Imię i nazwisko: **21Z kol1**

Proszę wpisywać odpowiedzi w miejscach na to przeznaczonych

Zadanie 1 - obiektowe wzorce projektowe (2pkt)

Proszę dostarczyć implementację funkcji **printError(e)**. Funkcja ta ma wypisać komunikat o błędzie na standardowe wyjście błędów. W systemie mamy 2 rodzaje błędów: PermissionException i NetworkException. W zadaniu proszę obsłużyć język polski (PL) oraz angielski (EN). Komunikat powinien być następujący: 'użytkownik ID nie ma uprawnień', 'no permission for user ID', 'błąd połączenia sieciowego', 'network error', ID to liczba, identyfikator użytkownika. W C++ komunikat na standardowe wyjście błędów można wygenerować poleceniem:

std::cerr << u8"Użytkownik " << 13 << u8" nie ma uprawnień" << std::endl;

Zadanie sprowadza się do zaimplementowania klas EnglishTranslator i PolishTranslator i podmiany kodu w kilku innych miejscach. Przepraszam za banalne i niepoprawne dostarczenie obiektów Translator w singletonie Settings, chciałem maksymalnie uprościć kod. Przykład użycia obok.

```
try {
  //kod ktory moze zglosic blad
} catch(Exception& e) {
    printError(e);
}
```

```
class Visitor {
public:
    virtual void visit(const PermissionException& e) = 0;
    virtual void visit(const NetworkException& e) = 0;
};
class Exception : public std::runtime_error { //klasa bazowa dla bledow
public:
    virtual void accept(Visitor& v) const = 0;
};
class PermissionException : public Exception { //wyjatek - brak uprawnien
public:
    PermissionException(int user_id = 0) : userId(user_id) {}
    void accept(Visitor& v) const override { v.visit(*this); }
    int getUserId() const { return userId; }
private:
    int userId; //identyfikator uzytkownika, ktory nie mial uprawnien
};
class NetworkException : public Exception { //wyjatek - blad wewnetrzny serwera
public:
    void accept(Visitor& v) const override { v.visit(*this); }
};
class Settings {
public:
    static Settings& getInstance() { if (! instance) instance = new Settings; return *instance; }
    void setLanguage(const std::string& l) { lang = l; }
    Translator& getTranslator() { //kod tej metody nie jest poprawny, powinna byc fabryka, prosze nie poprawiac
        if(lang == "PL") return *polishTranslator_;
        else return *englishTranslator_;
    }
private:
    static Settings* instance;
    Settings(const Settings&) = delete; Settings& operator=(const Settings&) = delete;
    Settings() {}
    ~Settings() { delete englishTranslator_; delete polishTranslator_; }
    std::string lang = "EN";
    //przechowywanie obiektow do translacji tak jak nizej to blad, prosze nie poprawiac
    Translator* englishTranslator_ = new EnglishTranslator;
    Translator* polishTranslator_ = new PolishTranslator;
};
```

