14. Interfejsy

Zadanie 01 Przeanalizuj kod zawarty w pliku `Figures.java`. Zidentyfikuj zależności między klasami. Czy można usunąć konstruktor bezargumentowy w klasie `Figure`?

Zadanie 02 Do programu dopisz klasę `Circle` dziedziczącą po klasie `Figure`, która będzie posiadała atrybut reprezentujący promień okręgu.

1 Słowo kluczowe `super`

Zadanie 03 W konstruktorach klas `Square` oraz `Circle` powtórzony jest kod inicjalizujący składowe klasy nadrzędnej `Figure`. Zastąp je wywołaniem odpowiedniego konstruktora nadklasy wykorzystując słowo kluczowe `super`.

Czy teraz możliwe jest usunięcie konstruktora bezargumentowego w klasie `Figure`? Dlaczego tak się dzieje?

2 Słowo kluczowe `final`

Zadanie 04 W klasie `Circle` wyodrębnij wartość PI `3.14` do atrybutu statycznego. Użyj słowa kluczowego `final` aby uniemożliwić modyfikację wartości tego atrybutu.

3 Interfejsy

Interfejsy (ang. *interfaces*) to abstrakcyjne typy posiadające zdefiniowane jedynie operacje, a nie same dane (tj. metody, nie atrybuty). Służą określeniu wymagań dla implementujących je klas.

```
1  interface Barable {
2      // kazda klasa implementujaca interfejs Barable bedzie musiala
3      // zdefiniowac ponizsza metode bar
4      public String bar(int baz);
5  }
6
7  class Foo implements Barable {
8      // ... skladowe klasy Foo
9
10     public String bar(int baz) {
11         return "Foo " + baz + "!";
12     }
13 }
```

Zadanie 05 Dopisz definicję interfejsu `Moveable` z jedną metodą postaci `void move(double a, double b)`, dzięki której będzie możliwe przemieszczanie obiektów figury.

Czy możliwe jest utworzenie obiektu typu `Moveable`?

Zadanie 06 W klasie `Figure` zaimplementuj interfejs `Moveable`, definiując odpowiednią metodę.

Zadanie 07 W głównej funkcji programu, stwórz tablicę przechowującą obiekty klas implementujących interfejs `Moveable`, np.:

```
1 Moveable[] moveables = { ... };
```

Umieść w niej kilka obiektów `Square` oraz `Circle`, a następnie w pętli wywołaj metodę `move`. Czy jest możliwe wywołanie innej metody tych klas?

4 Klasy abstrakcyjne a interfejsy

Klasy abstrakcyjne oraz interfejsy pozwalają realizować pewne funkcjonalności między klasami w podobny sposób, kiedy lepiej użyć klasy abstrakcyjnej, a kiedy interfejsu?

Według dokumentacji Javy, powinieneś rozważyć użycie klasy abstrakcyjnej, jeżeli przynajmniej jedna z poniższych sytuacji jest prawdziwa:

- Chcesz współdzielić kod pomiędzy kilka blisko powiązanych klas.
- Spodziewasz się, że klasy pochodne klasy abstrakcyjnej będą miały wiele wspólnych metod lub atrybutów, lub będą wymagały modyfikatorów dostępu innych niż publiczne (tj. prywatne lub chronione).
- Potrzebujesz zadeklarować inne niż statyczne lub stałe (`final`) atrybuty — pozwalają one na definiowanie metod, które mogą modyfikować stan obiektu.

Natomiast użycie interfejsu rozważ, jeżeli prawdziwa jest któraś z poniższych sytuacji:

- Spodziewasz się, że niepowiązane ze sobą klasy będą implementowały Twój interfejs. Na przykład interfejsy `Comparable` oraz `Cloneable` są implementowane przez wiele nie związanych ze sobą klas.
- Chcesz określić zachowanie pewnego typu danych, lecz na razie nie jest istotne kto implementuje to zachowanie.
- Chcesz wykorzystać wielokrotne dziedziczenie.

Zadanie 08 W klasie `Figure` zaimplementuj interfejs `Comparable` z standardowej biblioteki[1]. Porównywanie figur niech odbywa się poprzez porównywanie ich pól powierzchni.

[1] `http://docs.oracle.com/javase/6/docs/api/java/lang/Comparable.html`

Wskazówka: w metodzie `public int compareTo(Object object)` konieczne będzie rzutowanie argumentu `object` na instancję klasy `Figure`, aby uzyskać dostęp do metody `area()`. Możliwe jest również wykorzystanie metody `compareTo` z klasy `Double`.

Odkomentuj odpowiednie linie dla sprawdzenia poprawności swojego kodu.