Z tej lekcji dowiesz się

- co to są specyfikatory dostępu (access modifiers)

- poznasz 4 rodzaje specyfikatorów:

  - publiczny
  - prywatny
  - chroniony
  - pakietowy

Specyfikatory dostępu pełnią bardzo ważną rolę w obiektowym programowaniu w języku Java.Pozwalają one na określenie praw dostępu do składowych klas oraz do samych klas. Spróbuję CI wytłumaczyć na czym polega rola tych specyfikatorów i po co one są odpowiedzialne.

Aby to zrozumieć posłużę się przykładem samochodu. Jeśli przyjrzymy się większości nowych samochodów (czy każdemu innemu złożonemu urządzeniu: pralka, telewizor, telefon komórkowy, pilot TV itd.) to zauważymy jedną cechę szczególną: użytkownik ma dostęp do pewnych funkcji urządzenia poprzez udostępnione dla niego przyciski, pokrętła itd., jednak nie może on wpływać na ustawione przez producenta właściwości urządzenia, do których z reguły ma serwisant. Dzieje się tak dlatego, że do poprawnego użytkowania samochodu zwykły użytkownik nie potrzebuje dostępu do zaawansowanych funkcji oraz np. interfejsów diagnostycznych. Pracownik warsztatu za to może się podłączyć swoim komputerem do komputera pokładowego samochodu, dzięki czemu przeprowadzi diagnostykę. Czyli dostęp do różnych obszarów działania samochodu nie jest jednakowy. Występuje tu rozgraniczenie na dostęp dla każdego, jak również dostęp dla zaawansowanych użytkowników. Dlaczego tak się

dzieje? W przypadku samochodów - z prostej przyczyny: przeciętnemu użytkownikowi nigdy nie przydaje się dostęp do narzędzi diagnostycznych i nawet nie musi o nim wiedzieć. Co więcej, udostępnienie zwykłemu kierowcy wszystkich możliwych interfejsów oraz opcji spowodowałoby wrażenie, że samochód jest bardzo skomplikowany (bo w istocie jest) co negatywnie wpłynęłoby na komfort jego użytkowania. My potrzebujemy takiego interfejsu, który pozwoli nam w prosty i niezawodny sposób uruchomić samochód, zahamować, skręcać, włączać kierunkowskaz itd. Co więcej, bezpośredni dostęp do zaawansowanych ustawień poszczególnych podzespołów umożliwiłby zepsucie samochodu przez użytkownika. Znacznie trudniej zepsuć samochód używając tylko tych manipulatorów, które są udostępnione dla kierowcy. No i trzecia przyczyna: modyfikacje. Samochód można zmodyfikować wewnątrz (np. przerobić instalacje, jak w samochodach rajdowych lub np. dodać instalację gazową) nie zmieniając lub niewiele zmieniając to, co widzi kierowca. Czyli zasada działania pod maską samochodu zmienia ię diametralnie, natomiast kierowca nadal w sposób standardowy używa samochodu.

Bardzo analogicznie jest w przypadku programów w Javie: programista pisząc jakąś klasę nie musi odkrywać sposobu jej działania ani odsłaniać wszystkich jej mechanizmów dla innych klas czy użytkowników. To, co jest udostępnione "na zewnątrz", tzw. interfejs klasy (odpowiednik kokpitu kierowcy) wystarczy, aby w pełni wykorzystać możliwości klasy, ale bez możliwości grzebania w jej wnętrzu. Pisząc klasę udostępniamy na zewnątrz tylko to, co jest potrzebne do jej używania, a nie: zepsucia jej.

W języku java występuje następująca gradacja dostępu do obszarów klas (jak również samych klas):

- public (dostęp publiczny)
- private (dostęp prywatny)
- protected (dostęp chroniony)
- pakietowy

Dostęp publiczny: **public**

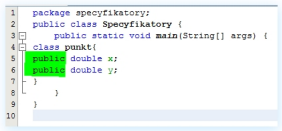
Jeśli ustawimy dostęp do klasy na publiczny, to dostęp do niej będą miały wszystkie inne klasy - nie będzie on ograniczony.

