IT UNIVERSITY OF COPENHAGEN

Grundlæggende Programmering E2020

Obligatorisk opgave D
Customer Tracker

Introduktion

Denne opgave skal laves og afleveres alene, men det er tilladt at snakke sammen med medstuderende om opgaven. BlueJ-projektet skal afleveres i en **.zip-fil** på LearnIT.

I denne opgave skal du implementere et system helt fra bunden. Du skal altså ikke tage udgangspunkt i et eksisterende projekt.

Opgave 1

En butiksleder har i de seneste uger indsamlet data for, hvor mange kunder, der besøger butikken i løbet af en dag. Butikslederen har nu brug for et program, der skal kunne bearbejde og vise det data, der er indsamlet.

1.1 For at komme i gang med projektet, skal du oprette en tekstbaseret "brugergrænseflade", du senere kan bruge til at teste de metoder, der behandler dataet. For at kunne oprette "brugergrænsefladen" uden at have implementeret de *faktiske* metoder endnu, skal du bruge et *interface*. Opret et interface kaldet `ICustomerTracker`.

1.2 `ICustomerTracker` skal indeholde følgende offentlige metoder:

- `int today()`
- `double avgThisWeek()`
- `double comparedToWeek(int week)`

1.3 Opret en ny klasse `TextView`. Klassen skal have et felt af typen `ICustomerTracker`. Derudover skal klassen have en konstruktør `TextView(ICustomerTracker cTracker)`, der sætter klassens felt til `cTracker`.

1.4 Klassen skal have tre metoder: `printToday()`, `printAvgThisWeek()` og `printComparedToWeek(int week)`. Altså én print-metode for hver metode i `ICustomerTracker`.

a) `printToday()` skal udskrive f.eks.: `"Today:  35 customers"`.

b) `printAvgThisWeek()` skal udskrive f.eks.: `"Average this week:  30.0 customers"`.

c) `printComparedToWeek(int week)` skal kalde interfacets metode `printComparedToWeek(int week)` og udskrive f.eks. `"Increase by 4 customers"`, `"Decrease by -4 customers"` eller `"No difference"`, afhængig af, om resultatet er positivt, negativt eller 0.

Opgave 2

Du skal nu lave en klasse, som implementerer interfacet `ICustomerTracker` og lave metoderne, der bruger det data, som butikken har indsamlet.

2.1 Opret en klasse kaldet `MockDB`. Kopiér koden fra dette link: https://pastebin.com/jnwpyYFE, og sæt den ind i `MockDB`. Klassen `MockDB` indeholder det data, som butikken har indsamlet. Dataet er indsat i et `HashMap`, der mapper fra et ugenummer (en `Integer`) til et array af antal kunder for hver dag i ugen (et `int[]`).

2.2 Opret en klasse kaldet `CustomerTracker`, der skal *implementere* interfacet `ICustomerTracker`. (Klassen kan ikke compile, før du har implementeret alle metoderne i `ICustomerTracker`.)

2.3 Klassen skal have to felter: `HashMap<Integer, int[]> customerData` samt `int currentWeek`. Konstruktøren `CustomerTracker(MockDB db, int currentWeek)` skal sætte feltet `customerData` til at være det `HashMap`, som `MockDB` indeholder. Derudover skal den instantiere `currentWeek`.

2.4 Implementér metoden `int today()`, som skal returnere det nyeste datapunkt i hashmappet `customerData`. Hvis man giver `currentWeek` som *key*, er det nyeste datapunkt det sidste tal i dét array, som returners.

2.5 Klassen skal have en *privat hjælpemetode* `double avgWeek(int week)`. Metoden skal udregne og returnere det gennemsnitlige antal kunder i den givne uge.

2.6 Implementér metoden `double avgThisWeek()`, som skal udregne og returnere det gennemsnitlige antal besøgende i den nuværende uge (`currentWeek`).

2.7 Implementér metoden `comparedToWeek(int week)`, som skal returnere forskellen mellem det gennemsnitlige antal besøgende i den nuværende uge og ugen, der bliver givet med som argument.

2.8 På BlueJs startside kan du nu oprette `MockDB`, `CustomerTracker` (hvor currentWeek er 47) og `TextView` objekter for at tjekke, at dine metoder i `TextView` fungerer som forventet.

- `printToday()` skal udskrive 35.
- `printAvgThisWeek()` skal udskrive 30.0.
- `printComparedToWeek(43)` skal udskrive $\sim$2.85.

Opgave 3

Programmet skal udvides med en testklasse, så du kan sikre dig, at metoderne i `CustomerTracker` virker som forventet.

3.1 Opret en klasse `CustomerTrackerTest`, som skal agere testklasse for `CustomerTracker`. Klassen skal have to felter: `MockDB db` samt `CustomerTracker cTracker`.

3.2 Lav en metode, `public void setUp()`, som initialiserer felterne `db` og `cTracker`. Metoden skal have annoteringen `@Before`, hvilket gør, at den automatisk bliver kaldt *før* hver af vores test-metoder bliver kaldt. Feltet `cTracker` initialiseres med argumenterne `db` og `47` (hvilket er den sidste uge i datasættet).

