Inledning

- Projektet är en Tkinter-UI baserad applikation där användaren kan undersöka energiproduktion från ett vindkraftverk och ett solkraftverk per månad. Användaren kan välja mellan en tabellarisk och en grafisk representation av data och använda sliders för att justera variabler som soltal, area, latitud och rotordiameter. Alla dessa parametrar uppdaterar data i realtid.
- En utmaning är att med matplotlib framställa uppdateringsbara grafer och tabeller. Datat måste rensas mellan uppdateringarna utan att tidigare värden lämnas kvar. En annan utmaning är att undvika koduppreprepningar eftersom både sol- och vindkraftverk genererar liknande data som har små skillnader (t.ex. rotordiameter för vindkraftverk och area för solkraftverk).
- Projektet har begränsningar i noggrannheten för beräkningarna. Exempelvis är energiproduktionen per dag baserad på en slumpmässig solighetsfaktor som är en float mellan 0 och 1, vilket innebär att månatliga värden kan variera, och detta kan minska noggrannheten i resultaten.

Användarscenarier

- Niklas driver Solar Energy AB och behöver avgöra var han ska expandera solcellsnätverket, i Norrland eller Skåne. Solceller är dyrare i Skåne, vilket innebär en mindre total area där. Han startar programmet och väljer solcell och tabellarisk representation av datat, därefter får han upp en tabell som han kan förändra genom att använda sliders för soltal, area och latitud. Han uppskattar sedan den årliga energiproduktion i båda områdena genom att addera månadsvärden i tabellvyn. Detta ger honom direkt en förståelse för om han bör installera solcellerna i Norrland med större area eller i Skåne med mindre area men mer sol.
- Agnes vill bygga ett vindkraftverk nära sin sommarstuga och undersöker om det kan generera tillräckligt med el under sommaren. Hon startar programmet och väljer vindkraftverk och grafisk vy. Sedan trycker hon på kör programmet. Detta ger henne ett stapeldiagramm där hon kan justera rotordiamtern och latitud efter hennes förutsättningar och analysera produktionen per säsong. Om energiproduktionen under sommaren är mindre än 30% av vinterproduktionen, planerar hon att installera solpaneler istället.

Kodskelett

```
class Solcell:
    """Denna klass returnerar en lista som kallas månadsdata som innehåller 12
    listor med W-värden för varje dag. Denna lista används för att
    beräkna statistik. För att skapa denna lista behövs  arean,
    soltalet och latituden."""

    def __init__(self, area, soltal, latitud):
        """Initierar viktiga parametrar"""

    def genereraW(self, dag):
        """Genererar energi per dag W som används i årsproduktion och tabell"""

    def returnerarmånadsdata(self):
        """Returnerar en lista med 12 månadslistor av dagliga W-värden"""
```

```
class Vindturbin:
    """Denna klass returnerar en lista som kallas månadsdata som innehåller 12
    listor med W-värden för varje dag. Denna lista används för att beräkna
    statistik. För att skapa denna lista behövs rotordiametern och latituden"""

    def genereraW(self, dag):
        """Genererar energi per dag W baserat på vindfaktor
        och årstidsfaktor"""

    def årstidsfaktor(self, dag):
        """beräknar årstidsfaktorn baserat på dagen"""

    def returnerarmånadsdata(self):
        """Returnerar en lista med 12 månadslistor av dagliga W-värden"""



class Diagram:
    def __init__(self, area, soltal, latitud, rotordiameter=None):
        """Denna klass definierar alla viktiga värden med hjälp av
        Solcells-klassen och Vindturbin-klassen och skapar diagrammet"""

    def skapasliders(self):
        """Denna implementering skapar alla sliders som ser till att man
        kan justera alla parametrar"""

    def skapastaplar(self):
        """Skapar en tabell med medelvärde, standardavvikelse, max och min
        för varje månad. För att skapa dessa staplar nyttjas funktionen
        beräkna_statistik"""

    def uppdatera(self, val):
        """Uppdaterar diagrammet när en slider ändras"""

    def show(self):
        """Visar det skapade diagrammet"""



class Tabell(Diagram):
    def __init__(self, area, soltal, latitud, rotordiameter=None):
        """Tabellklass som ärver från Diagram och visar data i tabellform"""

    def skapatabell(self):
        """Skapar en tabell med medelvärde, standardavvikelse, max och min
        för varje månad. För att skapa dessa staplar nyttjas funktionen
        beräkna_statistik"""

    def uppdatera(self, val):
        """Uppdaterar tabellen när en slider ändras"""

    def show(self):
```

```
        """Visar tabellen"""


def beräkna_statistik(månadsdata, alternativ):
    """Returnerar specifik statistik (medelvärde, standardavvikelse, min, max)
    för varje månad baserat på valt alternativ. Datat för denna statistik
    hämtas ifrån listan med W-värdenn som genereras i Solcells- och
    Vindturbins klassen"""

def f(dag, latitud):
    """Tar in vilken dag och latitud som anges av användaren"""

def startatkinter():
    """Denna funktion öppnar upp ett fönster med radioknappar och en knapp som
    kör programmet"""

def nyttfönster(valavkraftverk, valavdata):
    """Skapar ett nytt fönster för programmet"""

def aktivera_matplotlib(representation_var, kraftverk_var):
    """Beroende på vilka värden som radioknapparna har så anropas olika klasser
    med olika argument"""

def main():
    """Huvudfunktion som startar tkinter"""
```

Minnet

- All data genereras i klasserna Solcell och Vindturbin och används sen av funktionen beräknastatistik för att beräkna statistik som Diagram och Tabell klasserna behöver. Dessa klasser presenterar sedan datat tabellariskt eller använder den för att justera höjden på staplar i ett diagram. Funktionerna som hanterar Tkinter kallar endast på Diagram- och Tabell-klasserna, de fungerar genom att hantera användarens input ifrån Tkinter fönstret. Värdena som knapparna motsvarar anropar Diagram eller Tabell klasserna vilket leder till att ett nytt fönster öppnas. Input samlas alltås enbart in från sliders i respektive klass och ifrån Tkinter funktionerna. Man kan lägga märke till att ingen information av användaren sparas (förutom att öppnade fönster hålls öppna) utan programmet representerar enbart data i realtid.