Zobaczmy to na przykładzie:

Zad. 1 Napisz klasę punkt, której pola: x oraz y będą miały dostęp publiczny.

Stwórz nowy projekt (Java Application) o nazwie **punkty**.

Oto kod naszej klasy.



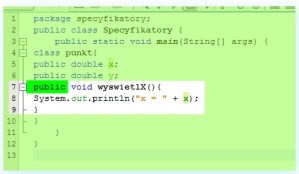
Jak widzimy, **specyfikator występuje na samym początku**, przed typem zmiennej"

public *double x;*

public *double y;*

Podobnie rzecz ma się z metodami:

Zad. 2: Zmodyfikuj powyższy program tak, aby dodać publiczną metodę *wyswietlX.*



Jak widać, specyfikator public występuje również na początku przed typem metody:

public *void wyswietlX(){*

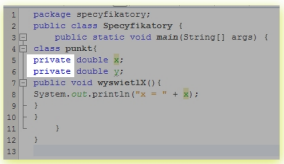
//kod metody

}

Dostęp prywatny - **private**

Dostęp prywatny to taki, gdzie dostęp do obiektu mają tylko składowe tej samej klasy. Oznacza to, że nie ma dostępu do obiektu dla innych klas oraz dla składowych tych innych klas. (odpowiednik elektroniki wewnętrznej w samochodzie, do której nie każdy ma dostęp). Zobaczmy to na przykładzie:

Zad. 3: Zmodyfikuj powyższy program tak, aby pola X oraz Y były polami prywatnymi (bez dostępu do nich z zewnątrz klasy).



```
package specyfikatory;
public class Specyfikatory {
    public static void main(String[] args) {
class punkt{
private double x;
private double y;
public void wyswietlX(){
System.out.println("x = " + x);
}
}
    }
}
```

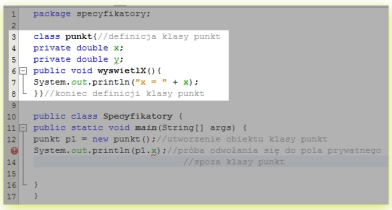
To było chyba najłatwiejsze zadanie w całym kursie - trzeba było zmienić z public na private w dwóch miejscach.

Jaki jest efekt tego zabiegu? Teoretycznie z zewnątrz klasy nie można mieć bezpośrednio dostępu do pól x oraz y. Sprawdźmy to:

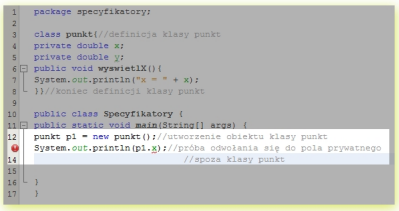
Zad. 4: Zmodyfikuj powyższy program tak, aby odwołać się do pól *x* oraz *y* spoza klasy.

Aby poprawnie napisać program musimy go troszkę przebudować:

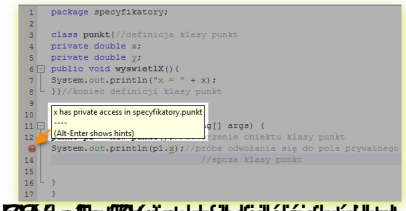
1. Klasę punkt przenosimy poza klasę *main oraz klasę Specyfikatory* Znajdzie sie ona więc na samym początku pliku, zaraz po nazwie pakietu:

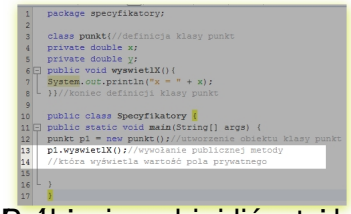


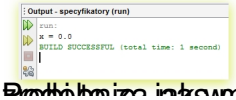
2. Następnie utworzymy obiekt klasy punkt i spróbujemy się odnieść do pola x tego obiektu:

