Milleaccendinis Världsrekord 🇮🇹

Milleaccendini har under en väldigt lång tid försökt att hamna i Guiness rekordbok. Till skillnad ifrån andra aspiranter har Milleaccendini hittat något som inte bara kommer sätta hans namn i boken, utan som även kommer sätta Italien på världskartan. Men han behöver vår hjälp att blanda ingredienserna, via objektorientering ska vi alltså hjälpa honom skapa den största coca-cola, mentos, nutella och durex explosionen!

Kravspec

Det har varit en aning svårt att tyda Milleaccendinis önskemål då han talar enbart italienska, men det vi kunde få ur honom är följande:

- Han vill ha ett program som blandar ingredienser till ett GUI.
- Ingredienserna måste vara korrekt angivna, då endast den rätta blandningen kommer skapa den explosion vi behöver.
- Ingredienslistan ska vara en Array.
- Ingredienslistan ska bestå av en Coca-Cola, en Mentos och en Nutella. Durexens syfte är endast att demonstrera hur kraftig vår coca-cola bomb kommer bli, men den måste också vara med.
- *Vid slutet av programmet ska en demo utav alla ingredienserna visas. Det här steget är redan löst.*

Jag kommer att beskriva allting i noggrann ordning, då Milleaccendini helst inte vill ha några misstag. (Om något inte görs exakt, så kommer inte programmet att starta. Men oroa dig inte, den här guiden beksriver allting i detalj)

1. Packa upp projektet med lämpligt program.

Förslagsvis Winrar, men Winzip fungerar det med. Projektet ligger i en mapp vid namn ItalianWorldRecord inuti zip filen. Lägg den mappen någonstans där du enkelt kan hitta den.

2. Öppna upp projektet i Eclipse eller IntelliJ

Skippa denna om du vet hur du öppnar andras projekt i just din IDE och gå vidare till steg 3!

Eclipse: File > Open Projects From File System, navigera och välj mappen ItalianWorldRecord.
Välj sedan mappen för projektet i package explorer/project explorer
Intellij : File -> Open, navigera och välj mappen ItalianWorldRecord.

3. Dags att objektorientera ingredienser!

I projektet finns 4 klasser. **Den enda du behöver arbeta i är Cauldron.class**, men tveka absolut inte på att kika över de andra klasserna senare. *(En av dem är väldigt lärorik inför Övning 2 bland annat)*

Notera att följa stegen, de kommer inte vara svårt men om du missar något utav dem kommer inte programmet kunna starta. Det är inte helt fel att analysera varför, men just nu är syftet att ge er en lättare förståelse av objektorientering.

Fortsätter på sida 3.

Följ dessa stegen:

1. Instansiera listan som ska innehålla våra ingredienser

Rad 9 har vi en Lista som innehåller ingredienser. **Listan är just nu endast en referens till objektet List**, och för att instansiera (*dvs skapa ett objekt av en klass baserad på List*) den behöver vi använda oss utav någon klass inom Java som ärver ifrån List.

Instansiera listan såhär(kopiera och klistra in på rad 10):

```
private List<Ingredient> ingredientList = new
ArrayList<>();
```

Oroa dig inte om det känns obegripligt det här kommer sätta sig med tiden, nästkommande steg är mycket enklare.

2. Nyttja en Setter metod

Rad 16 innehåller ett objekt av klassen Coca Cola, vars Sträng inte är satt.

Objektet i sig finns, men själva String-*en*(namnet på vår **Ingredient**) inuti är inte satt. Här nyttjar vi Setter metoden för att sätta strängen *ingredientName* i ***Ingredient klassen.***

Vi börjar med att använda oss utav objektet cocaCola. Du kan klistra in det här på rad 17:

```
cocaCola.setIngredientName("Coca Cola");
```

Här kallar vi på metoden setIngredientName, samt skickar in ett värde motsvarande en String. Metoden går alltså att använda till alla våra instanser av klassen Ingredient. Skulle vi vilja använda samma metod på ett annat objekt, gör vi precis samma sak fast vi byter bara ut *cocaCola* biten.

