

PandaMath



INNHALDSFORTEGNELSE

Intro	2
Navigasjonsmuligheter	3
Hjem	3
Spill	3
Preferanser	4
Statistikk	4
Design - fargevalg og ikoner	4
Logikk	6
Splash	7
Hjem	7
Spill	8
Preferanser	11
Statistikk	12

Intro

PandaMath er en matte applikasjon for å lære og utvikle seg i addisjonsstykker. PandaMath skal både være gøy, men samtidig lærerik. Målgruppen er alle barn under 12 år, med tanke på design, struktur og vanskelighetsgrad. Barn og voksne utenfor målgruppen vil også ha det gøy med PandaMath.

PandaMath inneholder 25 unike spørsmål. Bruker kan velge om han vil spille med 5, 10 eller 25 spørsmål før han starter et spill, hvor 5 er standard. Alle spørsmål vil bli gitt i tilfeldig rekkefølge og brukeren vil ikke ha muligheten til å få samme spørsmål, flere ganger iløpet av et spill.


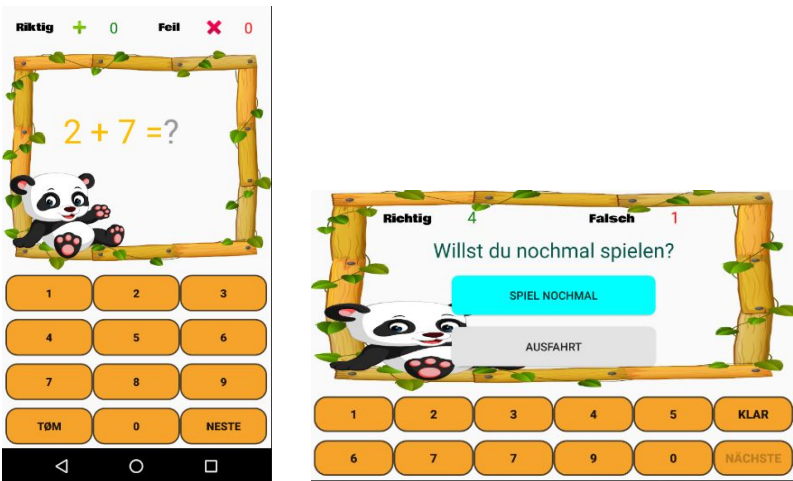
Når spillet er startet, kan brukeren fortsette å spille flere runder. Dersom brukeren velger 5 spørsmål, kan han spille 5 runder, før han går tom for spørsmål. Dersom brukeren velger 10 spørsmål, så kan han spille 2 runder, og dersom brukeren velger 25, så vil han bare kunne spille en runde. Om brukeren vil fortsette å spille selv om det ikke er flere spørsmål, kan han trykke på Avslutt og starte et nytt spill fra hjem-skjermen.

Antall feil og riktige svar vil både vises i runden, og legges til i statistikken på slutten av rundene. Der er det mulig å se på total poeng og nullstille statistikken.

PandaMath er tilgjengelig både på Norsk og Tysk, og dette kan velges av bruker under preferanser. Norsk er standard. Språk konfigurasjonen er alltid oppdatert.

Applikasjonen tar hensyn til lagring av data og tilstandsbevaring. Det er mulig å gå inn og ut av applikasjonen, snu mobilen, og fortsatt beholde riktig statistikk, antall spørsmål og språk.

Navigasjonsmuligheter

BESKRIVELSE	SKJERMBILDE
<p>Hjem</p> <p>Spill: Starter spillet med antall valgte spørsmål.</p> <p>Preferanser: Bruker kan velge språk og antall spørsmål.</p> <p>Statistikk: Bruker kan se total statistikk.</p> <p>Tilbake: Går ut av PandaMath.</p>	
<p>Spill</p> <p>Spill igjen: Kan starte ny runde under samme spill, dersom det er flere spørsmål igjen.</p> <p>Neste: Denne knappen vil føre deg til et nytt spørsmål og oppdatere resultatene.</p> <p>Tilbake/Avslutt: Avslutter spill og navigerer til Hjem skjermen. Dialog dersom du ikke er ferdig, som spør: "Er du sikker", velg "Ja" eller "nei".</p>	