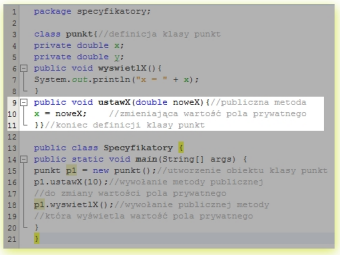


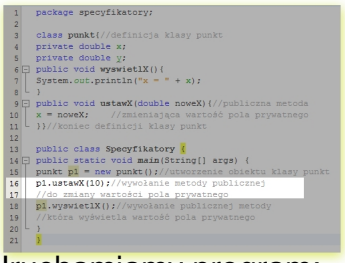
3. Część naszej instrukcji jest podkreślona na czerwono, na marginesie pojawił się wykrzyknik na czerwonym tle. oto, jaka informację o błędzie pokazuje nam kompilator:



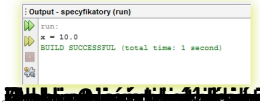


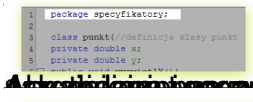


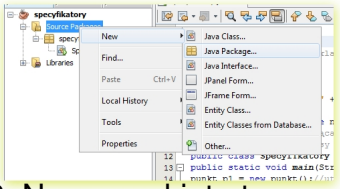




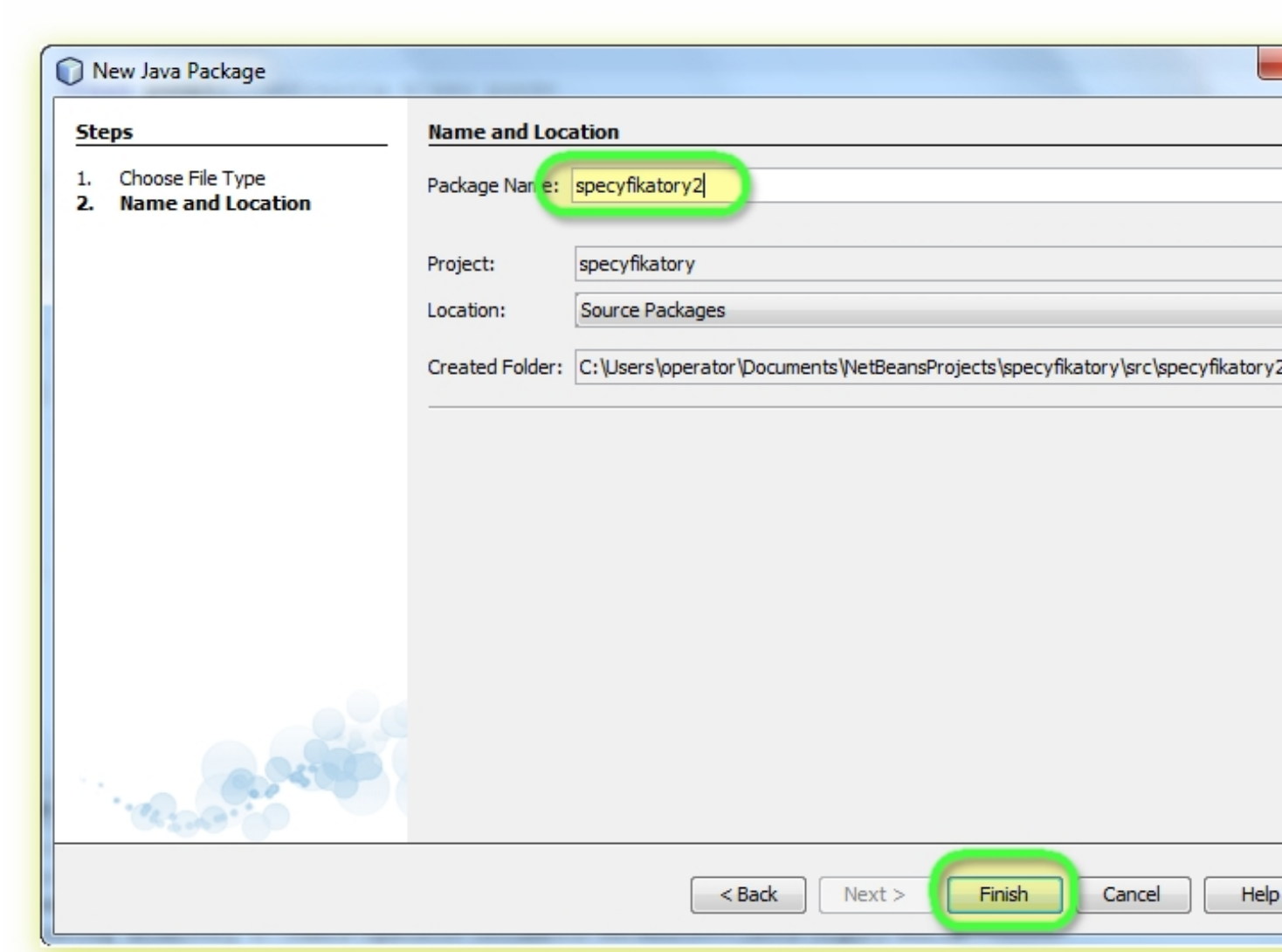
Uruchamiamy program:



