Klasy abstrakcyjne

Zad.23. Możemy powiedzieć, że rozwiązanie standardowego zadania matematycznego składa się z trzech części: wprowadzenia danych, znalezienia rozwiązania i wypisania wyników (w tej kolejności). Przykładowa klasa opisująca zadanie może więc wyglądać:

```
public class Zadanie {
  void wprowadzDane() {          //definicja metody 1
  }
  void znajdzRozwiazanie() {     //definicja metody 2
  }
  void wypiszWyniki() {          //definicja metody 3
  }
  void rozwiaz() {               //definicja metody 4
    wprowadzDane();              //wywołanie metody 1
    znajdzRozwiazanie();         //wywołanie metody 2
    wypiszWyniki();              //wywołanie metody 3
  }
}
```

Metody `wprowadzDane`, `znajdzRozwiazanie`, `wypiszWyniki` mają puste implementacje, ponieważ na tym etapie (jeszcze bez znajomości samego zadania) nie jesteśmy w stanie podać, jakie dane należy wprowadzić, jakiej metody użyć, aby otrzymać rozwiązanie, ani też jakie dane wyjściowe będą wypisywane. Nazwy tych metod muszą się jednak pojawić, ponieważ są one wykorzystywane w zwykłej (nieabstrakcyjnej) metodzie `rozwiaz`. Dzięki metodzie `rozwiaz` użytkownik nie musi pamiętać trzech nazw metod, ani kolejności w jakiej należy je wywoływać.

Klasa Zadanie mogłaby być z powodzeniem klasą abstrakcyjną:

```
public abstract class Zadanie {
  abstract void wprowadzDane();
  abstract void znajdzRozwiazanie();
  abstract void wypiszWyniki();
  void rozwiaz() {
    wprowadzDane();
    znajdzRozwiazanie();
    wypiszWyniki();
  }
}
```

Wpisz kod abstrakcyjnej klasy Zadanie do pliku Zadanie.java.

W pliku Main.java umieść poniższy kod i spróbuj go skompilować.

```
public class Main {
  public static void main(String[] args) {
    Zadanie z = new Zadanie();
  }
}
```

Wyjaśnij efekt tej kompilacji.

W jaki sposób zatem wykorzystywać metody i klasy abstrakcyjne? Należy zdefiniować klasę potomną, która uzupełni definicje metod abstrakcyjnych klasy `Zadanie`, np:

```
import java.util.Random;
public class UkladRownan extends Zadanie{
  //ax + by = e
  //cx + dy = f
  int a, b, c, d, e, f;
  float x, y;
  String s;
  void wprowadzDane() {
    Random r = new Random();
    a = r.nextInt(9) + 1;  b = r.nextInt(9) + 1;  e = r.nextInt(9) + 1;
    c = r.nextInt(9) + 1;  d = r.nextInt(9) + 1;  f = r.nextInt(9) + 1;
    System.out.println(a + "x + " + b + "y = " + e);
    System.out.println(c + "x + " + d + "y = " + f);
  }
  void znajdzRozwiazanie() {
    float W  = a*d - b*c;
    float Wx = e*d - b*f;
    float Wy = a*f - e*c;
    if (W != 0) {
      x = Wx / W;
      y = Wy / W;
    }
    else s = "Brak lub nieskonczenie wiele rozwiazan";
  }
  void wypiszWyniki() {
    if (s != null) System.out.println(s);
    else System.out.println("x = " + x + ", y = " + y);
  }
}
```

Przetestuj klasę `Zadanie` i `UkladRownan` w klasie `Main`:

```
public class Main {
  public static void main(String[] args) {
    UkladRownan u = new UkladRownan();
    u.rozwiaz();
  }
}
```

W tym samym projekcie utwórz klasę o nazwie `SymbolNewtona` dziedziczącą po klasie `Zadanie`. Celem klasy `SymbolNewtona` jest obliczenie symbolu Newtona według wzoru:

$$\binom{n}{k} = \frac{n!}{k!(n-k)!}$$

dla danych wejściowych n i k, takich że $0 \leq k \leq n$.
Do klasy `SymbolNewtona` możesz dodać dodatkową metodę obliczającą silnię przekazanej w parametrze liczby naturalnej:

```
long silnia (int p) {
  //tresc metody
}
```

Zad.24. Zdefiniuj klasę bazową `Liczba`, w której znajdą się metody abstrakcyjne

- `wczytaj` - nadająca wartość (np. losową) liczbie,
- `wyswietl` - wyświetlająca liczbę
- `obliczKwadrat` - obliczająca kwadrat liczby.

Utwórz klasy `Rzeczywista` i `Zespolona` dziedziczące po klasie `Liczba`. Do klas pochodnych wprowadź odpowiednie pola reprezentujące wartości tych liczb. Przetestuj napisany kod w klasie Main, np:

```
public class Main {
  public static void main(String[] args) {
    Rzeczywista r = new Rzeczywista();
    Zespolona z = new Zespolona();
    Liczba licz;

    licz = r;
    licz.wczytaj();
    licz.wyswietl();
    licz.obliczKwadrat();

    licz = z;
    licz.wczytaj();
    licz.wyswietl();
    licz.obliczKwadrat();
  }
}
```