1. Beregning af areal

I denne opgave skal du skrive et program, som beregner arealet af forskellige former. Lav så meget af opgaven, du kan nå.

1. Skriv et interface `Shape` med metoden: `public double calculateArea()`

2. Skriv to klasser, som implementerer interfacet.

Klassen `Square` skal have følgende attributter (som skal sættes via klassens constructor) :

- `height`
- `length`

Klassen skal implementere metoden `calculateArea()`. Arealberegningen foretages ved at gange højde med længde.

Klassen `Circle` skal have følgende attribut (som skal sættes via klassens constructor):

- `radius`

Klassen skal implementere metoden `calculateArea()`. Arealet af en cirkel er pi * radius^2 (hvilket kan skrives som Math.PI * (radius* radius)).

3. Skriv en klasse `Main` med en `main`-metode.

Opret et `Square`-objekt og et `Circle`-objekt i main-metoden og test at deres `calculateArea()`-metoder virker.

<u>Hvis du har tid:</u>

Skriv en klasse `Shapes`, som har en samling af `Shape`-objekter.

Skriv en metode `addShape(Shape s)` i klassen, som kan bruges til at tilføje `Shape`-objekter til samlingen.

Skriv en metode `public double totalArea()` som beregner det samlede areal af alle `Shape`-objekterne.

Test at metoden virker.

2. Tjek passwords

I denne opgave skal du skrive et program, som kan undersøge om et password opfylder et antal regler. Lav så meget af opgaven, som du kan nå.

1. Skriv en klasse `PasswordValidater` med attributterne

- `minLength`
- `maxLength`
- `numbersRequired`

Attributterne skal sættes i klassens constructor.

2. Skriv metoden `public boolean isValid(String password)` i klassen. Metoden skal undersøge om `password` opfylder følgende regler:

a. `password` må ikke være kortere end `minLength` eller længere end `maxLength`.
b. `password` skal indeholde minimum ét tal, hvis `numbersRequired` er true. Du kan bruge metoden `Character.isDigit()` til dette.

Hvis `password` opfylder begge krav, skal metoden returnere true og ellers skal den returnere false.

3. Skriv en klasse `Main` med en `main`-metode, hvor du tester, at din metode virker.

<u>Hvis du har tid:</u>

Udvid reglerne, så det er påkrævet, at passwordet indeholder både store og små bogstaver (`Character.isUpperCase()`).

3. Sammenlign byer

I denne opgave skal du skrive et program, der kan sammenligne byers størrelse. Lav så meget af opgaven, som du kan nå.

1. Skriv en klasse `City` med følgende attributter, der skal sættes via klassens constructor:

- `name`
- `population`

Skriv `getter`-metoder for attributterne.

2. Skriv en klasse `Cities`, der indeholder en samling af byer.

Skriv en metode `public void add(City city)` som kan bruges til at tilføje en by til samlingen. Skriv en metode `public City getBiggestCity()` som kan returnere den by, der har flest indbyggere.

3. Skriv en klasse `Main` med en `main`-metode.

I `main`-metoden skal du oprette et objekt af klassen `Cities`.

Tilføj dernæst fem `City`-objekter til `Cities`-objektet ved at bruge `addCity(City city)`-metoden. Dine fem objekter skal have følgende attributter:

- ”Kongens Lyngby”, 57500
- ”Ribe”, 8257
- ”Storvorde, 3425
- ”Tarm”, 4010
- ”Horreby”, 305

Kald metoden `getBiggestCity()` på `Cities`-objektet og print navnet ud på den by, der returneres.

<u>Hvis du har tid:</u>

Skriv en metode `public int getTotalPopulation()` i klassen `Cities`. Metoden skal returnere det samlede indbyggertal af byerne i samlingen i `Cities`.

Skriv en metode `public int getAveragePopulation()` i klassen `Cities`. Metoden skal returnere det gennemsnitlige indbyggertal af byerne i samlingen i `Cities`.

4. Slå med terninger

I denne opgave skal du skrive et program, der kan slå med terninger. Lav så meget af opgaven, som du kan nå.

1. Skriv en klasse `Die`, som repræsenterer en terning. Klassen skal have attributten

- `sides`

Attributten skal sættes via klassens constructor.

2. Klassen Die skal have metoden `public int roll()` som returnerer et tilfældigt tal mellem 1 og antal sider på terningen (begge inklusive). Du kan fx bruge metoden `nextInt()` fra klassen `Random` til at få genereret et tilfældigt tal.

3. Skriv en klasse `Cup`, som kan kaste med terninger. Klassen skal have én eller to attributter, der holder to `Die`-objekter. Attributterne skal sættes via klassens constructor.

4. Klassen `Cup` skal have metoden `public int howManyIdenticalThrows(int times)` som slår med terningerne et antal gange (times) og returnerer hvor mange gange der er slået to ens (fx to 1'er, to 2'ere osv).

5. Skriv en klasse `Main` med en `main`-metode, hvor du tester, at din kode virker.

Hvis du har tid:

Lav en metode `public int[] results(int times)` i klassen `Cup`. Metoden skal slå med terningerne et antal gange (`times`) og returnere et array som indeholder resultatet af hvert slag (det vil sige summen af de to terninger).

5. Dialog med bruger

I denne opgave skal du skrive en metode, der kan fortælle en bruger hvor længe der er til vedkommende kan gå på pension. Lav så meget af opgaven, som du kan nå.

1. Skriv en klasse `Dialog`. Klassen skal have attributten

    - `currentYear`

Attributten repræsenterer det år, vi er i og den sættes via klassens constructor.

2. Klassen `Dialog` skal have metoden `public void getYearOfBirth()`. I metoden skal der ske følgende

1. Metoden skriver til brugeren (`System.out.println()`) og beder brugeren indtaste sit fødselsår.
2. Metoden kalder en ny metode `public int getYearsToRetirement(int i)` med fødselsåret som argument (denne metode skal du lave om lidt).
3. Metoden skriver resultatet fra `getYearsToRetirement (int i)` til brugeren. Du kan fx skrive noget lignende "Du har nu 27 år til pensionering" til din bruger.

3. Metoden `getYearsToRetirement(int i)` skal gøre følgende

1. Lægge 67 til fødselsåret, som er givet som parameter til metoden (`int i`). Dette giver pensionsåret.
2. Trække `currentYear` fra pensionsåret for at finde ud af hvor længe der er til pension for brugeren.
3. Returner resultatet fra metoden.

<u>Hvis du har tid:</u>

Lav nogle tjek på det input du får fra brugeren. Input skal være et tal med fire cifre, det må ikke være negativt og det må ikke være højere end `currentYear`. Skriv til brugeren, hvis input er ugyldigt.

6. Shopping

I denne opgave skal du implementere en indkøbskurv-klasse. Lav så meget af opgaven, som du kan nå.

1. Skriv en klasse, `ShoppingCart`, som skal have følgende metoder og attributter:

- `itemList`: Attributten skal være en liste over elementer i indkøbskurven.
- `addItem(Item)`: Metoden skal tilføje et element til indkøbskurven. Klassen `Item` skal du lave om lidt.
- `removeItem(Item)`: Metoden skal fjerne et element fra indkøbskurven, hvis det findes i indkøbskurven.
- `getTotalPrice()`: Metoden skal returnere den samlede pris for alle elementer i indkøbskurven.

2. Skriv en klasse `Item`, som skal repræsentere hver enkelt vare i indkøbskurven. `Item`-klassen skal have følgende metoder og attributter:

- `getName()`: Metoden skal returnere navnet på elementet.
- `getPrice()`: Metoden skal returnere prisen for elementet.
- `name`: Attributten skal indeholde navnet på elementet.
- `price`: Attributten skal indeholde prisen for elementet.

Eksempel på brug af indkøbskurv-klassen:

```
Item apple = new Item("Æble", 10);
```

3. Skriv en klasse `Main` med en `main`-metode, hvor du tester at din kode virker. Opret et antal Item-objekter, tilføj dem til et `ShoppingCart`-objekt og kald metoden `getTotalPrice()`. Prøv også at fjerne nogle `Item`-objekter fra `ShoppingCart` igen og kald `getTotalPrice()` igen.

<u>Hvis du har mere tid:</u>

- Implementer metoden `getItems()`: Metoden skal returnere en liste over alle elementer i indkøbskurven.
- Lav en attribut `discount` i klassen `ShoppingCart`. Lav en setter-metode til attributten. Lad `getTotalPrice()` medtage rabat, hvis `discount` er over 0.

7. Eksamensbevis

I denne opgave skal du skrive en klasse, der repræsenterer et eksamensbevis, som kan skrives pænt ud. Lav så meget af opgaven, som du kan nå.

1. Skriv en klasse `Course` med attributterne
    - `courseName`
    - `grade`

Attributterne skal sættes via klassens constructor. Lav en `public String toString()`-metode til klassen, som returnerer en `String`, der indeholder `courseName` og `grade` for eksempel i formatet "kursusnavn: karakter".

2. Skriv en klasse `Diploma` med attributterne
    - `studentName`
    - `courses`

hvor `courses` er en samling af `Course`-objekter.

Attributten `studentName` skal sættes via klassens constructor.

3. Skriv metoden `public void addCourse(Course course)` i klassen Diploma. Metoden skal bruges til at tilføje nye `Course`-objekter til samlingen `courses`.
4. Skriv metoden `public String toString()` i klassen Diploma. Metoden skal returnere en `String` med `studentName` efterfulgt af `courseName` og `grade` fra hvert enkelt `Course` i samlingen `courses`.
5. Skriv en klasse `Main` med en `main`-metode. Opret objekt af klassen `Diploma`. Tilføj et antal `Course`-objekter til `Diploma`-objektets samling. Udskriv `Diploma`-objektet ved at anvende `toString()`-metoden.

<u>Hvis du har tid:</u>

- Lav et tjek i klassen `Course` så `grade` ikke kan få andre værdier end –3, 00, 02, 4, 7, 10 eller 12.
- Lav en metode i `Diploma` til at udregne gennemsnit af karaktererne fra `courses`.

8. Zoologisk Have

I denne opgave skal du skrive et program, der repræsenterer en samling af dyr, som alle skal kunne sige noget. Lav så meget af opgaven, som du kan nå.

1. Skriv en abstrakt klasse `Animal` med en abstrakt metode `public void makeSound()`.
2. Skrive en klasse `Lion` som extender (nedarver fra) `Animal`. Implementer metoden `makeSound()` så metoden printer en "løvelyd" (fx "Rooooar").
3. Skriv en klasse `Parrot` som extender (nedarver fra) `Animal`. Implementer metoden `makeSound()` så metoden printer en "papegøjelyd" (fx "Hello, world").
4. Skriv en klasse `Main` med en `main`–metode. I `main`-metoden skal du lave en `ArrayList<Animal>` og tilføje et `Lion`-objekt og et `Parrot`-objekt til listen.

For hvert element i listen skal du nu kalde metoden `makeSound()` så objektets "lyd" printes.

<u>Hvis du har tid:</u>

Udvid klassen `Animal` med attributten `isMammal`. Lad attributten blive sat i klassens constructor. Skriv constructors i de to underklasser `Lion` og `Parrot`, som kalder superklassens constructor.

Skriv en metode i klassen `Main`, der kan tage en liste af `Animal`-objekter og returnere en ny liste, som kun indeholder de `Animal`-objekter, som er pattedyr (mammals).

Skriv en metode i klassen `Main`, der kan tage en liste af `Animal`-objekter og returnere en ny liste, som kun indeholder de `Animal`-objekter, som er objekter af klassen `Lion`.

9. Sten, saks, papir

Skriv et program, der simulerer spillet “sten, saks, papi”r. Programmet skal give brugeren mulighed for at spille mod computeren. Lav så meget af opgaven, som du kan nå.

Spillet skal bestå af følgende trin:

1. Programmet skal bede brugeren om deres navn.
2. Programmet skal derefter bede brugeren om at vælge deres træk ved at indtaste enten "sten," "saks," eller "papir."
3. Programmet skal derefter tilfældigt vælge et træk for computeren (hint: du kan bruge klassen `Random` fra Javas API til at genere et tilfældigt tal).
4. Programmet skal derefter afgøre vinderen af runden baseret på følgende regler:
    a. sten slår saks
    b. saks slår papir
    c. papir slår sten
5. Programmet skal derefter udskrive resultatet af runden, inklusive brugerens træk, computerens træk, og vinderen.
6. Programmet skal derefter spørge brugeren, om de vil spille igen. Hvis brugeren indtaster "ja," skal spillet starte forfra ved trin 2. Hvis brugeren indtaster "nej," skal programmet slutte.

<u>Hvis du har tid:</u>

Giv mulighed for, at to (menneskelige) brugere spiller mod hinanden.

10. Lommeregner

Skriv et program, der repræsenterer en simpel lommeregner. Lav så meget af opgaven, som du kan nå.

Skriv en klasse kaldet `Calculator` med følgende funktioner og attributter:

- Attributten `result`, som starter med at være 0.
- En getter-metode til `result`.
- Metoden `public void add(double x)`, der lægger tallet x til `result`.
- Metoden `public void subtract(double x)`, der trækker tallet x fra `result`.
- Metoden `public void clear()`, som nulstiller `result`.
- Metoden `public void add(double[] numbers)`, der tager et array af doubles og lægger alle tallene til `result`.

Når du har implementeret klassen, skal du oprette et objekt af klassen i en `main`-metode og teste, om alle funktionerne virker som forventet. Du kan også gøre dette løbende hver gang du har implementeret en metode.

Hvis du har mere tid:

Kig i Javas API og se hvilke metoder `Math`-klassen tilbyder. Brug en eller flere af disse metoder til at udvide funktionaliteten af din lommeregner.