Zdefiniować klasy z poniższego programu:

```
public class Test {

  public static void main(String[] args) {

    Pacjent[] pacjenci = { new ChoryNaGlowe("Janek"),
                           new ChoryNaNoge("Edzio"),
                           new ChoryNaDyspepsje("Marian")
                         };

    for (Pacjent p : pacjenci) {
      System.out.println("Pacjent:     " + p.nazwisko() + '\n' +
                         "Chory na:    " + p.choroba() + '\n' +
                         "Zastosowano: " + p.leczenie() +"\n\n"
                         );
    }

  }

}
```

w taki sposób, aby otrzymać w wyniku następujący wydruk:

```
Pacjent:     Janek
Chory na:    głowa
Zastosowano: aspiryna
Pacjent:     Edzio
Chory na:    noga
Zastosowano: gips
Pacjent:     Marian
Chory na:    dyspepsja
Zastosowano: węgiel
```

Uwaga: metody choroba() i leczenie() muszą być **wywoływane polimorficznie**.

Uwaga: nazwy klas bez polskich znaków. Klasy z nazwami z polskimi znakaami nie będą sprawdzane.

Uwaga: utworzonej przez generator projektów klasy Test nie wolno zmieniać.

Kwiaciarnia (uwaga: niezbędny do wykonania tego zadania materiał zawiera punkt o polimorfizmie w wykładzie, dotyczącym programowania obiektowego)

Napisać aplikację, która symuluje zakupy w kwiaciarni "samoobsługowej".
W kwiaciarni są kwiaty, kwiaty mają swoje nazwy oraz kolory. Ceny kwiatów znajdują się w cenniku.
Do kwiaciarni przychodzą klienci. Klienci mają imiona oraz dysponują jakimś zasobem pieniędzy. Wybierają kwiaty i umieszczają je na wózku sklepowym. Następnie płacą za zawartość wózka i przepakowują ją do pudełka (jakiegoś pudełka na kwiaty :-).

Aplikacja wymaga zdefiniowania kilku klas i umiejętnego ich użycia, w taki sposób by następujący program działał poprawnie.

```
public class FloristsTest {
   // definicja metody sumowania wartosci kwiatów o podanym kolorze
   static int valueOf(Box box, String color) {
        /*<- tu trzeba wpisac kod metody */
   }

   public static void main(String[] args) {
     // Kwiaciarnia samoobsługowa
     // ustalenie cennika
     PriceList pl = PriceList.getInstance();
     pl.put("róża", 10.0);
     pl.put("bez", 12.0);
     pl.put("piwonia", 8.0);

     // Przychodzi klient janek. Ma 200 zł
     Customer janek = new Customer("Janek", 200);

     // Bierze różne kwiaty: 5 róż, 5 piwonii, 3 frezje, 3 bzy
     janek.get(new Rose(5));
     janek.get(new Peony(5));
     janek.get(new Freesia(3));
     janek.get(new Lilac(3));

     // Pewnie je umieścił na wózku sklepowyem
     // Zobaczmy co tam ma
     ShoppingCart wozekJanka = janek.getShoppingCart();
     System.out.println("Przed płaceniem " + wozekJanka);

     // Teraz za to zapłaci...
     janek.pay();

     // Czy przypadkiem przy płaceniu nie okazało się,
     // że w koszu są kwiaty na które nie ustalono jeszcze ceny?
     // W takim arzie zostałyby usunięte z wózka i Janek nie płaciłb
y za nie
```

```
            // Również może mu zabraknąc pieniędzy, wtedy też kwaity są odk
ładane.

            System.out.println("Po zapłaceniu " + janek.getShoppingCart());

            // Ile Jankowi zostało pieniędzy?
            System.out.println("Jankowi zostało : " + janek.getCash() + " z
ł");

            // Teraz jakos zapakuje kwiaty (może do pudełka)
            Box pudelkoJanka = new Box(janek);
            janek.pack(pudelkoJanka);

            // Co jest teraz w wózku Janka...
            // (nie powinno już nic być)
            System.out.println("Po zapakowaniu do pudełka
" + janek.getShoppingCart());

            // a co w pudełku
            System.out.println(pudelkoJanka);

            // Zobaczmy jaka jest wartość czerwonych kwiatów w pudełku Jank
a
            System.out.println("Czerwone kwiaty w pudełku Janka kosztowały:
 "
                + valueOf(pudelkoJanka, "czerwony"));

            // Teraz przychodzi Stefan
            // ma tylko 60 zł
            Customer stefan = new Customer("Stefan", 60);

            // Ale nabrał kwiatów nieco za dużo jak na tę sumę
            stefan.get(new Lilac(3));
            stefan.get(new Rose(5));

            // co ma w wózku
            System.out.println(stefan.getShoppingCart());

            // płaci i pakuje do pudełka
            stefan.pay();
            Box pudelkoStefana = new Box(stefan);
            stefan.pack(pudelkoStefana);

            // co ostatecznie udało mu się kupić
            System.out.println(pudelkoStefana);
            // ... i ile zostało mu pieniędzy
            System.out.println("Stefanowi zostało : " + stefan.getCash() +
" zł");
          }
        }
```