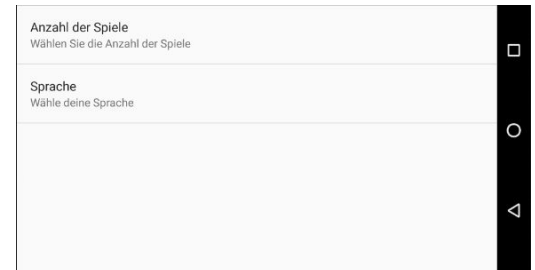
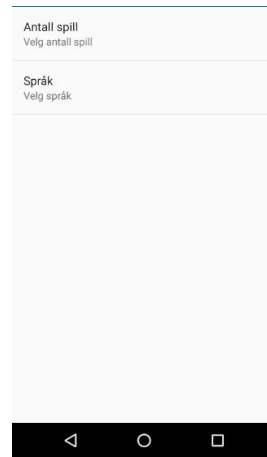
Preferanser

Gi brukeren mulighet til å endre innstillinger.

Antall spill: Bruker kan velge antall spørsmål.

Språk: Bruker kan velge mellom tysk og norsk.

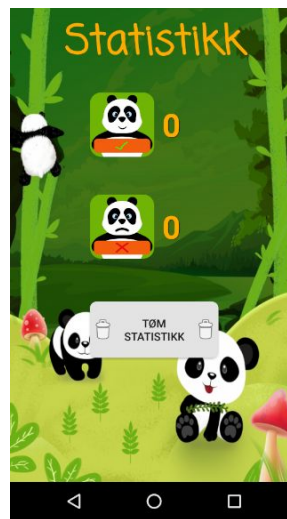
Tilbake: Navigerer tilbake til Hjem skjermen.



Statistikk

Tøm statistikk: Nullstiller statistikken. Dialog som spør: "Er du sikker", velg "Ja" eller "Nei".

Tilbake: Navigerer tilbake til Hjem skjermen.



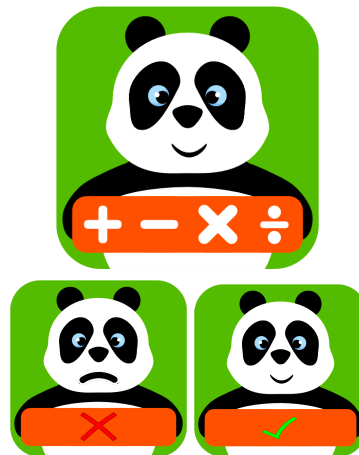
Design - fargevalg og ikoner

BESKRIVELSE	IKON
<p>Bakgrunn: Bakgrunnen lever til navnet og har et tema basert på pandaer. Fargene er livlige, og bakgrunnen inneholder gode detaljer. Med fargerike knapper på toppen av det hele, gir det et vakkert hjemmeside.</p>	

Logo: Logoen har samme farger som hjemmesiden og passer godt med både navn og utseende til applikasjonen.

Logoen blir brukt i spillet til å animere brukeren om det var et rett eller galt svar.

Logoen vises også i statistikkdelen hvor den viser den totale poengsummen.



Fargevalg:

Vi måtte sørge for at spillet både blir lærerikt og gøy.

Fargevalget vårt har vært bredt, der det er brukt rikelig med farger. Dette for at barn skal føle seg omgitt av kule farger. Vi har sørget for å ikke bruke sterke farger, når det kan være stimulerende for øyet. Vi har heller ikke brukt skarpe gule og røde farger som kan bli ukomfortabel for øyet over lang spilletid. Siden barn opplever farger forskjellig har vi holdt oss til nøytrale farger.

Oppgave-ramme: Vi har brukt en enkel ramme til gjøre det klart for brukeren hvor selve hendelsen foregår. Vi valgte å bruke mindre og lysere farger i selve spillet, siden dette vil gi bedre fokus.



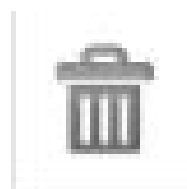
Spilltastatur:

Tallene er satt i standard-rekkefølge som er vanlig å finne i mobiltelefoner. Farge valget gjør det lett å se forskjell på knappene, så et barn på 6 år ikke har problemer med å taste det han har lyst på. Knappene er også større enn det som er default på en telefon.

Tastaturet er skreddersydd slik at bruker på lettest mulig måte kan taste inn svaret sitt.



Slett Statistikk: Som en selvfølge betyr søppelkassen å "kaste" noe. I dette tilfellet blir ikonet brukt på "Tøm statistikk" knappen både på høyre og venstre side. Dette hjelper med å skape gjenkjennelse hos brukeren, som i dette tilfellet er barn og kan være analfabete.