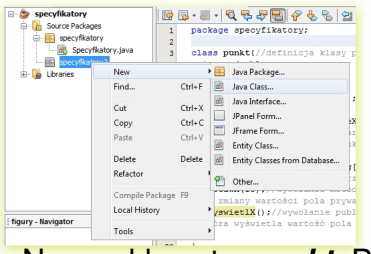




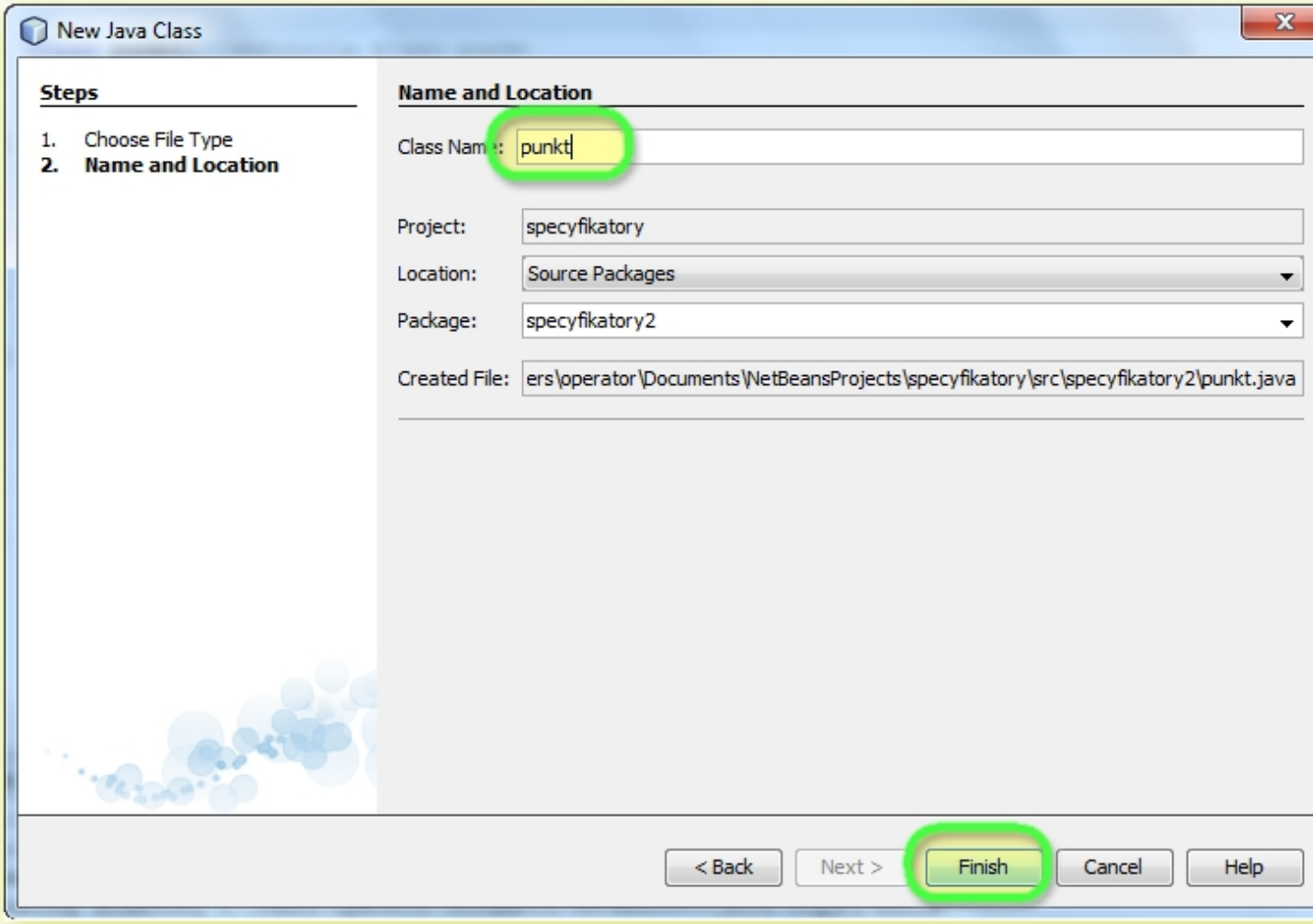
2. Nazwa pakietu to **specyfikatory2**. Po jej wpisaniu klikamy na **Finish**.



