Szoftver tesztelés projekt

Árvai Dávid

Számítástechnika IV.A

Bevezetés

Ez a dokumentáció egy Python nyelven írt, egyszerű alkalmazotti Employee rendszer és annak unit tesztjeihez kapcsolódik. A kód célja, hogy rövid áttekintést nyújtson a kódban megvalósított alkalmazotti kezelés, fizetszámítás és csapatvezetői funkciók unit tesztjeiről. A tesztek biztosítják, hogy az alkalmazottak adatai helyesen kerülnek feldolgozásra, a csapattagok száma alapján megfelelő bónusz számítás történik, valamint az email értesítések pontosan működnek.

Motivációk

A kód célja, hogy ellenőrizze az alkalmazottak kezelésére, fizetésének kiszámítására és a csapatvezetői funkciók működésére vonatkozó logikát. A tesztek segítségével:

- Biztosítja, hogy az alkalmazottak adatai (pl. név, születési dátum, belépési dátum, alapbér) helyesen kerülnek feldolgozásra.
- Ellenőrizzük a csapatvezetők esetében a csapattagok számától függő bónusz számítását.
- Valid áljuk az email küldési funkciót, amely a kiszámított fizetésről tájékoztatást küld.
- Ellenőrizi, hogy a keresési műveletek (alkalmazott megtalálása, csapat tagjainak lekérdezése) megfelelően működnek.

Telepítés

A tesztek futtatásához szükséges:

- Python 3 telepítése (ajánlott a 3.6 vagy újabb verzió).
- A kód és a tesztesetek egyaránt a Python beépített `unittest` modulját használják, így nincs szükség további könyvtárak telepítésére.

Adatstruktúra a Repository-ban

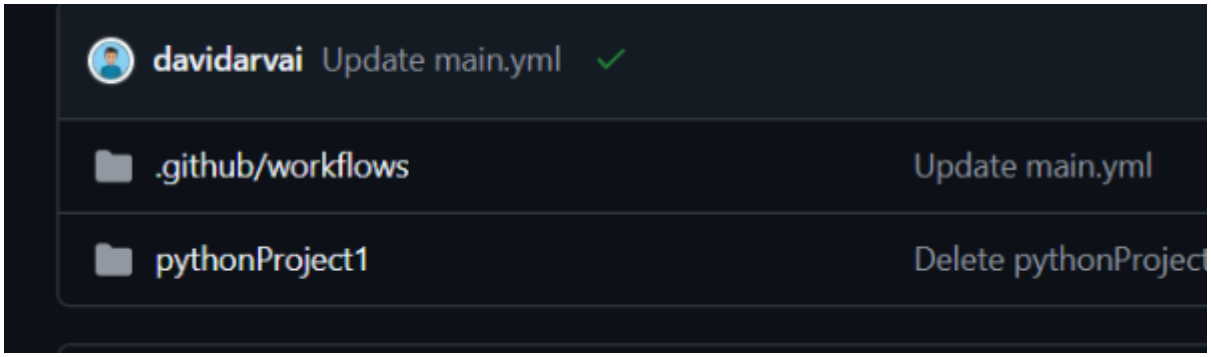
A repository az alábbi fő komponensekből épül fel:

Fő kódrész: Az alkalmazottak kezeléséért felelős osztályok:

- Employee: Az alkalmazott objektum, mely tartalmazza a személyes adatokat, alapbért és a csapattagok listáját.
- EmailSender: Szimulálja az email küldési folyamatot.
- EmployeeRelationsManager: Az alkalmazottak közötti kapcsolatok, csapatvezetői funkciók kezelését végzi.
- EmployeeManager: Az alkalmazottak fizetésének kiszámításáért és email értesítéséért felelős.

Teszt esetek: A `unittest` modul segítségével írt két teszt osztály:

- TestEmployeeRelationsManager: Az EmployeeRelationsManager funkcionalitásait teszteli.
- TestEmployeeManager: Az EmployeeManager működését ellenőrzi.



