GUI II

Klasy abstrakcyjne c.d., interfejsy, klasa anonimowa, AWT i JFrame

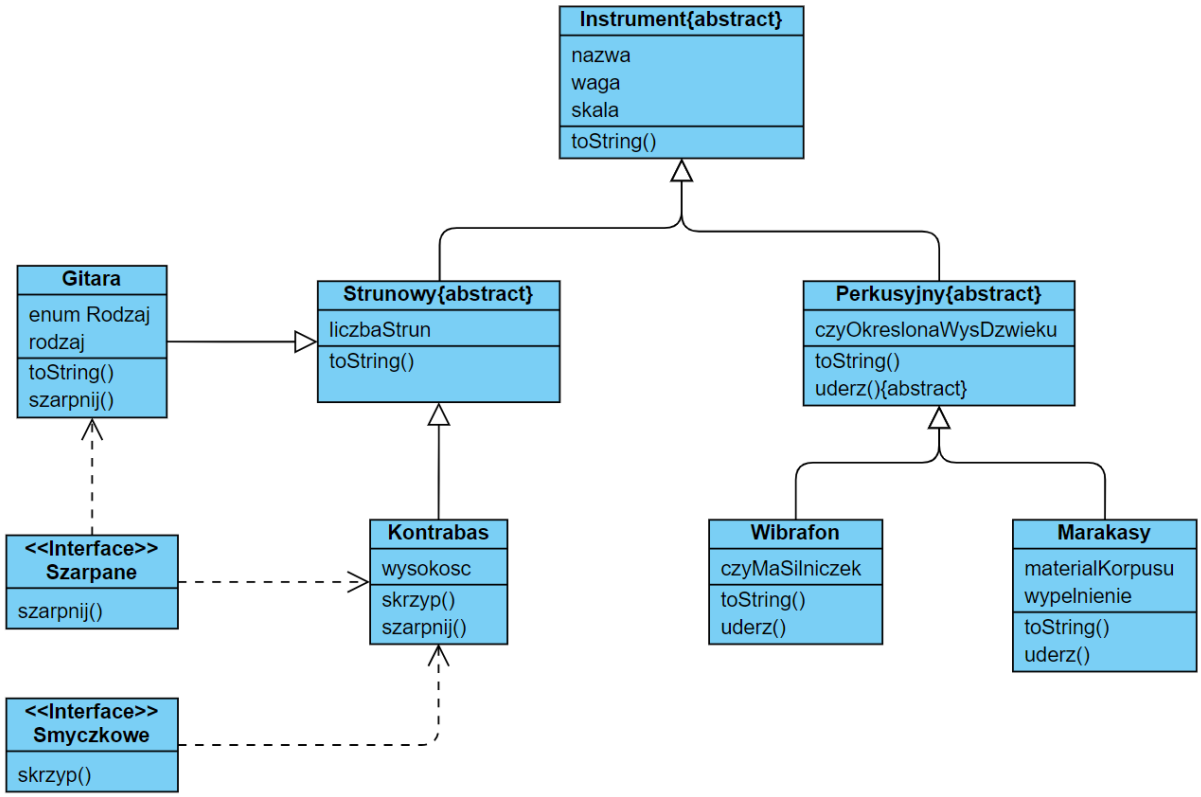
1) Stwórz program oparty o poniższy schemat

Ale zanim schemat, małe przypomnienie z wykładu:

Klasa	**Klasa abstrakcyjna**
Posiada pola, metody	Posiada pola, metody
Można utworzyć jej obiekt (jeśli konstruktor nie jest prywatny)	Nie można utworzyć obiektu
Metody muszą mieć ciało.	Metody abstrakcyjne nie muszą mieć ciała (niemniej mogą)
Musi implementować wszystkie metody interfejsów	Może implementować metody interfejsów

Klasa	**Interfejs**
Posiada pola, metody	Tylko metody, statyczno-finalne pola
Metody muszą zostać zaimplementowane (ciało metody)	Nie mogą (prawie) mieć implementacji metod.
Może mieć konstruktor	Nie może mieć konstruktora
Może dziedziczyć po jednej klasie (domyślnie po Object), może implementować wiele interfejsów (w tym żadnego).	Może dziedziczyć po wielu interfejsach (implementować wiele interfejsów)
Jeśli implementuje interfejs, musi mieć wszystkie metody z tego interfejsu.	Jeśli rozszerza interfejs, to nie wymaga uzupełnienia o metody bazowego interfejsu.

