Imię i nazwisko: | | | | | | | | | | | | | | | | | | | | | | | | | | | **25Z kol 1**

Proszę założyć, że udostępniona jest przestrzeń nazw std

Zadanie 1 - obiektowe wzorce projektowe (3 pkt)

Wektor obiektów `Drink`, czyli obiekt typu `vector<UDrink>`, można zapisywać do pliku tekstowego, a następnie go odczytywać. Uzupełnij metodę `load` klasy `DrinksFile`, aby to umożliwić. Klasy `Drink`, `Tea`, `Coffee` są poprawne.

```
class Drink {
public:
    virtual void save(ostream& os) const = 0;
    virtual void load(istream& is) = 0;
    virtual ~Drink() = default;
};
using UDrink = unique_ptr<Drink>;
using Drinks = vector<UDrink>;
class Tea : public Drink {
public:
    static const int ID = 1;
    Tea(const string& desc) : desc_(desc) {}
    void save(ostream& os) const override { os << ID << endl << desc_ << endl; }
    void load(istream& is) override { getline(is, desc_); }
private:
    string desc_;
};
UDrink createTea() { return UDrink(new Tea); }
class Coffee : public Drink {
public:
    static const int ID = 2;
    Coffee(const string& desc) : desc_(desc) {}
    void save(ostream& os) const override { os << ID << endl << desc_ << endl; }
    void load(istream& is) override { getline(is, desc_); }
private:
    string desc_;
};
UDrink createCoffee() { return UDrink(new Coffee); }
```

```
class DrinksFile {
public:
    DrinksFile(const string& filename) : filename_(filename) {}

    void save(const Drinks& drinks) {
        ofstream f(filename_);
        for(const UDrink& d : drinks) { d->save(f); }
    }

    Drinks load() {
        Drinks vec;
        try { //bledny plik itp.
            ifstream f;
            f.open(filename_);

            for( string line; getline( f, line ); ) { //czyta plik po linii
               int id = stoi(line); // string -> integer




            }

        } catch(std::exception& e) {
            cerr << e.what() << endl;
        }
        return vec;
    }
private:
    string filename_;
  };
```

Zadanie 2 - obiektowe wzorce projektowe (2pkt)

Obiekty klas pochodnych po `Member` reprezentują członków zespołów. i pracowników (obiekty typu `Staff`). Uzupełnij kod, funkcji `checkProper`, która bada poprawność zespołu. Zespół jest poprawny, gdy ma taką samą liczbę studentów (obiekty typu `Student`) i pracowników.

Jeżeli s1 i s2 to studenci, zaś s3 i s4 to pracownicy, zaś zespoły są takie jak poniżej,

```
Team t1 { &s1, &s3 };
Team t2 { };
Team t3 { &s1 };
```

to dla zespołu t1 i t2 funkcja powinna zwrócić `true`, zaś dla t3 `false`.

```
class Member {
public:
    virtual ~Member() = default;
    virtual void accept(Visitor& v) const = 0;
};
using VMember = Member*; //widok na obiekt, nie zarzadza obiektem
//Powinno byc 'using VMember = reference_wrapper<Member>', ale w tym zadaniu poprzednia definicja wystarczy
using Team = vector<VMember>;

class Visitor {
public:
    virtual void visit(const Student& s) = 0;
    virtual void visit(const Staff& a) = 0;
    virtual ~Visitor() = default;
};
class Student : public Member {
public:
    Student() : Member() {}
    void accept(Visitor& v) const override { v.visit(*this); }
};
class Staff : public Member {
public:
    Staff() {}
    void accept(Visitor& v) const override { v.visit(*this); }
};
```

```
bool checkProper(const Team& t) {
```

Uwagi do prowadzącego (R. Nowaka):

Zadanie 3 - wytwarzanie oprogramowania (2pkt)

Proszę odpowiedzieć na następujące pytania:

1. Czego należy oczekiwać od dobrze przeprowadzonego `code review`?

2. Czy to jest dobre wymaganie (jeśli nie – to dlaczego):

   Istniejący system należy w miarę możliwości wzbogacić o funkcję śledzenia zależności pomiędzy IRD a ICD oraz ICD a IRD.

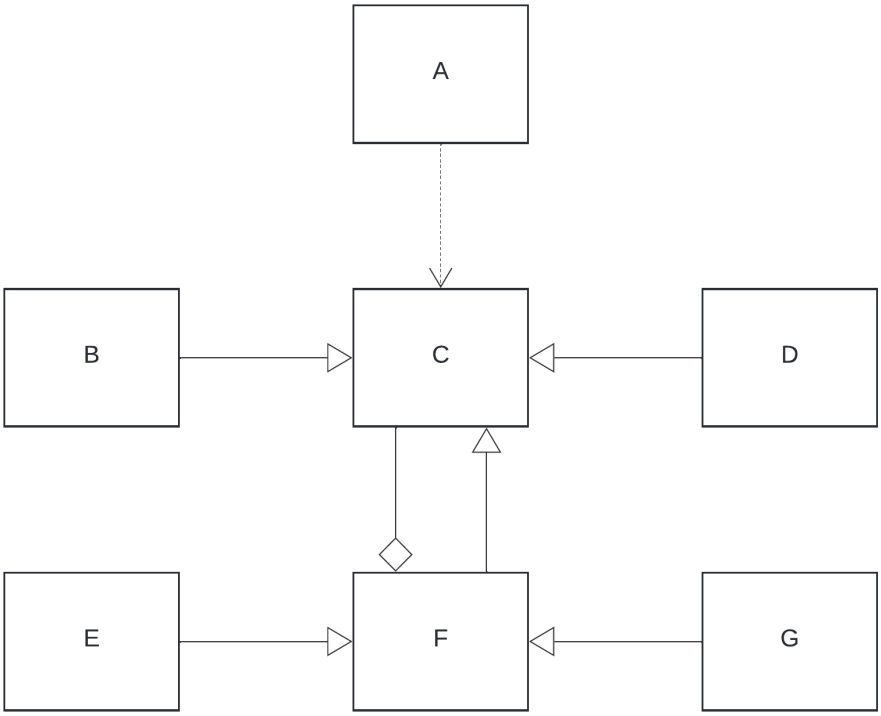
Zadanie 4 - SOLID (2pkt)

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

```
class Page
{
public:
  const std::vector<std::string>& words() const;

  bool isSpellingCorrect(std::weak_ptr<Dictionary> dictionary) const
  {
    return std::ranges::all_of(words(),
                               [&dictionary](const auto& w) {
                                 return dictionary->contains(w);
                               });
  }

  virtual void save(const std::ifstream& out)
  {
    std::ranges::copy(words(),
                      std::ostream_iterator<std::string>(out, ",␣"));
  }

private:
  // nieistotne dla zadania
};

class ColorPage : public Page
{
public:
  Color color() const;

  void save(const std::ifstream& out) override
  {
    out << color();
    Page::save(out);
    out.close();
  }

private:
  // nieistotne dla zadania
};
```

Zadanie 5 - UML (2pkt)

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



Uwagi do prowadzącego (K. Grochowskiego):