A kód a `pythonProject1/`Szoftver_Teszteles_Project.py fájlban található. A `.github/workflows` mappa a GitHub Actions CI/CD beállításait tartalmaza, amely segíthet a kód automatikus tesztelésében.

Miért a unittest?

- **Egyszerűség**: A `unittest` része a Python standard könyvtárának, így nincs szükség plusz telepítésre.
- **Könnyű integráció**: Számos Python IDE beépítve támogatja.
- **Széles körben használt**: Kiterjedt dokumentáció és közösségi támogatás érhető el.

Fejlesztett Unit Tesztek Leírása

test_team_leader_exists

- **Bemenet:**
  - Név: `"John Doe"`
  - Születési dátum: `"31.01.1970"`
- **Elvárt kimenet:**
  - `True`
- **Leírás:**
  Ez a teszt ellenőrzi, hogy a find_employee metódus és az is_team_leader metódus megfelelően azonosítja-e, hogy a megadott adatokkal rendelkező alkalmazott csapatvezető-e.

```
def test_team_leader_exists(self):  davidarvai
    question = "Kérdés: Van-e egy csapatvezető John Doe névvel, akinek a születési dátuma 31.01.1970?"
    print(question)
    result = erm.is_team_leader( name: "John Doe",  birthdate: "31.01.1970")
    print("Válasz:", result)
    self.assertTrue(result)
```

```
[START] EmployeeRelationsManager test case: test_team_leader_exists
Kérdés: Van-e egy csapatvezető John Doe névvel, akinek a születési dátuma 31.01.1970?
Válasz: True
[END] EmployeeRelationsManager test case: test_team_leader_exists
```

test_john_doe_team_members

- **Bemenet:**
  - Csapatvezető: `"John Doe"` (születési dátum: `"31.01.1970"`)
- **Elvárt kimenet:**
  - Lista, mely tartalmazza: `["Myrta Torkelson", "Jettie Lynch"]`
  - A lista hossza pontosan 2.
- **Leírás:**
  A teszt ellenőrzi, hogy a `get_team_members` metódus visszaadja-e a helyes csapattagok listáját John Doe esetében.

```python
def test_john_doe_team_members(self):  davidarvai
    question = "Kérdés: John Doe csapattagjai: Myrta Torkelson és Jettie Lynch?"
    print(question)
    team_members = erm.get_team_members( team_leader_name: "John Doe",  team_leader_birthdate: "31.01.1970")
    print("Válasz:", team_members)
    self.assertIn( member: "Myrta Torkelson", team_members)
    self.assertIn( member: "Jettie Lynch", team_members)
    self.assertEqual(len(team_members),  second: 2)
```

```
[START] EmployeeRelationsManager test case: test_john_doe_team_members
Kérdés: John Doe csapattagjai: Myrta Torkelson és Jettie Lynch?
Válasz: ['Myrta Torkelson', 'Jettie Lynch']
[END] EmployeeRelationsManager test case: test_john_doe_team_members
```

test_tomas_not_in_john_doe_team

- **Bemenet:**
  - Csapatvezető: `"John Doe"` (születési dátum: `"31.01.1970"`)
  - Keresett tag: `"Tomas Andre"`
- **Elvárt kimenet:**
  - A visszaadott lista nem tartalmazza a `"Tomas Andre"` nevet.
- **Leírás:**
  A teszt célja annak ellenőrzése, hogy a `get_team_members` metódus nem tartalmazza a nem csapattagként kapcsolt alkalmazottak nevét.

```python
def test_tomas_not_in_john_doe_team(self):  davidarvai
    question = "Kérdés: Van-e Tomas Andre John Doe csapatában?"
    print(question)
    team_members = erm.get_team_members( team_leader_name: "John Doe",  team_leader_birthdate: "31.01.1970")
    print("Válasz:", team_members)
    self.assertNotIn( member: "Tomas Andre", team_members)
```

```
[START] EmployeeRelationsManager test case: test_tomas_not_in_john_doe_team
Kérdés: Van-e Tomas Andre John Doe csapatában?
Válasz: ['Myrta Torkelson', 'Jettie Lynch']
[END] EmployeeRelationsManager test case: test_tomas_not_in_john_doe_team
```