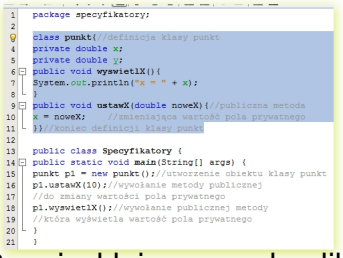
Na koniec obrobię nowego pakietu stworzymy nową klasę. W tym celu klikamy kolejno tak, jak



4. Nazwa klasy to ***punkt***. Po jej wpisaniu klikamy na ***Finish***.

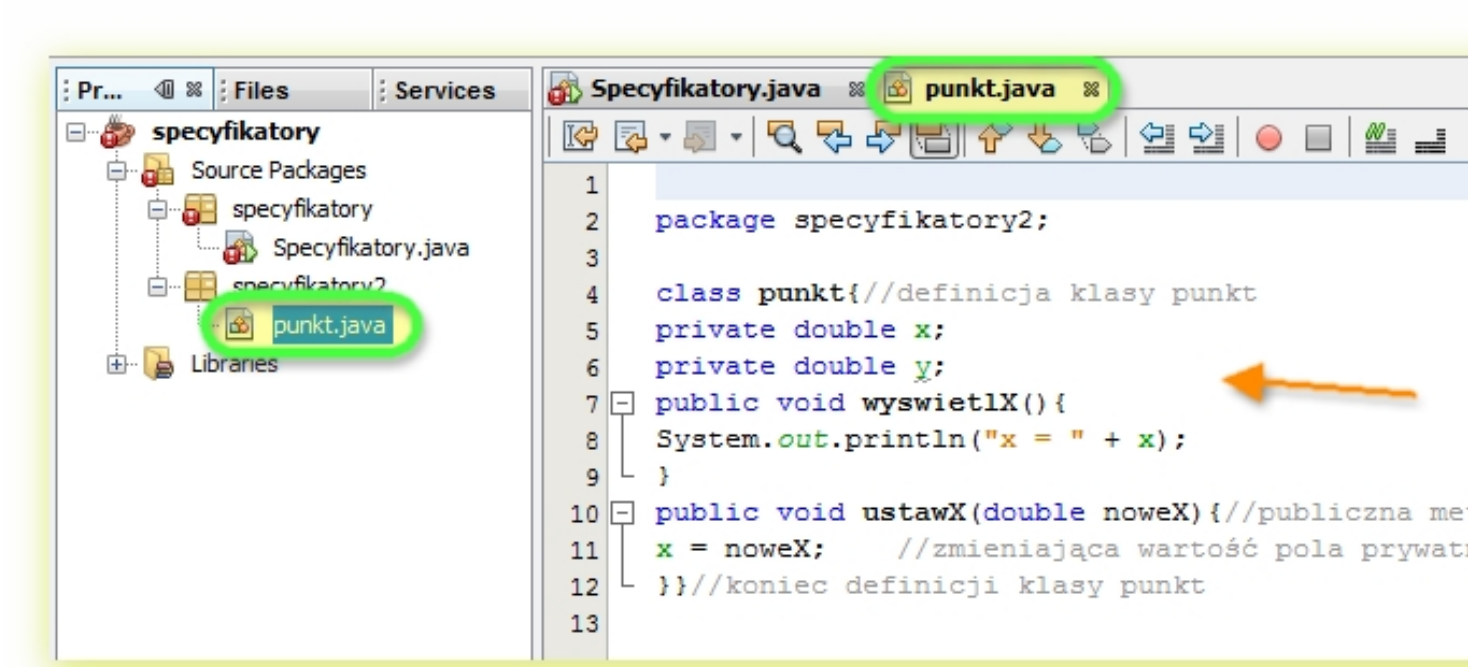


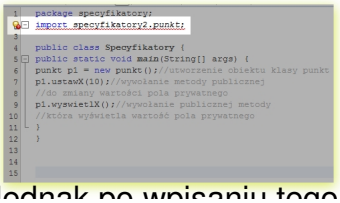
5. Z naszego pliku z programem wycinamy cały kod klasy ***punkt*** ...



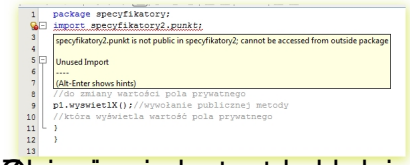
```
package specyfikatory;

class punkt{//definicja klasy punkt
private double x;
private double y;
public void wyswietlX(){