Logikk

Applikasjonen er implementert med hensyn til developer.android.com og forelesningsnotater.

Til felles tar klassene i bruk hjelpemetoder og hjelpeklasser. Disse kodene er felles for de forskjellige aktivitetklassene vi har:

For å endre språk:

For å sette riktig språk i applikasjonen henter vi språk-verdien fra `sharedpreferences`. Dette blir satt av bruker gjennom preferanser. Norsk vil være default språket. Når du lukker programmet og åpner det opp igjen vil det brukeren valgte sist fortsatt være det gyldige.

```
//SPRÅK
private void settLand(String landskode){
    Resources res = getResources();
    DisplayMetrics dm = res.getDisplayMetrics();
    Configuration cf = res.getConfiguration();
    cf.setLocale(new Locale(landskode));
    res.updateConfiguration(cf,dm);
}

private void hentLandskode(){
    String landskode = getSharedPreferences( name: "PREFERENCE", MODE_PRIVATE)
        .getString( s: "ValgtLanguage", s1: "no");
    settLand(landskode);
}
```

For å fjerne default navbar:

For å fjerne den øverste navbar'en som er default på telefoner bruker vi denne metoden. Denne kjører vi i klassene slik at vi får en enkel layout.

```
//LAYOUT
//Nav - Layout
private void layoutAndView() {
    requestWindowFeature(Window.FEATURE_NO_TITLE);
    getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
        WindowManager.LayoutParams.FLAG_FULLSCREEN);
}
```

Vi har implementert to hjelpeklasser som hjelper oss gjennom programmet:

Klassen `Toaster` holder styr på `Toast`, som er et popup melding til brukeren. Dette har vi laget en klasse av, og sørger for at vi kan lage en `Toast`, som enten er kort eller lang. Brukes for eksempel når bruker ikke har skrevet inn en verdi når han svarer på et spørsmål.

Dialog:

Klassen `Dialog` er brukt for å hente ut en dialogboks. For eksempel når bruker vil avslutte spillet før det er ferdig. Her er det brukt `Interface`. På denne måten har vi en felles løsning for alle situasjoner hvor vi trenger en dialogboks.

```
public class Toaster {
    //DATAFELT
    Context context;
    public Toaster(Context context) { this.context = context; }

    //Metode for kortvarig meldinger til brukeren
    public void makeShortToast( String msg) {
        Toast toast = Toast.makeText(this.context,msg,Toast.LENGTH_SHORT);
        toast.show();
    }

    //Metode for langvarig meldinger til brukeren
    public void makeLongToast(String msg) {
        Toast toast = Toast.makeText(this.context,msg,Toast.LENGTH_LONG);
        toast.show();
    }
}

//INTERFACE
public interface DialogClickListener {
    public void onPositiveClick();
    public void onNegativeClick();
}
```

Er du sikker?

NEI JA

```

@Override
public android.app.Dialog onCreateDialog(Bundle savedInstanceState) {
    return new AlertDialog.Builder(getActivity())
        .setTitle("Er du sikker?").setPositiveButton(
            "Ja", (dialogInterface, whichButton) -> {
                dialog.onPositiveClick();
            })
        .setNegativeButton("Nei", (dialogInterface, i) -> {
            dialog.onNegativeClick();
        })
        .create();
}

```

Splash

Splash-aktiviteten vår sørger for at vår logo vises i tre sekunder før brukeren blir ført til hovedaktiviteten. Denne vises ikke igjen dersom brukeren går ut av spillet, vises kun igjen dersom brukeren lukker spillet og åpner den igjen.

```

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_splash);

    ventStart.postDelayed(() -> {
        try {
            //starter en ny aktivitet(side). I dette tilfelle MainActivity
            Intent intent = new Intent(getApplicationContext(), MainActivity.class);
            startActivity(intent);
            finish(); //vil ikke vise det videre, f.eks. hvis en bruker trykker tilbake-knappen
        } catch (Exception ignored) {
            ignored.printStackTrace();
        }
    }, delayMillis: 3000);
}

```

Hjem

```
public class MainActivity extends AppCompatActivity
```

I Hjem-klassen har vi i Oncreate lagt til forskjellige metoder, slik at vi bruker god programmeringsskikk der all kode er lagt i metoder som blir kalt på ved nødvendighet. Metodene som blir kalt på her er blant annet *btnListner()* , *spillValgPreferanser()* og *onResume()*.

