Informatyka 2 **Kolokwium II – Grupa β**

Imię i nazwisko: Nr indeksu: Data: 2018-01-19

Grupa (pn / wt / śr / czw / pt , godzina):

Uwagi

- Na niniejszym arkuszu umieszczaj końcowe rozwiązania. Miejscem na eksperymenty jest brudnopis.

Zadanie 1 [3 p.]

Funkcjonalność sprawdzania poprawności numeru telefonu (logika metody `UserSettingService::isValid()`) będzie wykorzystywana również w innych klasach naszego systemu, przy czym klasy te nie potrzebują znajomości pozostałego fragmentu klasy `UserSettingService`. Mając to na uwadze, dokonaj takiej modyfikacji poniższego kodu, aby spełniał on zasady SRP/ISP:

```
class MobileNumber { /* ... */ };
class User { /* ... */ };

class UserSettingService {
  public:
    void changeMobile(User user,
        MobileNumber number) {
      if (isValid(number)) {
        // Change user's mobile number.
      }
    }
    bool isValid(MobileNumber number) {
      // Verify if the number is valid.
    }
};
```

Miejsce na rozwiązanie:

Zadanie 2 [3 p.]

W obecnej wersji systemu klasa `HealthInsuranceSurveyor` odpowiada za weryfikację roszczenia od strony formalnej, natomiast klasa `ClaimApprovalManager` je rozpatruje od strony merytorycznej. Obecnie system działa poprawnie, jednak pojawiło się nowe wymaganie: klasa `ClaimApprovalManager` będzie musiała wspierać również rozpatrywanie roszczeń z tytułu zdarzeń drogowych – klasę `VehicleInsuranceSurveyor` (analogiczną do klasy `HealthInsuranceSurveyor`). Zaprojektuj nową wersję systemu w taki sposób, aby spełniał zasadę OCP:

```
class HealthInsuranceSurveyor {
  public:
    bool isClaimValid() {
      /* claim validation logic */
      return true;
    }
};
```

```
class ClaimApprovalManager {
  public:
    void processHealthClaim(HealthInsuranceSurveyor
        surveyor) {
      if (surveyor.isClaimValid()) {
        /* approve and process further... */
      }
    }
};
```

Zadanie 3 [3 p.]

Wyjaśnij dlaczego poniższy kod nie spełnia zasady LSP oraz zaproponuj (opisz) sposób jego poprawy tak, aby spełniał tę zasadę:

```
class Bird {
  public:
    virtual void setLocation(double longitude, double latitude) = 0;
    virtual void setAltitude(double altitude) = 0;
};

class Owl : public Bird {
  public:
    void setLocation(double longitude, double latitude) override { /* ... */ }
    void setAltitude(double altitude) override { /* ... */ }
    // ...
};

class Penguin : public Bird {
  public:
    void setLocation(double longitude, double latitude) override { /* ... */ }
    void setAltitude(double altitude) override { throw std::logic_error("can't fly"); }
    // ...
};
```

Zadanie 4 [3 p.]

Zmodyfikuj poniższy kod tak, aby spełniał zasady SRP, OCI i ISP:

```
class Workable {
  public:
    virtual bool canCode() = 0;
    virtual void code() = 0;
    virtual void test() = 0;
    virtual ~Workable() = default;
};

class ProjectManagement {
  public:
    void processCode(Workable& member) {
      if (member.canCode()) {
        member.code();
      }
    }
};
```

```
class Programmer : public Workable {
    public:
      bool canCode() override { return true; }
      void code() override { /* ... */ }
      void test() override { /* ... */ }
};

class Tester : public Workable {
    public:
      bool canCode() override { return false; }
      void code() override {
          throw std::logic_error("can't code");
      }
      void test() override { /* ... */ }
};
```