013/05/java-access-modifiers/
  c. https://docs.oracle.com/javase/tutorial/java/javaOO/accesscontrol.html
  d. https://stackoverflow.com/questions/215497/in-java-difference-between-default-public-protected-and-private
  e. https://www.javatpoint.com/access-modifiers

Klasės narių panaudojimas

Java kalboje galima kontroliuoti klasės narių panaudojimą. Public (iki šiol dažniausiai naudotas) modifikatorius pažymi, kad klasės metodas ar kintamasis yra "demokratiškas" ir juo galima naudotis toje pačioje ir kitose klasėse laisvai.

public alternatyvos: private, (Nutylimas), protected

Modifikatorius	**Aprašymas**
Private	Narys pasiekiamas tik klasės viduje.
(Nutylimas)	Narys pasiekiamas toms paketo klasėms, kurios nepaveldi šios klasės.
Protected	Narys pasiekiamas paketo klasėms.
Public	Narys pasiekiamas kitoms klasėms.


More Restrictive
Private
Default
Protected
Public
Less Restrictive
Accessibilty

Klasės narių pasiekiamumas (panaudojimas)

- Panaudojimo modifikatorius leidžia programuotojui apsaugoti savo klasę nuo vartotojo neapgalvotų veiksmų.
- Gerai apgalvoti modifikatorių reikšmes naudinga ir pačiam programuotojui, nes grįžus po ilgesnės pertraukos prie klasės redagavimo užsimiršta detalės ir galima padaryti sunkiai aptinkamą klaidą neatsargiai pakeitus kokio nors kintamojo reikšmę.
- *Antra vertus **private** modifikatoriumi neverta piktnaudžiauti, nes jis mažina jūsų klasės panaudojimo lankstumą.*
- Tokia metodų ir kintamųjų panaudojimo valdymo koncepcija vadinama "duomenų maskavimu" (angl. data hiding).

Private. Pvz.: privatūs ir viešieji klasių nariai.

Iliustruokime duomenų panaudojimo valdymą paprastu pavyzdžiu. Tarkime klasė skirta aprašyti, kiek parduotuvė surinko pinigų ir kiek buvo atėję klientų. Aišku šie kintamieji saugumo prasme yra gana svarbūs ir juos aprašydami išnaudokime Java kalbos teikiamas "duomenų maskavimo" galimybes

```
public class SaleProcessor {
   private int revenue=0;
   private int numSales=0;
   public void recordSale(int newRevenue){
      revenue = revenue + newRevenue;
      numSales = numSales + 1;
}
   public int getRevenue() {
      return revenue;
}
   public int getNumSales() {
      return numSales;
}}
```

Kiekvieno pirkimo metu bus iškviestas ringUpCustomer metodas, kuris nurodytu kiekiu padidins surinktų pinigų sumą ir vienetu padidins parduotuvės klientų skaitiklį. Revenue ir numSales kintamųjų private modifikatorius apsaugoja nuo nesankcionuoto šių kintamųjų pakeitimo.

SaleProcessor

```
public class SaleProcessor {
   private int revenue=0;
   private int numSales=0;
   public void recordSale(int newRevenue){
      revenue = revenue + newRevenue;
      numSales = numSales + 1;
}
   public int getRevenue() {
      return revenue;
}
   public int getNumSales() {
      return numSales;
}
}
```

Panaudojimo taisyklės

- **private** modifikatorius.
  - Šis modifikatorius leidžia naudoti kintamąjį ar klasę tik tos klasės viduje.
  -

Duomenų apsauga

- **private** panaudojimo modifakatorius
  - Šis modifikatorius apsaugo kintamąjį nuo kitų klasių.
  - Jei kintamasis yra **private**, jį gali pakeisti tik **tos pačios klasės nariai** (kintamieji vidines nestatines klasės, konstruktoriai)

```
class A {
    private int i = 10;
}
class B {
    private A a = new A();
    int get() {
        return a.i;
    }
}
```

```
error: incompatible types: unexpected return value
        return a.i;
                 ^
1 error
```

Error: The field A.i is not visible

Duomenų apgauba

Duomenų apsaugojimo mechanizmas remiasi hierarchija tarp kintamųjų panaudojimo. Keičiant apsaugotus kintamuosius tenka kreiptis į specialiai tuo tikslu parašytą klasės metodą. Tokia kintamųjų apsauga vadinama **duomenų apgauba** (angl. **encapsulation**).