3.3 Lav en metode, `public void tearDown()`, som sætter felterne `db` og `cTracker` til `null`. Metoden skal have annoteringen `@After`, hvilket gør, at den automatisk bliver kaldt *efter* hver af vores test-metoder bliver kaldt.

3.4 Tilføj en testmetode, `public void today_returns35()`, som tester, at metoden `today()` i `CustomerTracker` returnerer 35. Dine test-metoder i klassen skal følge samme struktur som i eksemplet nedenfor:

```
@Test
public void add_given2Plus2_returns4() {
    //Arrange
    int expected = 4;

    //Act
    int actual = add(2,2);

    //Assert
    assertEquals(expected, actual);
}
```

Du kan tjekke, at dine tests virker, ved at kalde dem fra BlueJs startside.

3.5 Tilføj en testmetode, `avgThisWeek_returns30`, som tester, at metoden `avgThisWeek()` i `CustomerTracker` returnerer 30.0. Følg samme struktur som i den forrige test. Vær opmærksom på at `assertEquals()` tager tre argumenter, når man sammenligner *doubles*.

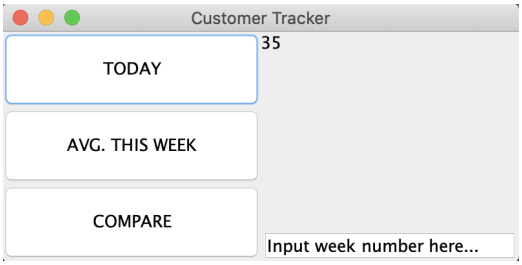
3.6 Lav to forskellige tests, som begge tester metoden `comparedToWeek()`. Den ene test skal teste, at $\sim$2.85 bliver returneret, når man sammenligner med uge 43. Den anden test skal teste, at 0 bliver returneret, når man sammenligner med den nuværende uge (47).

3.7 Tilføj en test, `public void comparedToWeek_given0_throwsAnException()`, som tester at en `NullPointerException` bliver kastet, hvis man kalder `comparedToWeek()` med argumentet 0. Vær opmærksom på, at når man tester for exceptions, kan vi ikke bruge samme struktur som ved de andre tests.

3.8 Tilføj en test, `public void comparedToWeek_given48_throwsAnException()`, som tester at en `NullPointerException` bliver kastet, hvis man kalder `comparedToWeek()` med argumentet 48.

Opgave 4 (Frivillig)

Systemet skal nu udvides med en brugergrænseflade *(Graphical User Interface (GUI))*, så en bruger kan interagere med programmet.

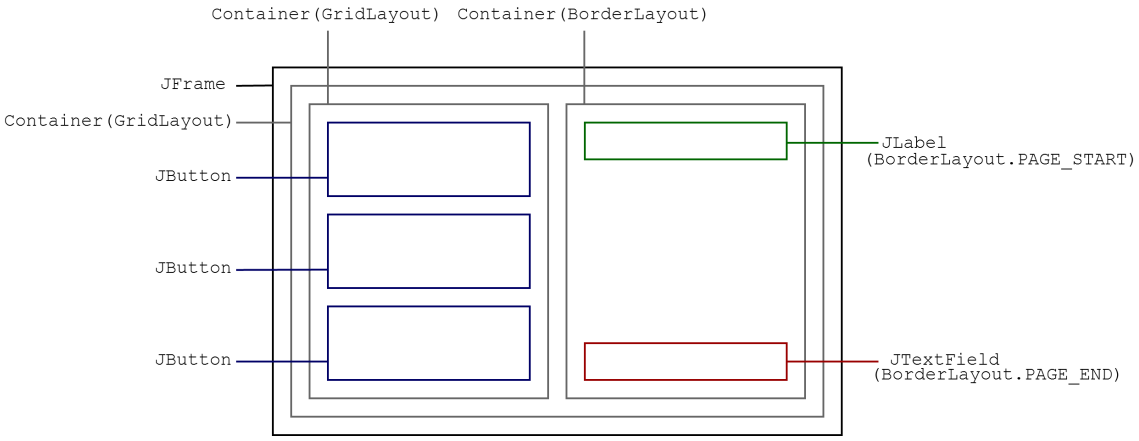


Figur 1: Eksempel på en brugergrænseflade.

4.1 Opret en klasse `GUIView`. Klassen har to felter: `MockDB db` samt `CustomerTracker cTracker`. Minimumskrav til brugergrænsefladen er:

- Den skal have tre knapper: én til hver `public` funktion i `CustomerTracker`.
- Den skal have et inputfelt, hvor man kan indtaste en uge til brug i `comparedToWeek(int week)`-metoden.
- Den skal have en måde at vise resultaterne fra metoderne.

Du kan lave brugergrænsefladen som du vil, så længe den opfylder kravene ovenfor og kan kalde de metoder, du har lavet i `CustomerTracker`. Figur 1 er et eksempel på en meget simpel brugergrænseflade, der opfylder kravene, som *kan* bruges til inspiration. Opbygningen af eksemplet er i Figur 2, hvor du kan se, hvilke *Java Swing elementer*, der er blevet brugt.



Figur 2: Teknisk opbygningen af brugergrænsefladen i Figur 1.