Vi har en *onClickListner* på knappene våre i hjemmesiden som fører deg til nye aktivitetssider (Spill, preferanser og statistikk).

```
//KNAPPER OG LISTENER
private void btnListener() {
    //Hva som skal skje når man trykker på StartSpill - knappen
    spillBtn.setOnClickListener((v) -> { åpneSpill(); });

    //Hva som skal skje når man trykker på preferanse - knappen
    preferanserBtn.setOnClickListener((v) -> { åpnePreferanser(); });

    //Hva som skal skje når man trykker på statistikk - knappen
    statistikkBtn.setOnClickListener((v) -> { åpneStatistikk(); });
}
```

SpillValgPreferanser() bruker *getSharedPreferences* og nøkkelen (*ValgtAktivitet*) for å hente ut antall spill som ble valgt i preferanse-skjerm bildet. Spillknappen på hjemskjermen blir satt til antall spill som ble valgt, dette vil hjelpe brukeren å se hvor mange spill han skal spille.

```
protected void spillValgPreferanser() {
    spillBtn.setText(getSharedPreferences( name: "PREFERENCE", MODE_PRIVATE).getString( s: "ValgtAktivitet", s1: "5")
        + " " + "Spill");
}
```

onResume() er en aktivitet i android-livet. Denne metoden aktiveres når brukeren returnerer til denne aktiviteten etter å ha vært gjennom en annen aktivitet. Vi sørger for å sjekke om antall aktiviteter og språk har blitt endret når man kommer tilbake til hovedaktiviteten, etter å ha vært gjennom andre aktiviteter.

```
@Override
public void onResume() {
    super.onResume();
    spillValgPreferanser();
    knappTekst();
    hentLandskode(); //Bytter språk
}
```

Spill

```
public class Spill extends AppCompatActivity implements Dialog.OnClickListener
```

Klassen har et datafelt med nødvendige variabler og hjelpe-klasser. Dette brukes for å unngå hardkoding, og heller ha en dynamisk klasse.

Variabler som *TOTAL_ANTALL* og *ANIMATION_TIME* settes som final, fordi

```
//DATAFELT
final int TOTAL_ANTALL = 25; // 25 spørsmål
final int ANIMATION_TIME = 1000; // 1 Sekund
int antall; //Oppdateres med preferanser
int antSvarteSpm, poeng, feil = 0;
ArrayList<String> regnestykkerUnik= new ArrayList<>();
ArrayList<Integer> fasitUnik = new ArrayList<>();
ArrayList<Integer> brukteRegnestykker = new ArrayList<>();
String[] regnestykker; //Hentes fra Arrays
int[] fasit; //Hentes fra Arrays
```


de er eksakte. Resterende variabler blir brukt for unngå duplikater og vil forandre seg utover i spillet. Hjelpklassene Toaster og Dialog er også brukt.

I *OnCreate* metoden blir språk satt, layout hentes og xml knapper importeres. Antall Spill, spørsmål og svar hentes fra sharedpreference og arrays.xml. Alle knapper får en Listener og en funksjon i en strukturert Switch-case metode.

Først vil en funksjon velge et "antall" spørsmål og gjøre de klar til bruker. Vi bruker en while løkke som bare vil legge til et spørsmål dersom det ikke er brukt. Når brukeren vil spille en ny runde, fungerer metoden fortsatt. Dersom det ikke er flere spørsmål igjen, vil den aldri gå inn i metoden, og brukeren vil få en melding.

```
// Unike spørsmål uten duplikat
private void randomSpill(){
    int i = 0;
    while(i < antall){
        int random = (int) (Math.random()*TOTAL_ANTALL);
        if(!brukteRegnestykker.contains(random)){
            brukteRegnestykker.add(random);
            regnestykkerUnik.add(regnestykker[random]);
            fasitUnik.add(fasit[random]);
            i++;
        }
    }
}
```

Venstre bilde: Brukeren har svart på 10 spørsmål og kan velge mellom å starte på nytt eller avslutte.