test_gretchen_base_salary

- **Bemenet:**
  - Alkalmazott: `"Gretchen Walford"`
- **Elvárt kimenet:**
  - Az alkalmazott `base_salary` értéke: `4000`.
- **Leírás:**
  Ez a teszt ellenőrzi, hogy a `find_employee` metódus segítségével megtalálható-e az alkalmazott, és hogy az alapbér értéke megfelel-e az elvárt értéknek.

```
def test_gretchen_base_salary(self):  davidarvai
    question = "Kérdés: Gretchen Walford alapbére 4000$?"
    print(question)
    employee = erm.find_employee("Gretchen Walford")
    base_salary = employee.base_salary if employee else None
    print("Válasz:", base_salary)
    self.assertIsNotNone(employee)
    self.assertEqual(base_salary, second: 4000)
```

```
[START] EmployeeRelationsManager test case: test_gretchen_base_salary
Kérdés: Gretchen Walford alapbére 4000$?
Válasz: 4000
[END] EmployeeRelationsManager test case: test_gretchen_base_salary
```

test_tomas_not_team_leader

- **Bemenet:**

    - Alkalmazott: `"Tomas Andre"` (születési dátum: `"12.12.1975"`)
- **Elvárt kimenet:**
    - `is_team_leader` metódus: `False`
    - `get_team_members` metódus: üres lista (`[]`)
- **Leírás:**
  A teszt célja annak ellenőrzése, hogy Tomas Andre nem csapatvezető, és hogy nincs hozzárendelve csapattag listája.

```
def test_tomas_not_team_leader(self):  davidarvai
    question = "Kérdés: Tomas Andre csapatvezető-e, és üres-e a csapattag lista?"
    print(question)
    is_leader = erm.is_team_leader( name: "Tomas Andre",  birthdate: "12.12.1975")
    team_members = erm.get_team_members( team_leader_name: "Tomas Andre",  team_leader_birthdate: "12.12.1975")
    print("Válasz: Csapatvezető:", is_leader, "Csapattagok:", team_members)
    self.assertFalse(is_leader)
    self.assertEqual(team_members,  second: [])
```

```
[START] EmployeeRelationsManager test case: test_tomas_not_team_leader
Kérdés: Tomas Andre csapatvezető-e, és üres-e a csapattag lista?
Válasz: Csapatvezető: False Csapattagok: []
[END] EmployeeRelationsManager test case: test_tomas_not_team_leader
```

test_jude_not_in_database

- **Bemenet:**
    - Keresett alkalmazott neve: `"Jude Overcash"`
- **Elvárt kimenet:**
    - `False`, mivel az alkalmazott nem szerepel az adatbázisban.
- **Leírás:**
  A teszt azt vizsgálja, hogy a `is_employee_in_database` metódus helyesen ad-e választ, amikor egy nem létező alkalmazottat keresünk.

```
def test_jude_not_in_database(self):  davidarvai
    question = "Kérdés: Jude Overcash szerepel-e az adatbázisban?"
    print(question)
    in_database = erm.is_employee_in_database("Jude Overcash")
    print("Válasz:", in_database)
    self.assertFalse(in_database)
```

```
[START] EmployeeRelationsManager test case: test_jude_not_in_database
Kérdés: Jude Overcash szerepel-e az adatbázisban?
Válasz: False
[END] EmployeeRelationsManager test case: test_jude_not_in_database
```