***Encapsulation Java supports access modifiers to protect data from unintended access and modifi cation. Most people consider encapsulation to be an aspect of object-oriented languages (OCA: Oracle® Certified Associate Java® SE 8 Programmer I Study Guide Exam 1Z0-808).

Užduotis1 Panaudojamuimas

Sukurkime klasę:

class A{

}

Klasėje A apsirašykime du private kintamuosius, ir du metodus kurie keičia laukų reikšmes.

Kitos klasės main metode susikurti A klasės objektų ir pabandyti pasiekti kintamuosius ir metodus.

Duomenų apsauga

- **private** panaudojimo modifakatorius
  - Šis modifikatorius apsaugo kintamąjį nuo kitų klasių.
  - Jei kintamasis yra **private**, jį gali pakeisti tik **tos pačios klasės nariai** (kintamieji vidines nestatines klasės, konstruktoriai)

```
class A {
    private int i = 10;
}
class B {
    private A a = new A();
    int get() {
        return a.i;
    }
}
```

```
error: incompatible types: unexpected return value
        return a.i;
                 ^
1 error
```

Error: The field A.i is not visible

Protected modifikatorius

`protected` modifikatorius suteikia teisę naudotis kintamuoju arba metodu tik toms klasėms, kurios yra tame pačiame pakete. Kokiam paketui yra priskiriama klasė yra paskelbiama jos aprašymo pradžioje:

```
package eu.somePackage;
```

Protected. PVZ

```java
package lt.codeacademy.sdudy.protect.one;

public class TestFields {

	public static void main(String[] args) {
		Field test = new Field();
		System.out.println(test.age);
	}

}
```

```java
package lt.codeacademy.sdudy.protect.one;

public class Field {
	protected int age = 100;
}
```

> 100

TestFields klasė gali pasiekti **age** kintamąjį.

Protected. PVZ

Skirtingi paketai

```
package lt.codeacademy.one;

import lt.codeacademy.two.Field;

public class TestFields {

     public static void main(String[] a) {
          Field test = new Field();
          System.out.println(test.age);
     }
}
```

```
package lt.codeacademy.two;

public class Field {
     protected int age = 100;
}
```

```
Exception in thread "main" java.lang.Error: Unresolved compilation problem:
        The field age is not visible

        at ...
```

TestFields klasė negali pasiekti **age** kintamojo.

Atsirado import eilutė, nes klasė kitame pakete

Protected. PVZ paveldėjimas

```
package lt.codeacademy.sdudy.protect.one;

import lt.codeacademy.two.Field;
public class TestFields extends Field {
     public static void main(String[]
args) {
          TestFields test =
                         new TestFields();
          test.spausdink();
     }
     public void spausdink() {
          System.out.println(super.age);
     }
}
```

```
package lt.codeacademy.sdudy.protect.one;

public class Field {
     protected int age = 100;
}
```

```
> 100
```

TestFields klasė gali pasiekti **age** kintamąjį.

Užduotis2 Protected

Susikurti tris klases viename pakete (PVZ: Masina, Ratas, Variklis). Kiekviena klasė turi po du **protected** laukus ir po du **protected** metodus.

1. Pabandyti pirmoje klasės susikurti kitos klasės objektą/-ų ir pabandyti pasiekti metodus, kintamuosius.
2. Klases: Ratas, Variklis, - perkelkime į kitą paketą

Eclipse demo

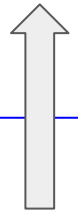
Nutylimas panaudojimo modifikatorius

Jei jūs nenurodote tiesiogiai metodo ar kintamojo panaudojimo modifikatoriaus, tai kompiliatorius leidžia juos naudoti einamojo katalogo klasių nariams. Tačiau jei vėliau nuspręsite įtraukti klasę į paketą, nutruks ryšys tarp einamojo katalogo klasių narių ir atsiras programos derinimo sunkumų. Todėl rekomenduojama vengti nutylimojo modifikatoriaus ir nuspręsti, koks panaudojimo modifikatorius geriausiai jums tiktų.

Nutilymas. PVZ

```
package lt.codeacademy.sdudy.protect.one;

public class TestFields {

     public static void main(String[] args)
{
          Field test = new Field();
          System.out.println(test.age);
     }

}
```

```
package lt.codeacademy.sdudy.protect.one;

public class Field {
     int age = 100;
}
```



TestFields klasė gali pasiekti **age** kintamąjį.