Program ten wyprowadzi na konsolę:

```
Przed płaceniem Wózek własciciel Janek
róża, kolor: czerwony, ilość 5, cena 10.0
piwonia, kolor: czerwony, ilość 5, cena 8.0
frezja, kolor: żółty, ilość 3, cena -1.0
bez, kolor: biały, ilość 3, cena 12.0
Po zapłaceniu Wózek własciciel Janek
róża, kolor: czerwony, ilość 5, cena 10.0
piwonia, kolor: czerwony, ilość 5, cena 8.0
bez, kolor: biały, ilość 3, cena 12.0
```

```
Jankowi zostało : 74.0 zł
Po zapakowaniu do pudełka Wózek właściciel Janek -- pusto
Pudełko właściciel Janek
róża, kolor: czerwony, ilość 5, cena 10.0
piwonia, kolor: czerwony, ilość 5, cena 8.0
bez, kolor: biały, ilość 3, cena 12.0
Czerwone kwiaty w pudełku Janka kosztowały: 90
Wózek właściciel Stefan
bez, kolor: biały, ilość 3, cena 12.0
róża, kolor: czerwony, ilość 5, cena 10.0
Pudełko właściciel Stefan
bez, kolor: biały, ilość 3, cena 12.0
Stefanowi zostało : 24.0 zł
```

Uwaga: kod tego programu można zmienic tylko w miejscu zaznaczonym na zielono.

Dodawanie do programu nowych rodzajów kwiatów ma byc bardzo łatwe. Przy dodaniu nowego rodzaju kwiatów nie wolno modyfikować żadnych innych klas programu.
Wymagania dodatkowe:

- należy wykorzystać klasy abstrakcyjne i **polimorfizm**
- należy zminimalizować kod klas ShoppingCart i Box
- należy zdefiniować klasę PriceList jako **singleton** (możemy mieć zawsze tylko jeden cennik)

Ważne uwagi.

- W kwiaciarni mogą być kwiaty, których zapomniano dodać do cennika. Wtedy przy płaceniu są one usuwane z naszego wózka.
- Może się okazać, że klient nie dysponuje odpowiednią kwotą pieniędzy aby zapłacić za całą zawartość wóżka. Wtedy z wózka usuwane są kwiaty, za które klient nie może zapłacić (ale nie pojedyńczo, tylko w kompletach np. po stefan.get(new Lilac(3)) usuwane są te trzy bzy na które Stefan nie ma pieniędzy).
- Warto zwrócić uwagę na odpowiednio zdefiniowanie metody toString() w niektórych klasach.

I na koniec: nie przejmujemy się tym, że np. róże mogą mieć wiele kolorów. Dla uproszczenia przyjęliśmy, że róże są czerwone itd.

Odwracanie

Zdefiniować interfejs Reversible z jedną metodą ***Reversible reverse()*** i zaimplementować ją w klasach ReversibleString i ReversibleDouble.
Metoda reverse dla Stringów odwraca napis, a dla liczb - odwraca liczbę (czyli z napisu kot robi tok a z liczby 3 robi 0.3333).
Metoda zwraca Reversible z aktualną (odwróconą) wartością.

