# Skriftlig eksamen for 1. semester datamatiker

# Forår 2018 Klasse l18dat1af

# **Prøvens varighed**: 4 timer Dato: 30.05.2018 kl. 8.30 til 12.30

# **Tilladte hjælpemidler:** Alle skriftlige hjælpemidler

# Det er ikke tilladt at kommunikere med andre under eksamen. Overtrædelse vil medføre bortvisning.

# IT afdelingen logger al trafik over skolens netværk, og de studerende skal aflevere deres mobiltelefoner ved prøvens start.

# **Materiale:** Du skal downloade et Netbeans projekt som zip fil fra Wiseflow, som din besvarelse skal tage udgangspunkt i (klasser og JUnit tests). JUnit tests er placeret i packages med samme navn som den tilhørende opgave. Eksempelvis ligger kode og unit tests til ”Opgave 1” begge i package med navnet opgave1.

# Eksamenssættet består af en række delopgaver. Vægtning af de enkelte opgaver fremgår af opgavebeskrivelserne.

# Procenttallene for opgaverne er vejledende.

# **Aflevering:** Du skal uploade din besvarelse som et Netbeans projekt eksporteret som .zip-fil på WiseFlow.

## Opgaverne

### Opgave 1 (24 %)

I hver delopgave skrives en metode i klassen StringHandler.

1. (4 %)

Implementér metoden public String stringManipulation(String str), der returnerer en streng, hvor forekomster af bogstavet ’c’ i input strengen er udskiftet med ’s’.

1. (4 %)

Implementér metoden public int stringLength(String str), der returnerer længden af inputstrengen. Varianter af streng input som null og den tomme streng ”” skal håndteres ved at returnere værdien ´0´.

1. (5 %)

Implementér metoden public int stringArrayLength(String[] arr), der returnerer den samlede længde af alle String objekter i input-arrayet.

F.eks. vil et array med de tre strenge ”hej” ”med” ”dig” have en samlet længde på 9.

1. (6 %)

Implementér metoden public String stringReverse(String str), der returnerer inputstrengen vendt om.

F.eks. bliver strengen ”hello” til ”olleh”.

1. (5 %)

Implementér metoden public int charCount(String str, char c), der returnerer antallet af forekomster af den givne char i den givne streng. Der skelnes ikke mellem store og små bogstaver.

F.eks. returneres tallet 3 for strengen ”[Ouagadougou](https://da.wikipedia.org/wiki/Ouagadougou)” og char ”o”.

### Opgave 2 (10 %)

Implementér en plat-eller-krone funktion i klassen FlipCoin. Metoden hedder flipCoin()og returnerer tilfældigt en af konstanterne HEAD eller TAIL i enum’en HeadTail.

### Opgave 3 (37 %)

Skriv metoderne i klassen Team. Metoderne benytter instanser af klassen TeamMember, som er implementeret.

Et objekt af typen TeamMember har et id, et navn og en rolle, f.eks. ”14”, ”Kurt”, ”Project Manager”

1. (4 %)

Implementér flg. metode i klassen Team, som tilføjer et teammedlem til en arrayliste i klassen:

public void addTeamMember(TeamMember m)

1. (4 %)

Implementér en metode til fjerne medlemmer fra arraylisten. Returnér true, hvis et medlem bliver fjernet eller false, hvis medlemmet ikke findes på listen.

public boolean removeTeamMember(TeamMember m)

1. (10 %)

Implementér en metode i klassen Team, som finder det næste ledige MemberID.

public int getNextId()

ID’er starter på 0 og bliver talt op for hvert medlem der tilføjes til listen, men det næste ID er altid det mindste ledige tal.

F.eks.

*Anders får ID 0, Bertram får ID 1 og Christian får ID 2, da de melder sig ind i den rækkefølge. Bertram stopper og ID 1 bliver ledigt, så når Dennis efterfølgende melder sig ind, får han   
derfor ID 1.*

Du kan gå ud fra at listen over medlemmer er sorteret efter ID.

1. (7 %)

Implementér flg. metode i klassen Team, som ud fra en given role som inputparameter returnerer en arrayliste af TeamMember objekter, som har denne rolle:

public ArrayList<TeamMember> retriveMembersByRole(String role)

1. (6 %)

Implementér en metode i klassen Team klassen, der sorterer listen af teammedlemmer efter navn i stigende orden.

public void sortByNameAsc(ArrayList<TeamMember> members)

1. (6 %)

Implementér en metode i klassen Team, der sorterer listen af teammedlemmer efter rolle i faldende orden.

public void sortByRoleDesc(ArrayList<TeamMember> members)

Det skal stadigt være muligt at sortere efter navn i stigende orden (jf. opgave d).

### Opgave 4 (12 %)

I denne opgave skal du lave klassen Order færdig, samt udfylde metoder i OrderFactory klassen.

1. (6 %)

Skriv en konstruktør til Order klassen.

Et Order objekt skal indeholde flg. attributter:

* id til unik identifikation af hver ordre.
* itemDescription til angivelse af varenavn, f.eks. ”t-shirt”
* created til dato for oprettelse af en ordre
* closed til dato for afslutning af en ordre. Den sættes i første omgang til null på et nyoprettet Order objekt (dvs. den er ikke med som parameter i konstruktørmetoden), men skal senere kunne ændres.

Skriv ligeledes metoden makeOrder i OrderFactory, som returnerer et Order objekt. Metoden modtager ovennævnte parametre undtagen closed.

1. (6 %)

Udvid metoden makeOrder med et tjek på at input parametre hverken er null eller tomme. Hvis dette er tilfældet kastes en IllegalArgumentException.

### Opgave 5 (17 %)

Færdiggør klassen Inventory.

1. (8%)

Udregn lagerbeholdningen. Metoden calculateInventoryValue får som input en tekst streng, der fortæller hvor mange af en given vare der er på lager og hvilken pris de står til:

En inputstreng kunne f.eks. se således ud:

6 Paraplyer til kr. 45,00

47 Sokker til kr. 13,50

12 Bowlingkugler til kr. 365,00

Metoden skal returnere den samlede pris for alle varerne.

1. (9%)

Implementér metoden addToStock, som tilføjer varer til lageret.

Metoden får følgende inputparametre:

* HashMap<String, Integer> stock, som er det nuværende lager. Map’et har varenavnet (itemName) som nøgle og antal af enheder (amount) på lager som værdi.
* String itemName, som er navnet på en vare
* int amount, som er antallet af vare der skal tilføjes til lageret.

Metoden skal indsætte varen på lageret med det givne antal, medmindre varen allerede er på lager. Hvis det er tilfældet, skal antallet i stedet opdateres ved at det nye antal lægges til det allerede eksisterende antal.