> 100

Nutilymas. PVZ

Skirtingi paketai

```
package lt.codeacademy.one;

import lt.codeacademy.two.Field;

public class TestFields {

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

```
package lt.codeacademy.two;

public class Field {
     int age = 100;
}
```

```
Exception in thread "main" java.lang.Error: Unresolved compilation problem:
        The field Field.age is not visible

        at lt.codeacademy.sdudy.protect.one.TestFields.main(TestFields.java:9)
```

TestFields klasė negali pasiekti **age** kintamojo.

Atsirado import eilutė, nes klasė kitame pakete

Nutilymas. PVZ paveldėjimas

```
package lt.codeacademy.sdudy.protect.one;

import lt.codeacademy.two.Field;
public class TestFields extends Field {
     public static void main(String[] args) {
          TestFields test = new TestFields();
          test.spausdink();
     }
     public void spausdink() {
          System.out.println(super.age);
     }
}
```

```
package lt.codeacademy.sdudy.protect.one;

public class Field {
     int age = 100;
}
```

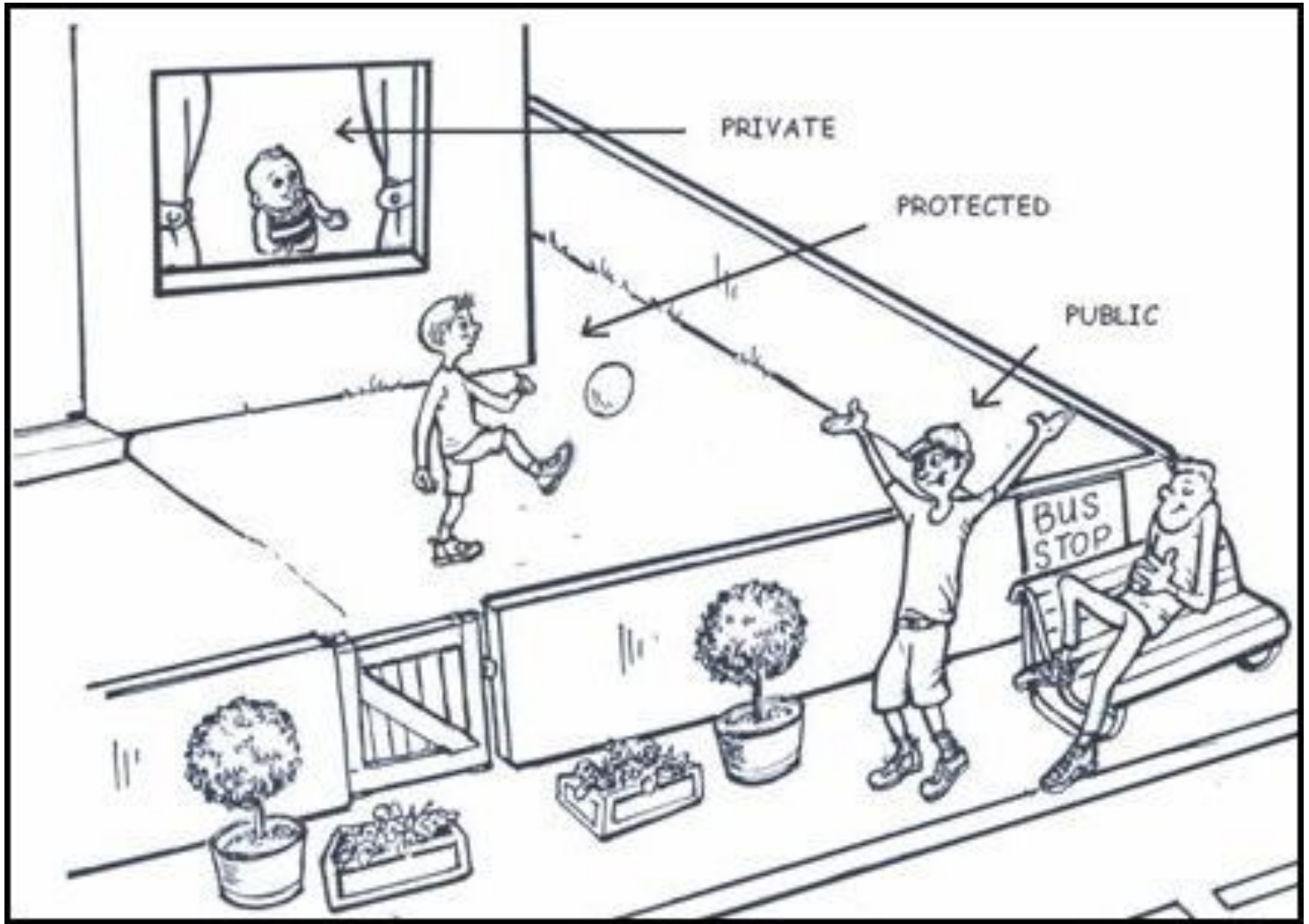
```
Exception in thread "main" java.lang.Error: Unresolved compilation
problem:
     The field Field.age is not visible