Zapewnić by następujący program wykonał się prawidłowo i dał pokazane wyniki:

```
public class ReverseTest {

  public static void main(String[] args) {

        Reversible[] revers = new Reversible[] {
            new ReversibleString("Kot"),
            new ReversibleDouble(2),
            new ReversibleDouble(3),
            new ReversibleString("Pies"),
            new ReversibleString("Ala ma kota i psa"),
            new ReversibleDouble(10),
        };

        System.out.println("Normalne:");
        for (Reversible r : revers) {
          System.out.println(r);
        }

        for (Reversible r : revers) {
          r.reverse();
        }

        System.out.println("Odwrócone:");
        for (Reversible r : revers) {
          System.out.println(r);
        }

        System.out.println("Przywrócone i zmienione:");
        for (Reversible r : revers) {
           /*<-  co tu trzeba napisać */
        }
      }
    }
```

```
Wynik:
Normalne:
Kot
2.0
3.0
Pies
Ala ma kota i psa
10.0
Odwrócone:
```

```
toK
0.5
0.3333333333333333
seiP
asp i atok am alA
0.1
Przywrócone i zmienione:
Tekst Kot
12.0
13.0
Tekst Pies
Tekst Ala ma kota i psa
20.0
```

Uwaga: kod klasy ReverseTest może (i musi) być zmieniony tylko w miejscu zaznaczonym na zielono. Inne modyfikacje pliku nie są dopuszczalne i skutkują otrzymaniem zera punktów.

Zadanie. Śpiewacy z klasą abstrakcyjną (4 punkty)

Zdefiniować klasę abstrakcyjną Spiewak reprezentującą śpiewaków. Każdy śpiewak posiada nazwisko oraz numer startowy (np. w konkursie talentów), nadany automatycznie przy tworzeniu obiektu. Klasa Spiewak powinna posiadać m.in. konstruktor Spiewak(String nazwisko) oraz następujące metody:

a. abstrakcyjną: abstract String spiewaj(), która docelowo zwraca tekst śpiewany przez śpiewaka w konkursie.
b. public String toString() zwracającą informację o śpiewaku.
c. statyczną:... najglosniej(...) przyjmującą tablicę obiektów/śpiewaków oraz zwracającą obiekt/śpiewaka, w którego śpiewanym tekście znajduje się najwięcej dużych liter (patrz. metodę spiewaj()).

W metodzie main klasy testującej Main:

a. stworzyć kilka (co najmniej 3) obiektów/śpiewaków poprzez użycie **anonimowych klas wewnętrznych** rozszerzających klasę Spiewak. Użycie tych klas polega na podawaniu tego, co śpiewak ma śpiewać w konkursie.
b. stworzyć tablicę śpiewaków składającą się z obiektów z punktu a.
c. testować metodę najglosniej(...) z klasy Spiewak.

Stworzyć klasę Spiewak w taki sposób, aby następująca metoda main z klasy Main:

```
public class Main {
    public static void main(String[] args)
    {
        Spiewak s1 = new Spiewak("Carrey"){
          /*<-  kod */
        };

        Spiewak s2 = new Spiewak("Houston"){
          /*<-  kod */
        };

        Spiewak s3 = new Spiewak("Madonna"){
          /*<-  kod */
        };

        Spiewak sp[] = {s1, s2, s3};

        for (Spiewak s : sp)
            System.out.println(s);

        System.out.println("\n" + Spiewak.najglosniej(sp));
    }
}
```

wyprowadziła prawidłową informację dokładnie jak poniżej:

```
(1) Carrey: ooooooooooo
(2) Houston: a4iBBiii
(3) Madonna: aAa

(2) Houston: a4iBBiii
```

Ważne: Kod klas Main utworzony przez generator może (i musi) być zmieniony tylko w miejscach zaznaczonych na zielono. Inne modyfikacje tego kodu nie są dopuszczalne i będą skutkować uzyskaniem 0 punktów.