test_non_team_leader_salary

- **Bemenet:**
  - Alkalmazott: `"Mark Spencer"`
  - Belépési dátum: `"10.10.1998"`
  - Alapbér: `1000`
- **Elvárt kimenet:**
  - Kiszámított fizetés: `3000`
    (Számítás: alapbér + (2018 - 1998) * 100 = 1000 + 20×100 = 3000)
- **Leírás:**
  A teszt ellenőrzi, hogy a nem csapatvezető esetében a fizetés kiszámítása a meghatározott képlet szerint történik-e.

```
def test_non_team_leader_salary(self):  davidarvai
    question = "Kérdés: Mark Spencer fizetése 3000$ (nem csapatvezető)?"
    print(question)
    salary = em.calculate_employee_salary( name: "Mark Spencer",  hire_date: "10.10.1998",  base_salary: 1000)
    print("Válasz:", salary)
    self.assertEqual(salary,  second: 3000)
```

```
[START] EmployeeManager test case: test_non_team_leader_salary
Kérdés: Mark Spencer fizetése 3000$ (nem csapatvezető)?
Válasz: 3000
[END] EmployeeManager test case: test_non_team_leader_salary
```

test_team_leader_salary

- **Bemenet:**
  - Alkalmazott: `"Lisa Monroe"`
  - Belépési dátum: `"10.10.2008"`
  - Alapbér: `2000`
  - Csapattagok száma: 3 (Member A, Member B, Member C)
- **Elvárt kimenet:**
  - Kiszámított fizetés: `3600`
    (Számítás: alapbér + (2018 - 2008) * 100 = 2000 + 10×100 = 3000; csapatvezetői bónusz: 3×200 = 600; összesen: 3000+600=3600)
- **Leírás:**
  A teszt célja annak ellenőrzése, hogy a csapatvezetői fizetés kiszámítása megfelelően alkalmazza a csapattagok számán alapuló bónuszt.

```
def test_team_leader_salary(self):  davidarvai
    question = "Kérdés: Lisa Monroe fizetése 3600$ (csapatvezető, 3 csapattaggal)?"
    print(question)
    salary = em.calculate_employee_salary( name: "Lisa Monroe",  hire_date: "10.10.2008",  base_salary: 2000)
    print("Válasz:", salary)
    self.assertEqual(salary,  second: 3600)
```

```
[START] EmployeeManager test case: test_team_leader_salary
Kérdés: Lisa Monroe fizetése 3600$ (csapatvezető, 3 csapattaggal)?
Válasz: 3600
[END] EmployeeManager test case: test_team_leader_salary
```

test_salary_calculation_email

- **Bemenet:**
  - Alkalmazott: `"Lisa Monroe"`
  - Belépési dátum: `"10.10.2008"`
  - Alapbér: `2000`
- **Elvárt kimenet:**

- A konzolra kiírt üzenet pontosan:
  `"Email sent to Lisa Monroe with message: Your calculated salary is 3600$."`

- **Leírás:**
  Ez a teszt ellenőrzi, hogy a `calculate_salary_and_send_email` metódus először helyesen számolja-e ki a fizetést, majd a `EmailSender.send_email` metódus meghívásával a megfelelő értesítő üzenetet küldi el.

```
def test_salary_calculation_email(self):  davidarvai
    question = "Kérdés: Az e-mail küldés megfelelő üzenettel történik-e?"
    print(question)
    captured_output = StringIO()
    with contextlib.redirect_stdout(captured_output):
        em.calculate_salary_and_send_email( name: "Lisa Monroe",  hire_date: "10.10.2008",  base_salary: 2000)
    output = captured_output.getvalue().strip()
    print("Válasz:", output)
    expected_message = "Email sent to Lisa Monroe with message: Your calculated salary is 3600$."
    self.assertEqual(output, expected_message)
```

```
[START] EmployeeManager test case: test_salary_calculation_email
Kérdés: Az e-mail küldés megfelelő üzenettel történik-e?
Válasz: Email sent to Lisa Monroe with message: Your calculated salary is 3600$.
[END] EmployeeManager test case: test_salary_calculation_email
```