     at
lt.codeacademy.sdudy.protect.one.TestFields.spausdink(TestFields.ja
va:13)
     at
lt.codeacademy.sdudy.protect.one.TestFields.main(TestFields.java:9
)
```

TestFields klasė NEgali pasiekti **age** kintamąjį.

Panaudojimas

Modifikatorius	Tos pačios klasės nariai	To paties paketo klasės	To paties paketo poklasės nariai	Poklasės nariai	Visos kitos klasės
private	Taip	Ne	Ne	Ne	Ne
	Taip	Taip	Taip	Ne	Ne
protected	Taip	Taip	Taip	Taip	Ne
public	Taip	Taip	Taip	Taip	Taip

Įsiminti!


PRIVATE
PROTECTED
PUBLIC
BUS STOP

Užduotis3 Panaudojamuimas

1. Sukurti klasę “Staciakampis”, kur būtų galima keisti jo ilgį, plotį tik per metodą nustatytiSkaciakampioDydi(a, b)
   a. Klasė turi turėti viešus metodus kurie, paskaičiuoja jo plotą ir perimetrą
2. main metode patestuoti...

Staciakampis

```
public class StaciakampisRun {
	public static void main(String[] args) {
		Staciakampis s = new Staciakampis();
		s.keisti(15, 14);
		System.out.println(s.plotas());
		System.out.println(s.perimetras());
	}
}
class Staciakampis {
	private int a, b;
	public void keisti(int a, int b) {
		this.a = a;
		this.b = b;
	}
	public int plotas() {
		return this.a * this.b;
	}
	public int perimetras() {
		return this.a * 2 + this.b * 2;
	}
}
```

Užduotis4 Panaudojamuimas

- Sukurti klasę, kuri atspausdiną konsolėje tusčias eilutes.
- nurodome kiek eiličių reikės atspausdinti konstruktoriuje ir užtikriname, kad niekas nepakeistu to skaičiaus reikšmės
- metodas, kuris atspausdina prieš tai nurodyta eilučių skaičių

main metode patestuoti

PVZ PrintLines

- Kintamasis negali turėti neigiamos reikšmės.
- Tik klasės metodai gali keisti apgaubtą kintamąjį ir belieka tik parašyti metodus taip, kad jie neleistų kintamajam tapti neigiamu sveikuoju skaičiumi.

```
class PrintLines {
	private int linesToPrint = 0;

	public void printSomeLines() {
		for (int i = 0; i <= linesToPrint; i++) {
			System.out.println("");
		}
	}
	public void setLinesToPrint(int j) {
		if (j > 0) {
			linesToPrint = j;
		}
	}
}
```

Analizuojant programą matyti, kad tik setLinesToPrint metodas gali keisti linesToPrint kintamojo reikšmę. Šis metodas ir pasirūpina, kad neigiamos reikšmės nebūtų priskiriamos.

Užduotis5 Panaudojamuimas

- Sukurti klasę, kuri turi metoda labas();
- Metodas išveda tekstą:
  - Labas, Pasauli!
- Metodas turi būti pasiekiamas tik to paties paketo klasėms.

1. Sukurkite kitą klasę su main metodu tam pačiam pakete ir pabandykite iškviesti anksčiau apsiraštos kalsės metodą labas();
2. Sukurti kitą klasę, kitame pakete ir pabandyti iškviesti labas() metodą

Klausimai

Next

Prepare maven

Prepare Finance app