System.out.println("x = " + x);
}
public void ustawX(double noweX){//publiczna metoda
x = noweX;    //zmieniająca wartość pola prywatnego
}//koniec definicji klasy punkt

public class Specyfikatory {
public static void main(String[] args) {
punkt p1 = new punkt();//utworzenie obiektu klasy punkt
p1.ustawX(10);//wywołanie metody publicznej
//do zmiany wartości pola prywatnego
p1.wyswietlX();//wywołanie publicznej metody
//która wyświetla wartość pola prywatnego
}
}
```

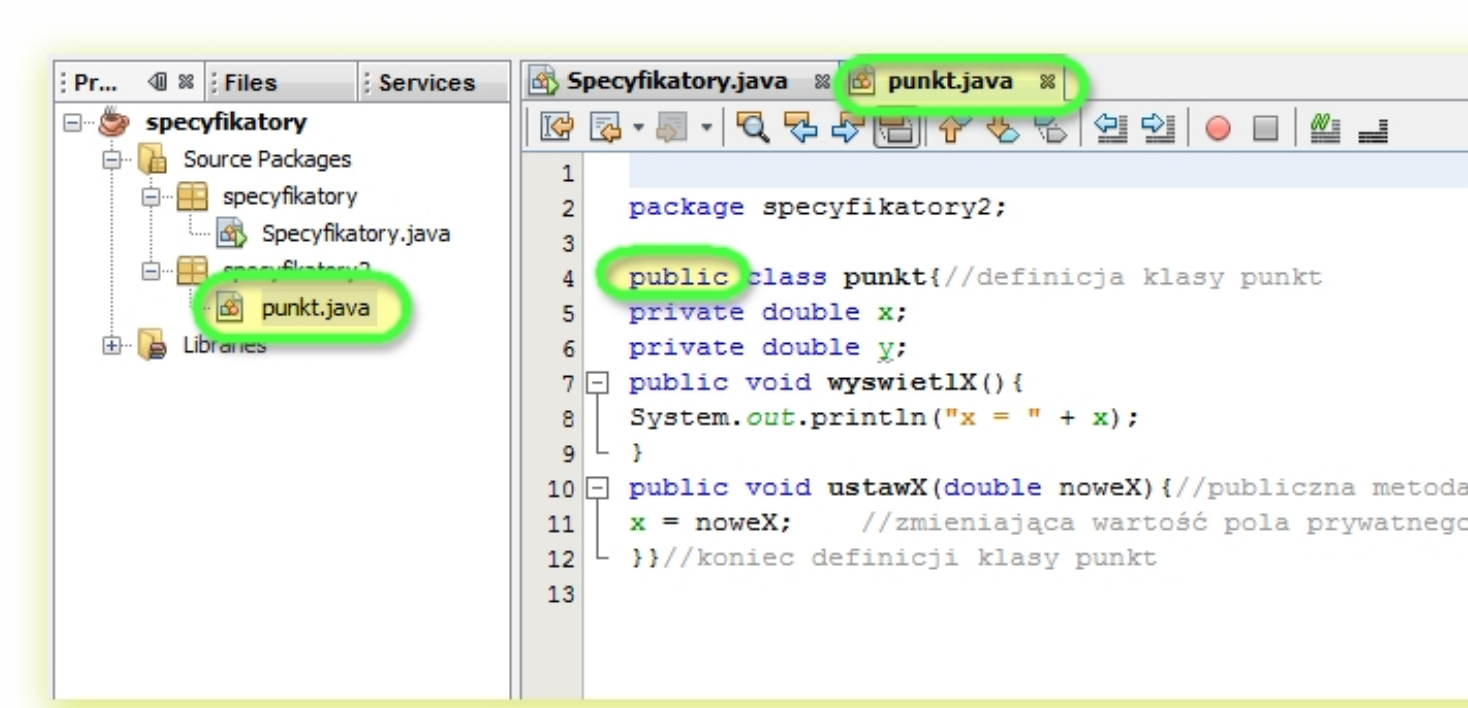
6. ... i wklejamy go do pliku klasy punkt