Høyre bilde: Brukeren har svart på 20 spørsmål, og har ikke flere spørsmål til å starte en ny runde.



```
//Gameover
private void rundeFerdig() {
    LinearLayout.setVisibility(View.GONE);
    rundeFerdigLayout.setVisibility(View.VISIBLE);
    rundeFerdigTxt.setVisibility(View.VISIBLE);
    svarBtn.setEnabled(false);
    svarTxt.setHint("");
    hjemBtn.setVisibility(View.VISIBLE);
    statistik();

    if((brukteRegnestykker.size()+antall) >= TOTAL_ANTALL){
        spillIgjenBtn.setVisibility(View.GONE);
        rundeFerdigTxt.setText("Du har ingen flere spørsmål!");
    }else {
        rundeFerdigTxt.setText("Vil du spille igjen?");
        spillIgjenBtn.setVisibility(View.VISIBLE);
    }
}
```



På slutten av hver runde blir poengsummen lagt til i Statistikken. Dette gjøres ved å hente (*getInt("totalpoeng",totalPoeng)*) Sharedpreference verdiene, addere nåværende poeng og sende nye verdiene til statistikken (*edit().putInt("totalpoeng",totalPoeng).apply()*).

```
//Oppdater Statistikk
private void statistikk(){
    SharedPreferences sharedPreferences = getSharedPreferences( name: "PREFERENCE",MODE_PRIVATE);

    int totalPoeng;
    int totalFeil;

    totalPoeng = sharedPreferences.getInt( s: "totalPoeng", i: 0);
    totalFeil = sharedPreferences.getInt( s: "totalFeil", i: 0);

    totalPoeng += poeng;
    totalFeil += feil;

    //Lager totalPoeng
    sharedPreferences
        .edit()
        .putInt( s: "totalPoeng", totalPoeng)
        .apply();

    //Lager TotalFeil
    sharedPreferences
        .edit()
        .putInt( s: "totalFeil", totalFeil)
        .apply();
}
```

For å sjekke om brukeren har svart riktig, så sammenligner vi svaret til brukeren med arrayet fra arrays.xml. Forskjellig animasjoner vil bli vist om brukeren svarer riktig eller galt. Dette skal hjelpe PandaMath med å gjøre morsomt og lærerikt. Om svaret er tomt, vil det komme opp en toast.



```
private void animation(ImageView image){
    final ImageView tmpImage = image;
    tmpImage.setVisibility(View.VISIBLE);
    final Drawable d = LinearLayout.getBackground();
    LinearLayout.setBackground(null);

    tmpImage.postDelayed(new Runnable() {
        @Override
        public void run() {
            tmpImage.setVisibility(View.GONE);
            LinearLayout.setBackground(d);
        }
    }, ANIMATION_TIME); //ET SEKUND
}
```



Tilstandsbevaringen fungerer ved at onSaveInstanceState lagrer data når for eksempel emulator roteres, og ved hjelp av restoreInstanceState(), får vi tak i den siste dataen og alt holdes oppdatert. Dersom applikasjonen starter for første gang vil spillet starte som vanlig.

```
//TILSTANDSBEVARING
@Override
protected void onSaveInstanceState(Bundle outState) {
    outState.putInt("ant_svar_spm", antSvarSpm);
    outState.putStringArrayList("regnestykker_unik", regnestykkerUnik);
    outState.putIntArrayList("fasit_unik", fasitUnik);
    outState.putInt("poeng", poeng);
    outState.putInt("feil", feil);
    super.onSaveInstanceState(outState);
}

private void restoreInstance(Bundle savedInstanceState){
    //Tilstandsbevaring
    if(savedInstanceState != null){
        antSvarSpm = savedInstanceState.getInt( key: "ant_svar_spm");
        regnestykkerUnik = savedInstanceState.getStringArrayList( key: "regnestykker_unik");
        fasitUnik = savedInstanceState.getIntegerArrayList( key: "fasit_unik");
        poeng = savedInstanceState.getInt( key: "poeng");
        feil = savedInstanceState.getInt( key: "feil");
        antallRiktigTxt.setText(String.valueOf(poeng));
        antallFeilTxt.setText(String.valueOf(feil));
    }else{
        randomSpill();
    }
}
```

Preferanser

```
public static class PrefsFragment extends PreferenceFragment
```

Preferanser klassen vår brukes for at brukeren kan endre antall spill og språk. I denne klassen bruker vi fragmenter av typen *PreferenceScreen*.

