Imię i nazwisko: **22L kol1**

Proszę wpisywać odpowiedzi w miejscach na to przeznaczonych

Zadanie 1 - obiektowe wzorce projektowe (2pkt)

Hierarchia opisuje samochód, który może mieć lub nie mieć kamerę, może mieć lub nie mieć Radar, oraz może mieć lub nie mieć Lidar. Samochód dostarcza metodę 'move', która jest modyfikowana w ten sposób, że jeżeli kamera lub radar lub lidar wykryje przeszkodę to nie wykonuje ruchu.

Uprość strukturę klas, aby nie powielać kodu.

W klasach dla uproszczenia sensory (kamera, lidar, radar) są reprezentowane przez zmienną logiczną. Taka reprezentacja jest wystarczająca w ramach tego zadania.

```
class Vehicle {
  virtual void move() { /* wykonuje ruch */ }
};

class VehicleLidar : public Vehicle {
  void move() override {
    bool lidar_alarm = false; //symuluje sensor
    if(!lidar_alarm) Vehicle::move();
  }
};

class VehicleRadar : public Vehicle {
  void move() override {
    bool radar_alarm = false; //symuluje sensor
    if(!radar_alarm) Vehicle::move();
  }
};

class VehicleCamera : public Vehicle {
  void move() override {
    bool camera_alarm = false; //symuluje sensor
    if(! camera_alarm) Vehicle::move();
  }
};

class VehicleLidarRadar : public Vehicle {
  void move() override {
    bool lidar_alarm = false; //symuluje sensor
    bool radar_alarm = false; //symuluje sensor
    if(!lidar_alarm && !radar_alarm )
        Vehicle::move();
  }
};

class VehicleRadarCamera : public Vehicle {
  void move() override {
    bool radar_alarm = false; //symuluje sensor
    bool camera_alarm = false; //symuluje sensor
    if(!radar_alarm && !camera_alarm)
        Vehicle::move();
  }
};

class VehicleCameraLidar : public Vehicle {
  void move() override {
    bool camera_alarm = false; //symuluje sensor
    bool lidar_alarm = false; //symuluje sensor
    if(!camera_alarm && !lidar_alarm)
        Vehicle::move();
  }
};

class VehicleRadarLidarCamera : public Vehicle {
  void move() override {
    bool radar_alarm = false; //symuluje sensor
    bool lidar_alarm = false; //symuluje sensor
    bool camera_alarm = false; //symuluje sensor
    if(!radar_alarm && !lidar_alarm && !camera_alarm)
        Vehicle::move();
  }
};
```

Zadanie 2 - obiektowe wzorce projektowe (2pkt)

Obiekt Note reprezentuje nutę wykorzystywaną do gry na instrumentach. Obiekt Song to utwór muzyczny. Dostarczamy 2 utwory: „Yankee Doodle" oraz „Hungarian Dance" grane na 3 różnych instrumentach: pianino, skrzypce, trąbka. **Uprość strukturę klas, aby nie powielać kodu.** Nie implementuj metod play.

```
struct Note {};
class Song {
    virtual void play() = 0;
};


class YankeeDoodlePiano : public Song {
public:
    YankeeDoodlePiano();
    //implementacja nie ma znaczenia w tym zadaniu
    void play() override { }
};

class HungarianDancePiano : public Song {
public:
    HungarianDancePiano();
    void play() override { }
};

class YankeeDoodleViolin : public Song {
public:
    YankeeDoodleViolin();
    void play() override { }
};

class HungarianDanceViolin : public Song {
public:
    HungarianDanceViolin();
    void play() override { }
};

class YankeeDoodleTrumpet : public Song {
public:
    YankeeDoodleTrumpet();
    void play() override { }
};

class HungarianDanceTrumpet : public Song {
public:
    HungarianDanceTrumpet();
    void play() override { }
};
```

Uwagi do prowadzącego (R. Nowaka):

Zadanie 3 - cykl życia oprogramowania (2pkt)

Proszę odpowiedzieć na następujące pytania:

1. Kiedy można powiedzieć o wymaganiu, że jest jednoznaczne? Podaj przykład – własny, taki, wskazujący na zrozumienie tematu.

2. W którym momencie procesu testowania uzyskujemy gwarancję poprawności programu? Kiedy możemy go dostarczyć klientowi?

Zadanie 4 - SOLID (2pkt)

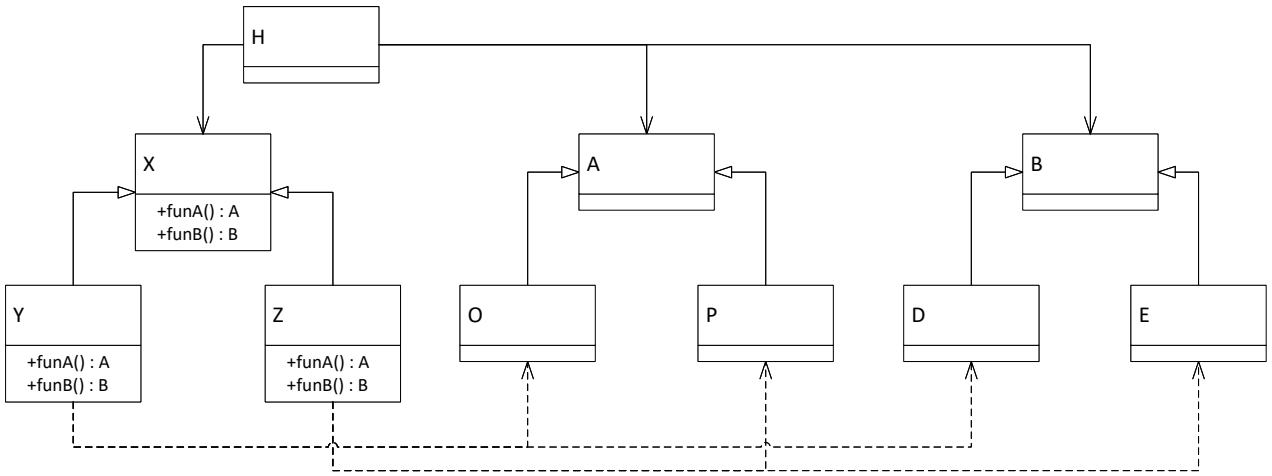
Czy ten fragment kodu jest zgodny z regułami SOLID? Jeśli nie, to w jaki sposób je narusza (może naruszać więcej niż jedną regułę)? Jak należałoby poprawić kod?

```
class Document
{
public:
  void save(std::string_view fileName)
  {
    std::ofstream out(fileName);
    out << title();
    out << "\n";
    out << data(); //  typ Data poprawnie obsłuży tę operację
    out << "\n.\n";
  }

  std::string title() const;
  Data data() const; //  implementacja typu Data jest nieistotna
  void combine(const Data& other)
  {
    //  szczegóły implementacji nie mają znaczenia
  }
};
```

Zadanie 5 - UML (2pkt)

Opisz wszystkie relacje jakie widzisz między klasami widocznymi na diagramie. Jeśli diagram przypomina znajomy wzorzec, zaproponuj ciekawsze nazwy klas.



Uwagi do prowadzącego (K. Grochowskiego):