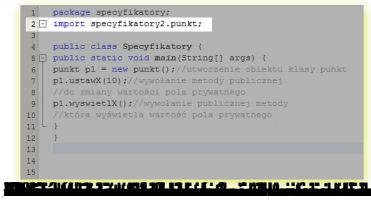


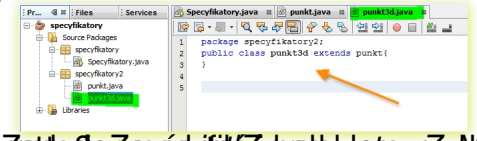


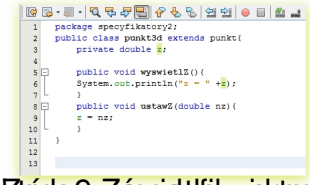
Jednak po wpisaniu tego polecenia program zgłasza nam błąd. jaki?

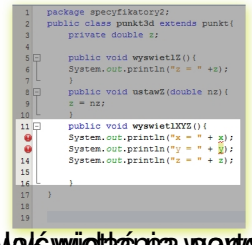


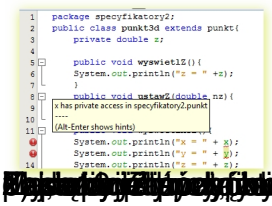

Pr...
Files
Services
specyfikatory
Source Packages
specyfikatory
Specyfikatory.java
punkt.java
Libraries
Specyfikatory.java
punkt.java
1 package specyfikatory2;
2 public class punkt{//definicja klasy punkt
3 private double x;
4 private double y;
5 public void wyswietlX(){
6 System.out.println("x = " + x);
7 }
8 public void ustawX(double noweX){//publiczna metoda
9 x = noweX; //zmieniająca wartość pola prywatnego
10 }}//koniec definicji klasy punkt

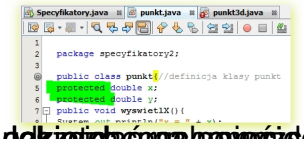


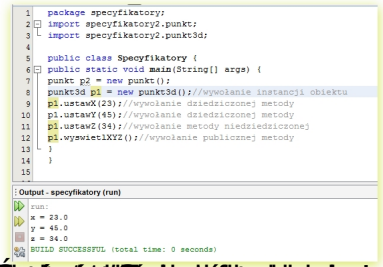












, *XYZ*, która sumuje