Python programozás –I. zárthelyi minta

Feladat: Adott egy SCRUM alapú projekt menedzsment rendszer exportja, amely tartalmazza egy fejlesztő csapat sprintek során elvégzett feladatait! Az adatokat szeretnénk átadni egy burndown chartot megjelenítő rendszer számára, amihez konvertálni kell a jelenlegi formátumot a célrendszer által elvárt formátumba. Mivel várhatóan több hasonló konverzióra lesz szükség, ezért a cél a megfelelő osztályok megvalósítása, amelyek képesek a főbb adatok kezelésére. Ennek érdekében a következő lépésekre lesz szükség:

a) Felhasználók tárolása:
   - Készíts egy User osztályt, amely tárolni fogja az egyes felhasználók főbb adatait: azonosító, név, jelszó és szerepkör. Az osztály biztosítja, hogy a print függvény paramétereként a következő információk jelenjen meg: 4ke83hsejs, John (developer)
   - Készíts egy Users osztályt, amely konstruktorában megkapja az exportból a users kulcsú struktúrát, amely alapján képes előállítani a User típusú objektumokat és egy belső listában tárolja azokat
   - A Users osztály biztosítson egy metódust, amely a paraméterben megadott id alapján visszaadja megfelelő User objektumot

b) Feladatok tárolása:
   - Készíts egy Task osztályt, amely tárolni fogja a feladatok azonosítóját és megnevezését, a feladatért járó storypointot, valamint a feladaton dolgozott fejlesztők munkaóráit.
   - A Task osztály biztosítson egy metódust, amely visszaadja a ledolgozott órák összegét

c) Sprintekhez kapcsolódó adatok tárolása:
   - Készíts egy Sprint osztályt, amely a konstruktorba megkapja az export sprints kulcsú elemét, amely alapján tárolja a sprint azonosítóját, valamint a sprinten belüli feladatok alapján létrehoz egy-egy Task osztályt A Sprint osztály biztosítson egy metódust, amely visszaadja a sprinthez tartozó storypontok összegét

(d) Konverter osztály megvalósítása:

   - Készíts egy Converter osztályt, amely a konstruktorba megkapja az export struktúrát, amely alapján létrehoz egy Users objektumot, valamint egy Sprint objektumokból álló listát, amely az id szerint van rendezve
   - A Converter osztály biztosítson egy metódus, amely a tárolt adatokat a következő formátumban adja vissza:

```
{
"sprints" : [
    { "_id" : "2", "sumStory" : 50, "sumHours" : 32, "developers" : [ "John" , "Robert"]},
    { "_id" : "3", "sumStory" : 56, "sumHours" : 43, "developers" : [ "Bob" , "Robert"]},
   ]
 }
```

ahol az _id a sprint azonosítója, a sumStory a storypontok összege, a sumHours a ledolgozott órák összege, a developers, pedig a sprinten dolgozott fejlesztők neve.

```
export = {
   "users": [
      { "_id" : "4ke83hsejs", "name" : "John",   "role" : "developer"},
      { "_id" : "34rkmlqek2", "name" : "Kevin",  "role" : "developer"},
      { "_id" : "83i2kkwwj3", "name" : "Joe",    "role" : "developer"},
      { "_id" : "12kw4jwmcq", "name" : "Bob",    "role" : "developer"},
      { "_id" : "4i5ii32313", "name" : "Robert", "role" : "developer"},
      { "_id" : "lkr3jwl4k5", "name" : "Mary",   "role" : "manager"},

    ],
   "sprints": [
      { "_id": "2",
        "tasks":[ { "_id" : "T0005", "storypoint" : 60, "task" : "Seach",
                 "hours": [ ("4ke83hsejs", 23), ("83i2kkwwj3", 15), ("4i5ii32313", 11)]},
               { "_id" : "T0006", "storypoint" : 20, "task" : "Filter",
                "hours": [ ("4ke83hsejs", 17), ("4i5ii32313", 12)]},
               { "_id" : "T0007", "storypoint" : 40, "task" : "List items",
                "hours": [ ("4i5ii32313", 3)]},
        ]
      },{ "_id": "3",
        "tasks":[ { "_id" : "T008", "storypoint" : 25, "task" : "Basket",
                 "hours": [ ("4ke83hsejs", 6), ("83i2kkwwj3", 15), ("4i5ii32313", 23)]},
               { "_id" : "T0009", "storypoint" : 20, "task" : "Subscription",
                "hours": [ ("4ke83hsejs", 3), ("83i2kkwwj3", 24), ("4i5ii32313", 23)]},
               { "_id" : "T0010", "storypoint" : 35, "task" : "Order",
                "hours": [ ("83i2kkwwj3", 13), ("4i5ii32313", 5)]}
        ]
      },{ "_id": "1",
        "tasks":[ { "_id" : "T0001", "storypoint" : 50, "task" : "Login",
                 "hours": [ ("4ke83hsejs", 13), ("83i2kkwwj3", 5), ("4i5ii32313", 11)]},
               { "_id" : "T0002", "storypoint" : 20, "task" : "Registration",
                "hours": [ ("4ke83hsejs", 28), ("4i5ii32313", 12)]},
               { "_id" : "T0003", "storypoint" : 30, "task" : "Menu",
                "hours": [ ("83i2kkwwj3", 7), ("4i5ii32313", 5)]},
               { "_id" : "T0004", "storypoint" : 40, "task" : "Design",
                "hours": [ ("4ke83hsejs", 17)]}
        ]
      },
   ]

}
```