class Translator

class PolishTranslator : public Translator

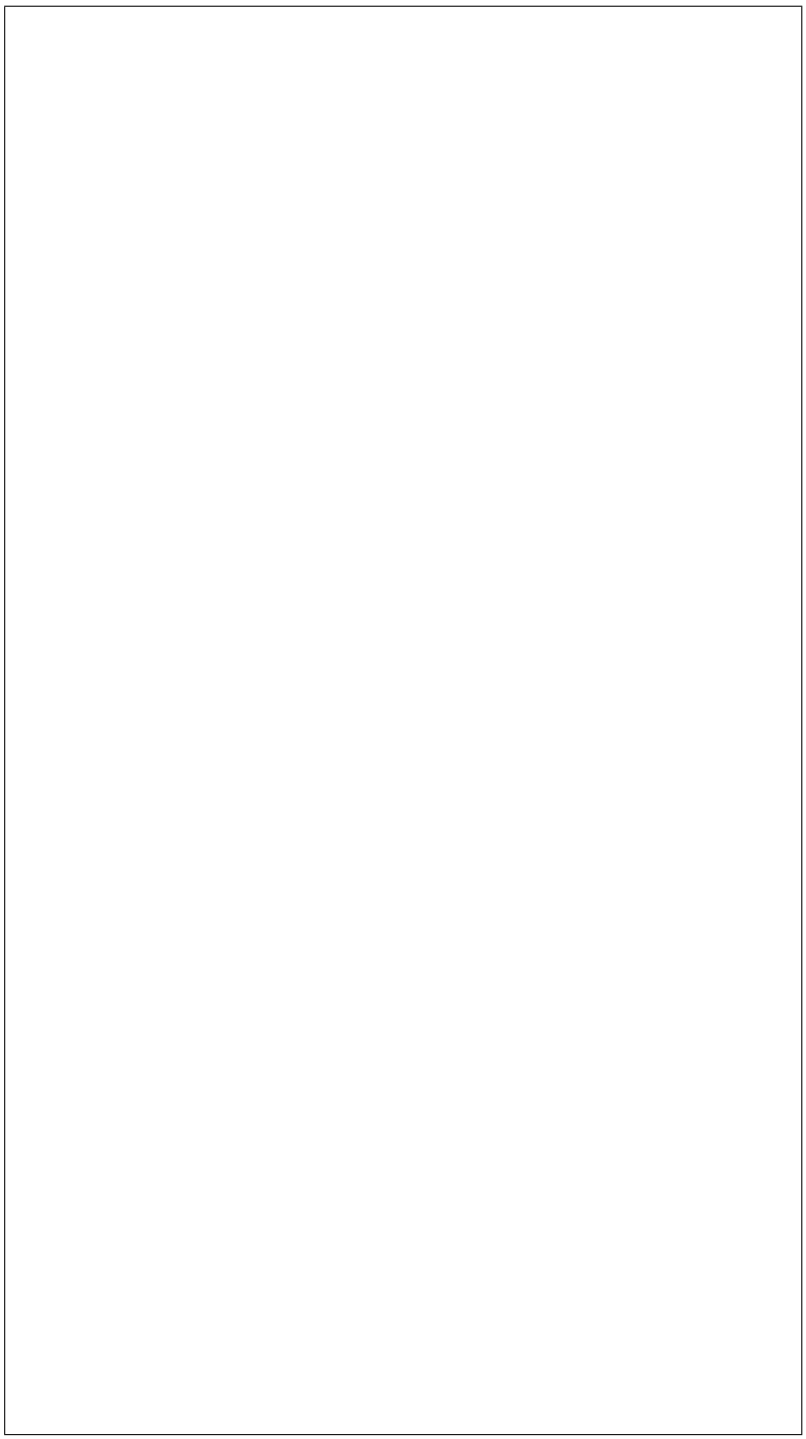
class EnglishTranslator : public Translator

void printError(const Exception& e) { Translator& t = Settings::getInstance().getTranslator();

Zadanie 2 - obiektowe wzorce projektowe (2pkt)