Schemat:



Zanim przejdziesz do właściwego wykonania zadania, spróbuj stworzyć obiekty dla obu powyższych interfejsów. Jeżeli ci się udało to stworzyłeś obiekt klasy anonimowej.

Zadanie zaczynamy od stworzenia prawej strony schematu.

Skala w klasie Instrument dotyczy rozpiętości tonalnej można podać jako liczba oktaw.

Zwracamy uwagę na brak potrzeby definiowania metody *uderz()* w nadklasie.

Tworząc obiekty klasy *Marakas* bądź *Wibrafon* musimy sprawdzać wartość pola *czyOkreslonaWysDzwieku (boolean)*. Wynika to z faktu iż marakasy nie są instrumentami z których wydobywający się dźwięk ma określoną wysokość, a w przypadku wibrafonu już tak. Chodzi o to, że w konstruktorze obu tych klas musimy patrzeć, czy wartość wyżej wspomnianego pola jest odpowiednia, a jeżeli nie jest to należy podnieść w konstruktorze jakiś *exception* (do zaimplementowania).

W klasie *Gitara* tworzymy typ enum *Rodzaj*, który może przyjąć 3 stany: *akustyczna*, *elektryczna* lub *basowa*. Przy tworzeniu obiektów tej klasy, posługujemy się tym enumem.

Należy zdefiniować działanie wszystkich metod z interfejsów w klasach implementujących je. Może być to zwyczajny *println()* w którym napiszą sobie państwo onomatopeję.

Proszę stworzyć kilka obiektów i przedstawić ich działanie.

Dodatkowe:
Za pomocą słowa kluczowego *default* zdefiniuj w interfejsach domyślne definicje metod. Co się zmieniło w klasach implementujących te interfejsy?

2) Posługując się poniższym przykładem, dopisz kod (w klasie *ShapesPanel* w metodzie *paintComponent*) pozwalający na wylosowanie koloru |w formacie RGB (wartości od 0 do 255) oraz wylosowanie wymiarów figury. Następnie zmodyfikuj wywołanie metod *drawRect, fillRect, drawOval, fillOval* tak aby rysowana figura pojawiała się na środku kursora. Pola obiektowe *x* oraz *y* są koordynatami kursora myszki.

Klasa Main:

```
public class Main {

    public static void main(String[] args) {

        new GUI();

    }
}
```

Klasa GUI:

```
import javax.swing.*;
import java.awt.*;

public class GUI extends JFrame {

    public GUI() {

        this.setTitle("GUI II");
        this.setSize(600, 400);
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        this.setResizable(false);
        this.setLocationRelativeTo(null); //Okno będzie pojawiaś się na
środku (o ile mamy jeden ekran)

        this.setLayout(new BorderLayout());
        JPanel panel = new ShapesPanel();

        this.add(panel);
        this.setVisible(true);

    }
}
```

Klasa ShapesPanel:

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

public class ShapesPanel extends JPanel {

    int x, y;

    public ShapesPanel() {

        this.x = 0;
        this.y = 0;

        this.setSize(600, 400);
        this.setBackground(Color.WHITE);

        MouseHandler mouseHandler = new MouseHandler();
        this.addMouseListener(mouseHandler);

    }

    @Override
    public void paintComponent (Graphics gr){
        super.paintComponent(gr);
        Graphics2D g2d = (Graphics2D) gr;

        //tu losujemy wartości
        int r =
        int g =
        int b =

        int width =
        int height =

        g2d.setColor(new Color(r, g, b));

        //tu edytujemy punkt wyrysowania figury
        if (Math.random() > 0.5) {
            g2d.drawRect(x, y, width, height);
            g2d.fillRect(x, y, width, height);
        } else {
            g2d.drawOval(x, y, width, height);
            g2d.fillOval(x, y, width, height);
        }


    }

    private class MouseHandler extends MouseAdapter {

        @Override
        public void mouseClicked(MouseEvent event) {
            x = event.getX();
            y = event.getY();
            System.out.println(x + ", " + y);
            repaint();
        }
    }
}
```