Megoldás:

Az a) feladat 1. pontjának megfelelően létrehozásra kerül a User osztály. A print() függvény megfelelő eredményéhez a __str__ metódus felülírása szükséges.

```
class User:


  def __init__(self, _id, name, role):
      self._id=_id
      self.name = name
      self.role = role

   def __str__(self):
      return "{0}, {1} ({2})".format(self._id,self.name, self.role)
```

Az a) feladat 2. és 3. pontjának megfelelően létrehozásra kerül a Users osztály, ahol a konstruktorba átadott listát szűrni kell a role szerint. Az id alapján egy User objektumot visszadó metódus is megoldható lambda függvénnyel, vagy ciklussal is.

```
class Users:

   def __init__(self, usersList):
      self.users = []
      for user in list(filter(lambda x: x["role"]=="developer",usersList)):
         self.users.append(User(user["_id"], user["name"], user["role"]))

   def getUserById(self, id):
      return list(filter(lambda x: x._id==id, self.users))[0]

#Teszt
users = Users(export["users"])
#print(users.getUserById("123kewkmfn"))
```

A b) feladatnak megfelelő Task osztály létrehozása. A ledolgozott órák összegzésénél figyelni kell, hogy a lista Tuple-ket tartalmaz, amiket nem kulccsal indexelünk, hanem számmal.

```
class Task:

   def __init__(self, _id, sp, task, hours):
      self._id = _id
      self.sp = sp;
      self.task = task
      self.hours = hours

   def getWorkedHours(self):
      sum = 0
      for item in self.hours:
         sum+=item[1]
      return sum
```

A c) feladatnak megfelelően a Sprint osztály létrehozása. A konstruktorba létre kell hozni egy listát, ami a Task objektumokat fogja tartalmazni. A Storypontok összegzésénél figyelni kell, hogy objektumra hivatkozunk!

```
class Sprint:
   def __init__(self, sprint):
      self._id = sprint["_id"]
      self.tasks = []
      for task in sprint["tasks"]:
         self.tasks.append(Task(task["_id"], task["storypoint"],task["task"],task["hours"]))

   def getStoryPointSum(self):
      sum = 0
      for task in self.tasks:
         sum+= task.sp
      return sum
```

A d) feladatnak megfelelően a Converter osztály létrehozása. A konstruktorban megkapja a teljes export struktúrát. Az export[„users”] alapján a Users objektum egyből létrehozható, viszont a Sprint osztályoknak egy külön listát kell létrehozni. Már itt a feltöltésnél is lehet rendezni az export[„sprints”] listát a sorted() függvénnyel, de a már kész listán is meg lehet hívni a sort()-ot a megfelelő beállításokkal.

```
class Converter:
   def __init__(self,export):
      self.users = Users(export["users"])

      self.sprints= []
      for sprint in sorted(export["sprints"], key = lambda x: x["_id"]):
         self.sprints.append(Sprint(sprint))
```

Az utolsó pont a convert() függvény megvalósítása, ahol a cél a következő struktúra előállítása:

```
{
"sprints" : [
      { "_id" : "2", "sumStory" : 50, "sumHours" : 32, "developers" : [ "John" , "Robert"]},
      { "_id" : "3", "sumStory" : 56, "sumHours" : 43, "developers" : [ "Bob" , "Robert"]},
     ]
}
```

Itt létre kell hozni egy dictionary változót, amit feltöltünk a megfelelő adatokkal. Először létre kell hozni a sprints kulccsal a listát, majd ehhez fűzzük hozzá a dictionary-ket. Ehhez végig kell menni a konstruktorba tárolt sprinteken, ahol a taskok alapján meg lehet határozni a sumHours értéket, valamint a taszkok alapján kell összegyűjteni a fejlesztők neveit is. Itt lehet használni Set-et, de jó sima listával is, ahol minden iterációban ellenőrizzük, hogy az adott név benne van-e már a listába. Fontos itt is, hogy a taszkon belül a hours lista Tuple-ket tartalmaz, ahol most a 0. indexű elemre van szükség, amely alapján megkapjuk az id-t, és itt használhatjuk az users osztály getUserById() metódusát. Ennek visszatérési értéke egy User objektum, amiből csak a name adattagra van szükség. Ezt követően az append()-nél láthatjuk, hogy tudjuk összetenni a fenti struktúrát.

```
    def convert(self):
        result = {}
        result["sprints"] = []
        for sprint in self.sprints:
            sumHours = 0
            developers = set()
            for task in sprint.tasks:
                sumHours+=task.getWorkedHours()
                for hour in task.hours:
                    developers.add(self.users.getUserById(hour[0]).name)

            result["sprints"].append(
                {
                    "_id" : sprint._id,
                    "sumStory": sprint.getStoryPointSum(),
                    "sumHours": sumHours,
                    "developers": developers
                }
            )
        return result

#Test Converter
converter = Converter(export)
print(converter.convert())
```