Obiekt State reprezentuje stan w grze w warcaby, tzn. położenie pionów, damek oraz informacja o tym, kto się rusza. Obiekt States = std::vector<State> to kolekcja stanów. Dostarczamy dwóch różnych wersji gry: warcaby włoskie i warcaby niemieckie i hiszpańskie, wersje używają planszy 8×8, różnią się możliwością bicia pionem do tyłu oraz możliwymi ruchami damek, dlatego metody 'correctMoves' dla różnych klas mogą zwracać różną liczbę obiektów. Zmiany reguł gry nie wpływają na implementację metody 'nextMove', która z kolei zależy od algorytmu gracza. Dostarczamy 3 algorytmy: algorytm losowy - wybiera jeden z dostępnych ruchów losowo, algorytm AI - wybiera najlepszy ruch używając heurystyk oraz algorytm, który zwraca ruch dostarczony przez użytkownika. **Uprość strukturę klas, aby nie powielać kodu.** Nie implementuj metod nextMove oraz correctMoves.

```
using States = std::vector<State>;
class Player {
 virtual State nextMove(State s) = 0;
};
class ItalianAIPlayer : public Player {
public:
 ItalianAIPlayer();
 State nextMove(State s);
private:
 States correctMoves(State s);
};
class GermanAIPlayer : public Player {
public:
 GermanAIPlayer();
 State nextMove(State s);
private:
 States correctMoves(State s);
};
class SpanishAIPlayer : public Player {
public:
 SpanishAIPlayer();
 State nextMove(State s);
private:
 States correctMoves(State s);
};
class ItalianRandomPlayer : public Player {
public:
 ItalianRandomPlayer();
 State nextMove(State s);
private:
 States correctMoves(State s);
};
class GermanRandomPlayer : public Player {
public:
 GermanRandomPlayer();
 State nextMove(State s);
private:
 States correctMoves(State s);
};
class SpanishRandomPlayer : public Player {
public:
 SpanishRandomPlayer();
 State nextMove(State s);
private:
 States correctMoves(State s);
};
class ItalianHumanPlayer : public Player {
public:
 ItalianHumanPlayer();
 State nextMove(State s);
private:
 States correctMoves(State s);
};
class GermanHumanPlayer : public Player {
public:
 GermanHumanPlayer();
 State nextMove(State s);
private:
 States correctMoves(State s);
};
class SpanishHumanPlayer : public Player {
public:
 SpanishHumanPlayer();
 State nextMove(State s);
private:
 States correctMoves(State s);
};
```



Uwagi do prowadzącego:

Zadanie 3 - cykl życia oprogramowania (2pkt)

Proszę odpowiedzieć na następujące pytania:

1. Kiedy można powiedzieć o wymaganiu, że jest atomowe? Podaj przykład.
2. Co oznacza, że test jest typu black-box? Podaj przykład.

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?

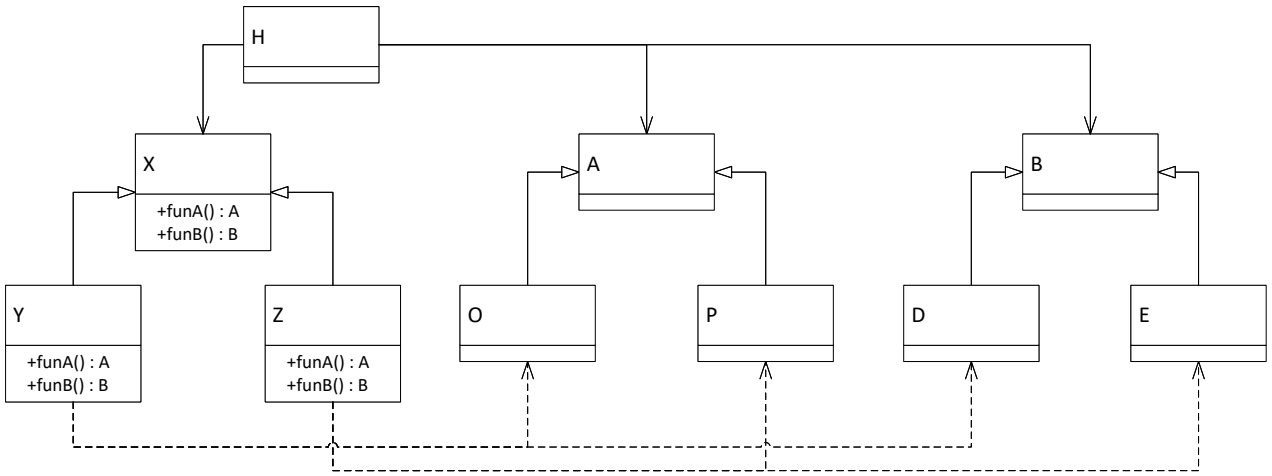
```
class Document
{
public:
  std::string title() const;
  Data data() const; //  szczegóły Data nie są istotne

  void combine(Data&& other)
  {
    //  szczegóły implementacji nie mają znaczenia
  }

  void print()
  {
    ColorPrinter p;
    //  szczegóły implementacji nie mają znaczenia
    //  będą użyte title() i data()
    p.print();
  }
};
```

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: