2022

Oppgave 1

a. Lag kode for klassen Elev. Klassen Elev har attributtene navn, alder og klassetrinn.

b. En av elevene på skolen heter Rune. Han er 16 år og går i 5. klasse. En annen heter Birger.
Han er 15 år og går også i 5. klasse. Lag kode som oppretter to objekter som instanser av
klassen Elev. Lag kode som skriver ut informasjon om elevene.

c. Lag kode for klassen Lærer som arver egenskaper fra klassen Ansatt. Klassen Ansatt har
attributtene rolle, avdeling, månedslønn. Klassen Lærer har attributtene navn og alder.

d. En av lærerne på skolen heter Lars. Han er 28 år. Lag kode som oppretter et objekt som en
instans av klassen Lærer. Alle lærere har rolle underviser, tilhører avdelingen fag og har en
månedslønn på 25000 kr. Lag kode som skriver ut informasjon om Lars.

e. Lag kode for klassen Fag. Klassen Fag har attributtene fagnavn, elever og lærere. Et fag kan ha
flere elever og lærere.

f. Et av fagene er samfunnsfag. Faget har kun to elever Rune og Birger og undervises av Lars.
Lag kode som oppretter et objekt som en instans av klassen Fag.

```
// 2022 eksamen

// Oppgave 1

class Elev {

    // 1a
```

```
    navn: string;
    alder: number;
    trinn: number | null;
    constructor(navn: string, alder: number, trinn: number | null){
        this.navn = navn;
        this.alder = alder;
        this.trinn = trinn;
    }
}

// 1b
const rune = new Elev("Rune", 16, 5);
const birger = new Elev("Birger", 15, 5);

console.log(Elev);

// 1c
class Ansatt {
    rolle: string;
    avdeling: string;
    mndLonn: number;
    constructor(rolle: string, avdeling: string, mndLonn: number){
        this.rolle = rolle;
        this.avdeling = avdeling;
        this.mndLonn = mndLonn;
    }
}

class Laerer extends Ansatt {
    navn: string;
    alder: number;

    // static ROLLE: string = "Underviser";
    // static AVD: string = "Fag";
    // static LONN: number = 25000;

    constructor(navn: string, alder: number, rolle: string,
                avdeling: string, mndLonn: number){
        // super(rolle, avdeling, mndLonn);

        // if(!rolle && !avdeling && !mndLonn){
        //     this.rolle = Laerer.ROLLE;
        //     this.avdeling = Laerer.AVD;
        //     this.mndLonn = Laerer.LONN;
        // }

        super("underviser", "fag", 25000);

        this.navn = navn;
        this.alder = alder;
    }
}

const lars = new Laerer("Lars", 28, this.rolle, this.avdeling, this.mndLonn);
// const lars = new Laerer("Lars", 28);

console.log(lars.toString());
```

```
class Fag {
    fagNavn: string;
    elever: Elev[] = [];
    ansatte: Ansatt[] = [];

    constructor(fagNavn: string, elever: [], ansatte: []){
        this.fagNavn = fagNavn;
        this.elever = elever;
        this.ansatte = ansatte;
    }

}

const samfunnsfag = new Fag("samfunnsfag", this.elever, this.ansatte);
console.log(samfunnsfag);
```

Oppgave 2

I denne oppgaven skal du lage noe av koden til et yatzee spill.
Koden skal simulere terningkast med 5 terninger. Terningene kan som kjent gi tilfeldige verdier fra
en til seks.
Det skal gjennomføres tre terningkast der brukeren har mulighet til å ta vare på enere, toere,
treere, firere, femmere eller seksere i hvert kast ved å trykke på knapper på en webside. Dvs at
det kun er terningene med annen verdi som endrer seg i neste kast.
Verdiene i hvert kast skal vises på en webside som vedlagt eksempel.
Spillet er ferdig etter tre kast. Da må brukeren ta endelig stilling til hvilke terninger som beholdes
og det beregnes en score som tilsvarer antall like ganget med verdien til terningen.
Til slutt fjernes muligheten til å trykke på knappene.

ALT 1 - meh

```
//Oppgave 2 på nytt:


class Terning {
    antKast = 0;
    disabled = false;

    verdi = Math.floor(Math.random() * 6 + 1);
```

```
    kast(){
        spareTerninger = [];
        antKast++;

        switch(this.antKast) {
            case 1:
                for (i=0; i < 5; i++) {
                    this.kast1Terninger.push(this.verdi);
                }
                if(spareTerninger.length === 5){
                    this.avslutt();
                }
                break;

            case 2:
                for(i=0; i < 5 - spareTerninger.length; i++){
                    this.kast2Terninger.push(this.verdi);
                }
                if(spareTerninger.length === 5){
                    this.avslutt();
                }
                break;

            case 3:
                for(i=0; i < 5 - spareTerninger.length; i++){
                    this.kast3Terninger.push(this.verdi);
                }
                if(spareTerninger.length === 5 || this.antKast === 3){
                    this.avslutt();
                }
                break;
        }

        this.kast();
    }

    spar(){
        this.spareTerninger.push(utfall);

    }

    avslutt(){
        let sum = 0;

        for(let number of this.spareTerninger) sum += number;

        return sum;
    }

    render(){
        muligUtfall = [1,2,3,4,5,6];
        kast1Terninger = [];
        kast2Terninger = [];
        kast3Terninger = [];

        return(
            <div>
                <button onclick={this.kast()}>Kast Terning</button>
```

```
                <div id="knapperDiv">
                   { this.muligUtfall.map((utfall) => {
                        <button id="button"
                            value={utfall}
                            onclick={() => {
                                this.spar(utfall);
                                this.kast();
                            }}
                        >
                        {utfall}
                        </button>
                        })
                    }
                </div>

                <card>

                </card>

                <card id="kastDiv">
                    <div>Sparte terninger: { sparteTerninger.map((terning) => {
                        <div>{terning.verdi}</div>
                    }) }</div>
                </card>

                <button id="finishBtn" onclick={this.avslutt()}>Reset</button>

            </div>
        )
    }

}
```

ALT 2 - løst med funksjoner

```
// Creating the Dice class
 let valueArr = [1, 2, 3, 4, 5, 6];

 class Dice {
     constructor() {
         this.side = 0;
     }

     roll(){
         this.side = valueArr[Math.floor(Math.random() * valueArr.length)];
     }
 }

 // Creating buttons
 let result = document.createElement('p');
 result.id = 'result';
```

```
 document.body.appendChild(result);

 let diceArr = [];

 // start button & dice
 let startButton = document.createElement('button');
 startButton.innerText = 'Roll dice';
 startButton.id = 'start';
 startButton.onclick = () => {
     result.innerText += `First roll: \n`;
     for(i=0; i<5; i++){
         let dice = new Dice();
         dice.roll();
         diceArr.push(dice);
         result.innerText += ` ${dice.side}`;
     }
     keepDiceInfo.style.visibility = 'visible';
     buttonBox.style.visibility = 'visible';
     startButton.style.visibility = 'hidden';
 }

// Creating info boxes & buttons
let keepDiceInfo = document.createElement('p')
keepDiceInfo.id = 'diceInfo';
keepDiceInfo.style.visibility = 'hidden';
keepDiceInfo.innerText = 'Which dice do you keep?'
document.body.appendChild(keepDiceInfo)
let buttonBox = document.createElement('div');
buttonBox.id = 'buttonBox';
buttonBox.style.visibility = 'hidden';
document.body.appendChild(buttonBox);

valueArr.forEach(value => {
 let button = document.createElement('button');
 button.innerText = value;
 button.onclick = () => reRoll(value);
 document.getElementById('buttonBox').appendChild(button);
 document.body.appendChild(startButton);
});

// defining count to keep track of rerolls
let count = 1;

// reroll function
function reRoll(x){
 if(count === 3){
     final(x);
     return;
 }
 count +=1;
 result.innerText += `\n Roll ${count}:`;
 diceArr.forEach(dice => {
     if(dice.side !== x){
         // set this index to be new dice
         let die = new Dice();
         die.roll();
         diceArr[diceArr.indexOf(dice)] = die;
     }
```

```
 });
 show(diceArr);
}

function show(x){
result.innerText += `\n`;
x.map(dice => result.innerText += ` ${dice.side} `)
}

// reset button
let resetBtn = document.createElement('button');
resetBtn.innerText = 'Reset';
resetBtn.id = 'reset';
resetBtn.onclick = () => {
diceArr = [];
count = 1;
document.getElementById('result').innerText = '';
document.getElementById('reset').style.visibility = 'hidden';
document.getElementById('start').style.visibility = 'visible';
}
resetBtn.style.visibility = 'hidden';
document.body.appendChild(resetBtn);

function final(x){
result.innerText += `\n Score: \n`;
let score = diceArr.reduce((a, b) => {
 if(b.side === x){
     return a + x;
 }
 return a;
}, 0);
result.innerText += ` ${score}`;
document.getElementById('reset').style.visibility = 'visible';
document.getElementById('buttonBox').style.visibility = 'hidden';
document.getElementById('diceInfo').style.visibility = 'hidden';
}
```

Oppgave 3

Du skal lage en liten nettbutikk som skal bestå av en side der brukeren kan legge til varer i en
handlekurv og en side som skal vise handlekurven og hva handleturen kommer til å koste.

```
CREATE TABLE Items (
  id INT NOT NULL AUTO_INCREMENT,
  name TEXT NOT NULL,
  description TEXT,
  price INT NOT NULL,
  count INT NOT NULL,
  PRIMARY KEY(id)
);
```

```
INSERT INTO Items (name, description, price, count) VALUES ('Eple',
'Norske epler', 10, 200);
INSERT INTO Items (name, description, price, count) VALUES
('Appelsin', 'Fra Spania', 20, 100);
INSERT INTO Items (name, description, price, count) VALUES ('Melk',
'Tine Lettmelk', 25, 50);
INSERT INTO Items (name, description, price, count) VALUES ('Brød',
'Dansk rugbrød', 30, 20);

CREATE TABLE Orders (
  id INT NOT NULL AUTO_INCREMENT,
  itemId INT NOT NULL,
  itemCount INT NOT NULL,
  PRIMARY KEY(id)
);
```

Du kan gå ut i fra at databaseoppkoblingen allerede er satt opp. I tillegg kan du programmere alt i
en fil, for eksempel index.js eller index.tsx.
Applikasjonen skal ha følgende funksjonaliteter:

I varesiden skal en kunne legge til varer i handlekurven (bruk da Orders tabellen), og følgende informasjon skal vises:

- varenavn
- varebeskrivelse
- antall varer lagt til i handlekurven
- antall varer tilgjengelig (for eksempel 200 epler)

I handlekurvsiden skal følgende informasjon vises om varene som er lagt til i handlekurven:

- varenavn
- varepris
- antall for hver vare
- totalpris for hver vare
- totalpris for alle varene

