UnitTest

Dog

1. Skriv klassen **Dog** med attributterne name og age.
2. Skriv en metode, der beregner hundens alder i hundeår (1 menneskeår = 7 hundeår).
3. Implementér funktionalitet til at finde den ældste hund i en liste.

JUnit Testkrav

- Test metoden, der beregner hundens alder med forskellige alder-værdier.
- Test metoden som finde den ældste hund i listen:
  - Når der er flere hunde i listen.
- Test getters for at sikre korrekt adgang til attributter.

Match

1. Skriv klassen **Match** med attributterne team1Score og team2Score.
2. Skriv en metode, der afgør vinderen af en kamp (eller om det er uafgjort).
3. Opret en liste af kampe og find den kamp med den største scoreforskel.

JUnit Testkrav

Du skal skrive unittest-klasser, som tester de forskellige funktionaliteter i Match-klassen. Dette inkluderer at:

- Teste metoden, der afgør vinderen af en kamp.
- Teste metoden, der finder den kamp med den største scoreforskel i en liste af kampe.
- Teste metoden, der sorterer en liste af kampe efter scoreforskel.

Recipe

1. Skriv klassen **Recipe** med attributterne name og ingredients.
2. Skriv en metode, der returnerer opskriften som en pæn tekst.
3. Skriv en metode, der tjekker, om en given ingrediens findes i opskriften.

JUnit Testkrav

- Skriv JUnit tests til at validere funktionerne i Recipe-klassen. Du skal teste:
    - Om du kan tilføje ingredienser korrekt.
    - Om du kan fjerne ingredienser korrekt.
    - Om systemet korrekt tjekker, om en ingrediens findes.
    - Om opskriften returneres korrekt som en formateret tekst.

Toy

1. Skriv klassen **Toy** med attributterne name og minimumAge.
2. Skriv en metode, der tjekker, om et barn på en given alder må lege med legetøjet.
3. Skriv klassen **ToyStore**, som holder styr på en liste af legetøj og returnerer det mest populære legetøj baseret på en popularitetsscore.

JUnit Testkrav

- Bekræft, at når et legetøj er tilføjet, kan det findes i butikken.
- Test, at Toy korrekt vurderer, om et barn på en given alder kan lege med legetøjet.
- Test, at metoden korrekt finder og returnerer det mest populære legetøj baseret på popularitetsscoren.

Flower

1. Skriv klassen **Flower** med attributterne species og color.
2. Skriv klassen **Garden**, som har en liste af blomster.
3. Skriv en metode, der returnerer alle blomster med en bestemt farve.

JUnit Testkrav

- Tilføj blomster med forskellige farver til haven, og kontroller, at metoden returnerer alle blomster, der matcher farven (f.eks. "Red").
- Test et scenarie, hvor der kun findes én blomst af en bestemt farve, og kontroller, at listen indeholder denne blomst.

Bonus Opgaver til de hurtigere:

Udvidelsemuligheder til Match:

- Brug en Scanner til at tilføje kampe dynamisk til ArrayList.
- Sortér kampene efter scoreforskel ved hjælp af Collections.sort().

Udvidelsesmuligheder til Recipe:

- Du kan tilføje et attribut for mængden af ingredienser. For eksempel, i stedet for bare at have en liste af ingredienser som String, kan du bruge et objekt, der holder både navnet på ingrediensen og mængden.

Udvidelsesmulighederne til Toy

- Tilføj muligheden for at kategorisere legetøj (f.eks. byggesæt, dukker, brætspil, elektronisk legetøj, osv.).
- Lad kunderne rate og anmelde legetøj. Brug et ratingsystem for at vise, hvilke legetøj der er mest populære.

Udvidelsesmuligheder til Flower

- Returner antallet af blomster i haven, evt. fordelt på art eller farve.
- Find ud af, hvilken farve der er mest repræsenteret i haven.
- Beregn den gennemsnitlige højde af blomster i haven.

- Simuler bestøvning mellem blomster for at oprette en ny blomst med kombinerede egenskaber.
- Fjern blomster, hvis haven bliver "overfyldt".