I *PreferenceScreen* har vi to *ListPreference*, som lister ut verdiene vi vil vise. All tekst er hentet fra strings.xml og verdiene som vises hentes fra arrays.xml.

```
<PreferenceScreen xmlns:android="http://schemas.android.com/apk/res/android">
```

```
    <ListPreference
        android:key="antallSpill"
        android:summary="Velg antall spill"
        android:title="Antall spill"
        android:entries="@array/antallSpill_value"
        android:entryValues="@array/antallSpill_value"
        android:defaultValue="5">
    </ListPreference>
```

```
    <ListPreference
        android:key="language"
        android:title="Språk"
        android:summary="@string/velgSprak"
        android:entries="@array/spraak"
        android:entryValues="@array/spraakvalg"
        android:defaultValue="Norsk">
    </ListPreference>
```

I onCreate metoden har vi *prefListener()*. Metoden bruker en *onSharedPreferenceChangeListener* som blir kalt på når en preferanse verdi blir endret. F.eks. hvis en bruker velger et annet språk eller antall spill. Verdien blir sammenlignet med key i *PreferenceScreen* og kjører ut ifra hvilken *ListPreference* som har blitt endret.

```
//LISTENER
private void prefListener() {
    sharedPreferences = PreferenceManager.getDefaultSharedPreferences(context, this);
    prefListener = (OnSharedPreferenceChangeListener) (sharedPreferences, s) -> {
        if (s.equals("language")) {
            language();
            recreate();
        }
        if (s.equals("antallSpill")) {
            preferences();
        }
    };
    sharedPreferences.registerOnSharedPreferenceChangeListener(prefListener);
}
```

Metodene *language()* og *preferences()* bruker *getSharedPreferences* for å lagre og manipulere dataene. I begge metodene brukes *getSharedPreferences* for å lagre verdiene fra *ListPreference*. Det blir gitt en nøkkelverdi til *getSharedPreferences* som blir brukt når vi skal hente ut verdiene vi har lagret. Verdien(valgt) blir satt til det som brukeren har valgt som alternativ.

```
//Antall spill
protected void preferences() {
    sharedPreferences = PreferenceManager.getDefaultSharedPreferences( context: this);
    String valgt = sharedPreferences.getString( s: "antallSpill", s1: "");
    getSharedPreferences( name: "PREFERENCE",MODE_PRIVATE)
        .edit().putString( s: "ValgtAktivitet", valgt).apply();
}

//Språk
protected void language() {
    sharedPreferences = PreferenceManager.getDefaultSharedPreferences( context: this);
    String valgt = sharedPreferences.getString( s: "language", s1: "");
    getSharedPreferences( name: "PREFERENCE",MODE_PRIVATE)
        .edit().putString( s: "ValgtLanguage", valgt).apply();
}
```

Statistikk

```
public class Statistikk extends AppCompatActivity implements Dialog.OnClickListener
```

Statistikk klassen har i oppgave å vise brukeren total sum av poeng og feil. Etter å ha importert riktig språk, satt opp xml og importert variabler, blir det utført en metode som oppdaterer statistikken:

Metoden *oppdaterStat()* henter verdiene fra nøklene totalPoeng og totalFeil, fra sharedPreferences og oppdaterer riktigStatTxt og feilStatTxt.

```
private void oppdaterStat(){
    SharedPreferences sharedPreferences = getSharedPreferences( name: "PREFERENCE",MODE_PRIVATE);

    int totalPoeng = sharedPreferences.getInt( s: "totalPoeng", i: 0);
    int totalFeil = sharedPreferences.getInt( s: "totalFeil", i: 0);

    riktigStatTxt.setText(String.valueOf(totalPoeng));
    feilStatTxt.setText(String.valueOf(totalFeil));
}
```

Når brukeren skal nullstille statistikken, for han opp en dialog som er arvet fra Dialog klassen, og dialog fragmentet kjøres. Det kommer opp en dialogboks, som ber om bekreftelse. Dersom bruker trykker "Ja" vil sharedPreferences verdiene til totalPoeng og totalFeil nullstilles.

```
@Override
public void onPositiveClick() {
    SharedPreferences sharedPreferences = getSharedPreferences( name: "PREFERENCE",MODE_PRIVATE);

    //Nullstiller totalPoeng
    sharedPreferences
        .edit()
        .putInt( s: "totalPoeng", i: 0)
        .apply();

    //Nullstiller TotalFeil
    sharedPreferences
        .edit()
        .putInt( s: "totalFeil", i: 0)
        .apply();

    riktigStatTxt.setText("0");
    feilStatTxt.setText("0");
}
```