Ukrywanie implementacji[i]

Hermatyzacja to proces (postępowanie) mające na celu „oddzielenie rzeczy, które się zmieniają od rzeczy, które pozostają takie same”. Język Java dostarcza modyfikatory dostępu. Poziom dostępu to:

- public,
- protected,
- *“przyjazny”* (bez określonego słowa kluczowego),
- private.

Modyfikatory dostępu umieszcza się przed definicją składnika klasy. Modyfikator należy ustawić przed każdym składnikiem klasy osobno.

Dostęp „przyjazny”

Domyślny stopień dostępu to dostęp przyjazny. Nie posiada on słowa kluczowego. Nazywany również dostępem pakietowym.

Dostęp „przyjazny” pozwala na grupowanie spokrewnionych klas w pakiety. Mogą one więc ze sobą współdziałać.

Klasa ma kontrolę nad tym, jaki kod ma dostęp do jej składników. Kod z innego pakietu nie ma dostępu do takich klas.

Aby kod z innego pakietu otrzymał dostęp do składnika należy:

- uczynić go składnikiem publicznym,
- w klasie innego pakietu dokonać stosownego importu.

Klasa pochodna ma dostęp do składników przyjaznych tylko wówczas gdy znajduje się w tym samym pakiecie.

Dostęp publiczny

Public powoduje, że składnik staje się dostępny dla każdego z dowolnego miejsca. W danej jednostce kompilacji (w pliku) może znajdować się tylko jedna klasa publiczna.

```
package dessert;
public class Cookie{

public Cookie(){
              System.out.println("Konstruktor klasy Cookie");
}

void bite(){
          System.out.println("metoda bite");
  }
}
```

```
import dessert.*;
public class Dinner{

public Dinner(){
               System.out.println("Konstruktor klasy Dinner");
     }

public static void main(String[] args){
               Cookie x = new Cookie();
               //! x.bite(); // Brak dostępu
     }
}
```

Dostęp prywatny

Private oznacza, że do danego składnika klasy nie ma dostępu nikt poza jego własną klasą.

```
class Sundae{

private Sundae() {}

static Sundae makeASundae(){
                         return new Sundae();
          }
}

public class IceCream{

public static void main(String[] args){
     //! Sundae x = new Sundae();
     Sundae x = Sundae.makeASundae();
     }
}
```

Modyfikator private umożliwia kontrolowanie sposobu tworzenia obiektu. W ten sam sposób możemy uczynić prywatną każdą metodę w danej klasie.

Dostęp chroniony

Protected jest związane z koncepcją dziedziczenia. W wyniku dziedziczenia mamy dostęp tylko do składników publicznych klasy bazowej. Modyfikator protected umożliwia dostęp do danego składnika klasy tylko klasie dziedziczącej (klasie potomnej) i w klasach należących do tego samego pakietu.

Jeżeli tworzymy nowy pakiet i dziedziczymy po klasie z innego pakietu, wtedy jedynymi składowymi, do których mamy dostęp, są składowe publiczne i chronione. Poprzez dziedziczenie nie uzyskujemy dostępu do składowych prywatnych i "przyjaznych".

```
public class ChocolateChip extends Cookie{

public ChocolateChip(){
               System.out.println("Konstruktor klasy ChocolateChip");
}

public static void main(String[] args) {
ChocolateChip x = new ChocolateChip();
//! x.bite(); // Nie ma dostępu do bite
}
}
```

Jak zmodyfikować klasę Cookie, aby miała ona dostęp do metody bite()?

```
public class Cookie{

public Cookie(){
               System.out.println( "Konstruktor klasy Cookie");
     }
protected void bite(){
                    System.out.println("metoda bite");
     }
}
```

Dostęp do klas

Język Java umożliwia również określenie, które klasy mogą być dostępne dla klas z innych pakietów. Aby klasa była dostępna wówczas należy umieścić specyfikator dostępu publicznego przed jej definicją:

```
public class Testowa
```

Klasy nie mogą być ani typu private, ani typu protected (wyjątkiem klasy wewnętrzne). Aby uniemożliwić dostęp do danej klasy, należy jej wszystkie konstruktory uczynić prywatnymi (lub zależnie od potrzeby określić dostęp na "przyjazny" lub chroniony).

```
class Soup{
     private static Soup ps1 = new Soup();
     private Soup() {}
//(1) Umożliwienie tworzenia obiektu poprzez metodę statyczną:
public static Soup makeSoup(){
     return new Soup();
}
//zwracanie na żądanie referencji do statycznego obiektu (stworzonego wcześniej)
public static Soup access(){
     return ps1;
}
public void f() {}
}

class Sandwich //Wykorzystuje klasę Lunch
{
     void f(){
               new Lunch();
     }
}
//W pliku możliwa jest tylko jedna klasa publiczna:
public class Lunch{
               void test()
{
//Tego nie wolno! Konstruktor jest prywatny:
//Soup priv1 = new Soup();
Soup priv2 = Soup.makeSoup();
Sandwich f1 = new Sandwich();
Soup.access().f();
}
}
```

Metoda access() zwraca referencję do obiektu klasy Soup. Klasa Soup pokazuje jak uniknąć jawnego tworzenia obiektu – wszystkie konstruktory są prywatne.

✸ Modyfikatory dostępu - zestawienie

Tabela pokazuje poziomy dostępu określane przez każdy modyfikatorów:

Modyfikator	klasa	podklasa	pakiet	wszędzie
private	X			
protected	X	X	X	
public	X	X	X	X
"package"	X		X	

[i] *"Aspekty obiektowości" J.Wiślicki*