Almuthanna Jamal Aljajan

Rapport

Oppgave 2.0

Du skal skrive en liten rapport om prosjektet.

UML med forklaring:

Oppgaven spør om å lage en applikasjon for registrering av dyreobservasjoner. Etter jeg har lest oppgaven, tenkte at jeg skal først en Animal-klasse som skal inneholde følgende instans variablene: String name, gender, diet , pictureURL;. og bestemte å la denne klassen skal være abstract. Grunnen til det, at den klassen finnes i programmet for å abstrahere de egenskapene som er felles for 3 grupperinger av dyr og at jeg skal ikke operette noe objekter av den klassen. Deretter tenkte jeg å presentere de 3 grupperinger av dyr med klasser og la de klassene arver fra Animal. Så tenkte å definere en Bird klasse med følgende spesifikke egenskaper: double wingLength, String abilityToFly, en Amphibian klasse med følgende spesifikke egenskaper: String breathingSystem, int numberOfLegs , og en Invertebrate klasse med følgende spesifikke egenskaper: String invertebrateType, String color. Der oppgaven spør også om å registrere den lokasjonen for observasjonene og at den lokasjonen skal inneholde et navn, samt lengde- og breddegrad og at den knytter til planeten og biom som befinner seg i. så tenkte jeg å lage først en Planet klasse med følgende instans variablene: String planetName, String planetSystemName ,Og å lage også en Biom Klasse med følgende instans variablene: String name, type.Til slutt bestemte å lage en Location klasse med følgende instans variablene: String locationName, List<Biom> biomes , Planet planet, double latitude, double longitude. Grunnen til at jeg bestemte å lage en Liste med biomer, fordi der står i oppgaven at en lokasjon kan være knyttet til flere biomer samtidig. Til slutt definerte jeg en Observation klasse med følgende instans variablene: String id, String name, Animal animal, Location location, LocalDate date, int numberOfAnimals, String pictureURL, String comment.

NB: I alle klassene som er beskrevet opp over, skal jeg lage konstruktører og get- og set-metoder.

Forklaring av hvordan jeg har løst oppgavene:

Oppgaven 3.2

Først lagte jeg et nytt Gradle-prosjekt og fikset java versjonen og dependencies. I model pakken definerte de klassene som er beskrevet i UML-skisse og implementerte toString() i alle klasser. Til slutt bestemte å la klassene Animal og Observation implementerer Comparable interfacet og lagte implementasjon av metoden compareTo basert på navn i begge klassene.

Oppgaven 3.3

Jeg har opprettet en Main-klasse og definerte 6 objektene av typen Observation og skrivet de ut i konsollen.

Oppgaven 3.4

Jeg har definerte IMapOfLifeRepository med følgende abstrakte metoder:

```
public interface IMapOfLifeRepository {
```

```
    void readFromFile(String fileName);

    void writeToFile(List<Observation> observationList, String fileName);

    Observation getObservation(String observationName);

    ArrayList<Observation> getObservations();

    Animal getAnimal(String observationName);

    void deleteObservation(String observationName);

    void creatObservation(String[] formList);

    void updateObservation(String observationName, String[] formList);
}
```

og lagte jeg MapOfLifeCSVRepository klasse som skal implementere dette interfacet. Når det kommer for å lese og skrive til filen, bestemte jeg å velge CSV isteden av Json, fordi å jobbe med Json er det enkelt og greit Json og vi trenger ikke så mye å kode for oppnå dette, men med CSV filene vi må gjøre alt på egen hånd, grunnen til det bestemte å bruke CSv.

I tillegg valgte jeg å bruke List for å holde dataene på isteden av HashMap. Egentlig det er ikke så mye forskil mellom List og HashMap. Den eneste er i HashMapet vi bruker nøkkelen for å aksessere verdiene, men i Listen vi bruker indeksen og begge to kan oppnå oppgaven. På grunn av det valgte listen fordi den er lettest å jobbe med.

I denne oppgaven jeg brukte kode eksempler for **forelesning 11** for å generere automatisk navn og id for å slippe å lage det på egen hånd, men resten av oppgaven jeg gjorde seg selv og fra mine forrige obligene.

Til slutt brukte MapOfLifeCSVRepository for å skrive de objektene i Main til CSV-filen og leste den tilbake.

Oppgaven 3.5

I siste oppgaven jeg velgte å bruke Javalin isteden av JavaFx, fordi jeg har ingen idea om hvordan JavaFX fungerer.

Så definerte jeg Application klasse og MapOfLifeController i controller pakken.

I Application brukte get() metoder for å definere pathene og binde de med vue-filene og definerte også api-pathene og la MapOfLifeController håndtere de forskjellige requester.