Index

```
import * as React from 'react';
import { createRoot } from 'react-dom/client';
import { Component } from 'react-simplified';
import { NavLink, HashRouter, Route } from 'react-router-dom';
import { Items, Orders, shopService } from './services';
import { Alert, Card, Row, Column, Button, Form, NavBar } from './widgets';
import { createHashHistory } from 'history';

class Menu extends Component {
  render() {
      return (
          <div>
              <NavBar brand="Shop">
                  <NavLink exact to ="/" activeStyle={{ color: 'darkblue' }}>
                    Items
                  </NavLink>{ ' ' }
                  <NavLink exact to ="/shoppingCart" activeStyle={{ color: 'darkblue' }}>
                    Shopping Cart
                  </NavLink>{ ' ' }
              </NavBar>
          </div>
      )
  }
}


class ShopItems extends Component {
  items: Items[] = [];

  render() {
    return(
       <Card title="Items">
        <Row>
          <Column width={4}>
            <h4>Items</h4>
          </Column>
          <Column width={4}>
            <h5>Added / Available</h5>
          </Column>
        </Row>
        {
          this.items.map((item, index) => {
            return <Row key={index}>
                      <Column width={4}>
                        <b>
                          { item.name }
                        </b>
                        <p>
                          { item.description }
                        </p>
                      </Column>
                      <Column width={4}>
                      { item.itemCount ? item.itemCount : 0 }/{ item.count }
                      </Column>
                      <Column width={4}>
                        <Button.Light
                          onClick={() => this.addToCart(item)}
```

```
                        >
                          Add to cart
                        </Button.Light>
                      </Column>
                    </Row>
          })
        }
      </Card>
    );
  }


  addToCart(item: Items){
    if(item.itemCount && item.itemCount >= item.count) return console.log("ikkeno mer");

    shopService.addItemToCart(item.id);
    shopService.updateCart(item.itemCount+1, item.id);

    console.log("added one " + item.name + " to cart");
    shopService.getItems((items) => {
      this.items = items;
    });
  }

  mounted() {
    shopService.getItems((items) => {
      this.items = items;
    });
  }
}

class ShoppingCart extends Component {
  items: Items[] = [];

  render() {
    return(
      <Card>
        <Row>
          <Column width={2}>
            <h4>Name</h4>
          </Column>
          <Column width={2}>
            <h4>Price per item</h4>
          </Column>
          <Column width={2}>
            <h4>Count</h4>
          </Column>
          <Column width={2}>
            <h4>Total</h4>
          </Column>
        </Row>
        { this.items.map((item, index) => {
          return <Row key={index}>
            <Column width={2}>
              { item.name }
            </Column>
            <Column width={2}>
              { item.price }
```

```
            </Column>
            <Column width={2}>
              { item.itemCount }
            </Column>
            <Column width={2}>
              { item.itemCount * item.price }
            </Column>
          </Row>
        }) }

        <Row>
          <Column width={6}>
            <h4>Sum</h4>
          </Column>
          <Column>
            <h4>{ this.calculateSum() }</h4>
          </Column>
        </Row>
      </Card>
    )
  }

  mounted() {
    // shopService.getOrders((orders) => {
    //   this.orders = orders;
    // });
    shopService.getItems((items) => {
      this.items = items;
    });
  }

  calculateSum(): number{
    let sum = 0;

    if(this.items.length){
      for (let i = 0; i < this.items.length; i++) {
        sum += this.items[i].itemCount * this.items[i].price;
      }
      return sum;
    }
    else return 0;
  }

}


let root = document.getElementById('root');
if (root)
    createRoot(root).render(
        <div>
            <Alert />
            <HashRouter>
              <Menu />
                <div>
                    <Route exact path="/" component={ShopItems} />
                    <Route exact path="/shoppingCart" component={ShoppingCart} />
                </div>
            </HashRouter>
```

```
        </div>
    )
```

Services

```
import { pool } from './mysql-pool';
import type { RowDataPacket, ResultSetHeader } from 'mysql2';


export class Items {
  id!: number;
  name: string | undefined;
  description: string | null | undefined;
  price!: number;
  count!: number;
  itemCount: any;
}

export class Orders {
  id!: number;
  itemId: number | undefined;
  itemCount: number | undefined;
}

class ShopService {
  getItems(success: (items: Items[]) => void){
    pool.query(
'SELECT DISTINCT Items.*, Orders.itemCount FROM Items'
'LEFT JOIN Orders ON Items.id = Orders.itemId',
      (error: Error, results: RowDataPacket[]) => {
        if(error) return console.error(error);

        success(results as Items[]);
      }
    )
  }

  // getOrders(success: (orders: Orders[]) => void){
  //   pool.query(
  //     'SELECT DISTINCT Items.name, Items.price, Orders.*
  //FROM Orders INNER JOIN Items ON Orders.itemId = Items.id',
  //     (error: Error, results: RowDataPacket[]) => {
  //       if(error) return console.error(error);

  //       success(results as Orders[]);
  //     }
  //   )
  // }

  addItemToCart(id: number){
    pool.query(
      'INSERT INTO Orders(itemId, itemCount) VALUES (?,1)',
```

```
      [id], (error: any, results: RowDataPacket[]) => {
        if(error) return console.error(error);

        return (results);
      }
    )
  }

  updateCart(itemCount: number, id: number){
    pool.query(
      'UPDATE Orders SET itemCount = ? WHERE itemId = ?',
      [itemCount, id], (error: any, results: RowDataPacket[]) => {
        if(error) return console.error(error);

        return(results);
      }
    )
  }
}

export let shopService = new ShopService();
```

OUTPUT