3. Skapa ett Ingredient objekt via konstruktorn

Nästa ingrediens är inte heller där, vi saknar Nutella-n som kommer slöa ner Mentos tabletten innan den faller ner i flaskan.

För att skapa en instans av **Ingredient.class** kan vi nyttja **konstruktorn** som vi kan kalla på ur själva klassen Ingredient.class. Lättast här är att använda den konstruktor som också fyller vårt syfte då vi behöver en String. *(Men att skapa via den tomma konstruktorn och sedan använda en Setter så som vi gjorde med cocaCola är också helt okej, men det kommer bli mer kod)*

Nedanstående kan du klistra in på rad 18:

```
Ingredient nutella = new Ingredient("Nutella");
```

Här använder vi *konstruktorn* i klassen Ingredient för att *instansiera* en Ingredient i programmet. Precis som hur vi instansierade vår List tidigare, fast nu instansierar vi objektet Ingredient som dessutom tillhör vårt projekt. Du kan alltså ändra den här klassen om du vill ändra något i den.

Fortsätter på sida 4.

4. Lägg till alla ingredienser i vår ingredientList!

Samtliga av våra 4 ingredienser måste in i listan som vi skapade i början. Enklast är att använda metoden **add** som finns i alla klasser som ärver utav List.

På rad 22 kan du klistra in:

```
ingredientList.add(cocaCola);
ingredientList.add(nutella);
ingredientList.add(mentos);
ingredientList.add(durex);
```

Tips, kika gärna på vilka fler metoder ArrayList har. Samtliga finns i alla typer av listor som ärver ifrån Javas List interface och är något vi som utvecklare jobbar med väldigt mycket! Exempelvis kan man se storleken genom ingredientList.size() eller se om den är tom med metoden ingredientList.isEmpty(). Samtliga metoder kommer alltså alla listor få så länge de tillhör den här typen utav ***List****.*

Det som händer här är endast att vi lägger till ingredienserna i listan. **Eftersom List-*an* som vi definierat den kan innehålla objekt av klassen Ingredient så** kan vi alltså lägga till dessa objekt. Hade vår *List* varit en *List<String>* hade detta inte gått. *(Men som tur var hade Milleaccendini tänkt på detta)*

5. Skapa en Getter metod för ingredientList

Just nu har vi vår lista, men vi har ingen metod för att hämta den ur klassen Cauldron. (Och således ur objekt som då är baserade på Cauldron klassen) Listan i sig finns där, men den måste läsas in i Main för att alla våra ingredienser ska kunna presenteras i vårt gränssnitt.

På rad 25 kan du klistra in följande metod som motsvarar en helt vanlig Getter:

```
public List<Ingredient> getIngredientList() {
    return ingredientList;
}
```

Du hade fått exakt samma resultat om du infogat en Getter via din IDE. Det enda du vill returnera här är själva Listan. Den här Getter metoden fungerar egentligen exakt likadant som den Getter metod som finns i Ingredient vid namn getIngredientName(), fast den returnerar en String och inte en Lista då förstås. Syftet är alltså för att hämta privata värden, i vårt fall till gränssnittet.

Toppen! Då har vi blandat ihop ingredienserna.

Milleaccendini är mycket nöjd med resultatet. Nu är det dags att se vårt resultat! Tryck på kör så som du normalt sätt kör ett program i Eclipse/Intellij och luta dig tillbaka!

I den här övningen har inte syftet varit att ni ska utmanas utan mer få en liten repetition. Det vi har gått igenom är alltså instansiering utav klasser, vi har satt värden genom Setter metoder, vi har även instansierat objekt genom konstruktorer med inparametrar, använt oss utav objektorientering för att nyttja andra objekts inbyggda funktioner och slutligen hämtat ett objekt genom en Getter metod.

Testa gärna själv, och uppgiften innehåller dessutom lite inspiration inför kommande JavaFX övningar som ni får ta del av. Happy coding!