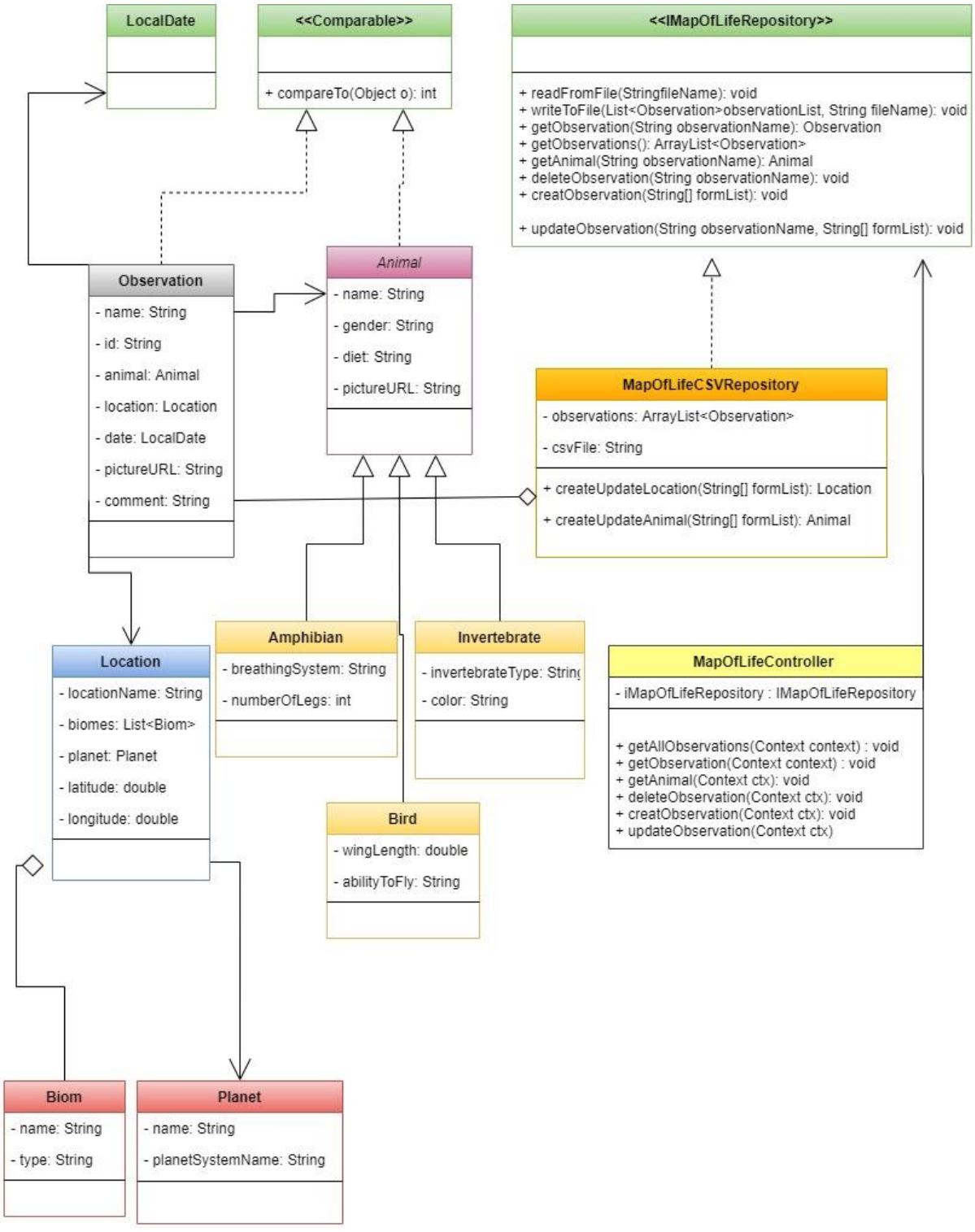
Bonusoppgaver

I bonusoppgavene måtte jeg å bruke de vue filene fra oblig-5 for å kunne vise detaljer for observasjoner, ett dyr, slette observasjon, opprette og oppdatere en ny observasjon. fordi jeg har ikke gode kunsskap for å lage mine egne vue-filen, så jeg bare har fikset det for å få det fungert.

Oppgave 3.1 - UML

Basert på beskrivelsen over, lag en UML-skisse som viser de klassene du mener er nødvendig for å lage modellen for systemet.

NB: Denne UML-skissen er er den første versjonen, etter at jeg har begynt å løse bonusoppgaver i obligen, så måtte jeg å oppdatere noen nye instans variabler og noen metoder for å løse det, så jeg har ikke oppdaterte UML-skissen også.



Relasjonen mellom klassene

- - tegn indikerer at variablene eller metodene er definert som **private**.
- MapOfLifeController klassen innholder instans varibel av IMapOfLifeCSVRepository interfacet derfor har jeg en **assosiasjon** fra klassen MapOfLifeController til interfacet IMapOfLifeCSVRepository
- klassen MapOfLifeCSVRepository **implementerer** IMapOfLifeCSVRepository interfacet, så indikerer implementasjon med en slik pil
- Location klassen innholder et ArrayList som holder objekter av typen Biom og dermed Location klassen **agggregerer** objekt av klassen Biom.
- MapOfLifeCSVRepository klassen innholder et ArrayList som holder objekter av typen Observation og dermed MapOfLifeCSVRepository klassen **agggregerer** objekt av klassen Observation.
- Location klassen innholder en instansvariabel av typen Planet og derfor har jeg en **assosiasjon** fra klassen Location til klassen Planet
- Observation klassen innholder en instansvariabel (objekt) av typen Location og derfor har jeg en **assosiasjon** fra klassen Observation til klassen Location
- Observation klassen innholder en instansvariabel (objekt) av typen Animal og derfor har jeg en **assosiasjon** fra klassen Observation til klassen Animal
- Observation klassen innholder en instansvariabel (objekt) av typen LocalDate og derfor har jeg en **assosiasjon** fra klassen Observation til klassen LocalDate
- Begge klassene Observation og Animal **implementerer** Comparable interfacet, så indikerer implementasjon med en slik pil
- Jeg definerte Animal klassen som **abstract** fordi den klassen finnes i programmet for å abstrahere de egenskapene som er felles for 3 grupperinger av dyr og at jeg skal ikke operette noe objekter av den klassen. og den klassen presenteres i UML-skissen ved å skrive navn på klassen i kursiv
- Klassene Amphibian, Invertebrate og Bird som inneholder spesifikke egne skaper, **arver** fra Animal, så indikerer arv med en slik pil 

Kilder

Forelesning07_Uke05_ArrayList_UML