Computer Project File

Code:-

```
import tkinter as tk
from tkinter import ttk, messagebox
from datetime import datetime
from tkcalendar import DateEntry  # Install: pip install tkcalendar

class MultiCalculatorApp:
    def __init__(self, root):
        self.root = root
        self.root.title("Multi-Calculator Application")
        self.root.geometry("700x550")
        self.root.config(bg="#f0f0f0")

        # Title
        title_label = tk.Label(
            self.root,
            text="Multi-Purpose Calculator",
            font=("Arial", 26, "bold"),
            bg="#f0f0f0",
            fg="#2c3e50"
        )
        title_label.pack(pady=20)

        # Create Notebook (Tabbed Interface)
        self.notebook = ttk.Notebook(self.root)
        self.notebook.pack(expand=True, fill='both', padx=20, pady=10)

        # Create Tabs
        self.create_unit_converter_tab()
        self.create_age_calculator_tab()
        self.create_bmi_calculator_tab()

    # ==========================================
    # TAB 1: UNIT CONVERTER
    # ==========================================
    def create_unit_converter_tab(self):
        # Create frame for Unit Converter
        unit_frame = tk.Frame(self.notebook, bg="white")
        self.notebook.add(unit_frame, text=" 🔄 Unit Converter")

        # Main container
        container = tk.Frame(unit_frame, bg="white")
```

```
container.pack(expand=True, fill='both', padx=30, pady=30)

# Title
tk.Label(
    container,
    text="Unit Converter",
    font=("Arial", 20, "bold"),
    bg="white",
    fg="#3498db"
).pack(pady=(0, 20))

# Conversion Type Selection
type_frame = tk.Frame(container, bg="white")
type_frame.pack(pady=10)

tk.Label(
    type_frame,
    text="Select Conversion Type:",
    font=("Arial", 12),
    bg="white"
).pack(side=tk.LEFT, padx=5)

self.conversion_type = ttk.Combobox(
    type_frame,
    values=["Temperature", "Weight", "Length"],
    font=("Arial", 11),
    state="readonly",
    width=15
)
self.conversion_type.pack(side=tk.LEFT, padx=5)
self.conversion_type.current(0)
self.conversion_type.bind("<<ComboboxSelected>>", self.update_unit_options)

# Input Frame
input_frame = tk.LabelFrame(
    container,
    text="Input",
    font=("Arial", 12, "bold"),
    bg="white",
    padx=20,
    pady=15
)
input_frame.pack(pady=15, fill='x')
```

```
# Value Entry
value_frame = tk.Frame(input_frame, bg="white")
value_frame.pack(pady=5)

tk.Label(
    value_frame,
    text="Value:",
    font=("Arial", 11),
    bg="white"
).pack(side=tk.LEFT, padx=5)

self.unit_value = tk.Entry(value_frame, font=("Arial", 11), width=20)
self.unit_value.pack(side=tk.LEFT, padx=5)

# From Unit
from_frame = tk.Frame(input_frame, bg="white")
from_frame.pack(pady=5)

tk.Label(
    from_frame,
    text="From:",
    font=("Arial", 11),
    bg="white"
).pack(side=tk.LEFT, padx=5)

self.from_unit = ttk.Combobox(
    from_frame,
    font=("Arial", 11),
    state="readonly",
    width=18
)
self.from_unit.pack(side=tk.LEFT, padx=5)

# To Unit
to_frame = tk.Frame(input_frame, bg="white")
to_frame.pack(pady=5)

tk.Label(
    to_frame,
    text="To:",
    font=("Arial", 11),
    bg="white"
).pack(side=tk.LEFT, padx=5)
```

```
        self.to_unit = ttk.Combobox(
            to_frame,
            font=("Arial", 11),
            state="readonly",
            width=18
        )
        self.to_unit.pack(side=tk.LEFT, padx=5)

        # Convert Button
        tk.Button(
            container,
            text="Convert",
            font=("Arial", 12, "bold"),
            bg="#3498db",
            fg="white",
            padx=30,
            pady=10,
            command=self.convert_units,
            cursor="hand2"
        ).pack(pady=15)

        # Result Display
        self.unit_result = tk.Label(
            container,
            text="Result will appear here",
            font=("Arial", 14),
            bg="#ecf0f1",
            fg="#2c3e50",
            padx=20,
            pady=15,
            relief=tk.RIDGE
        )
        self.unit_result.pack(pady=10, fill='x')

        # Initialize unit options
        self.update_unit_options()

    def update_unit_options(self, event=None):
        conversion_type = self.conversion_type.get()

        if conversion_type == "Temperature":
            units = ["Celsius", "Fahrenheit", "Kelvin"]
        elif conversion_type == "Weight":
            units = ["Kilograms", "Pounds", "Grams", "Ounces"]
```

```
    else:  # Length
        units = ["Meters", "Feet", "Kilometers", "Miles", "Centimeters", "Inches"]

    self.from_unit['values'] = units
    self.to_unit['values'] = units
    self.from_unit.current(0)
    self.to_unit.current(1)

def convert_units(self):
    try:
        value = float(self.unit_value.get())
        from_u = self.from_unit.get()
        to_u = self.to_unit.get()
        conv_type = self.conversion_type.get()

        if conv_type == "Temperature":
            result = self.convert_temperature(value, from_u, to_u)
        elif conv_type == "Weight":
            result = self.convert_weight(value, from_u, to_u)
        else:
            result = self.convert_length(value, from_u, to_u)

        self.unit_result.config(
            text=f"{value} {from_u} = {result:.4f} {to_u}",
            fg="#27ae60"
        )
    except ValueError:
        messagebox.showerror("Error", "Please enter a valid number!")

def convert_temperature(self, value, from_unit, to_unit):
    # Convert to Celsius first
    if from_unit == "Fahrenheit":
        celsius = (value - 32) * 5/9
    elif from_unit == "Kelvin":
        celsius = value - 273.15
    else:
        celsius = value

    # Convert from Celsius to target
    if to_unit == "Fahrenheit":
        return celsius * 9/5 + 32
    elif to_unit == "Kelvin":
        return celsius + 273.15
    else:
```

```
        return celsius

    def convert_weight(self, value, from_unit, to_unit):
        # Convert to kilograms first
        to_kg = {
            "Kilograms": 1,
            "Pounds": 0.453592,
            "Grams": 0.001,
            "Ounces": 0.0283495
        }

        kg_value = value * to_kg[from_unit]
        return kg_value / to_kg[to_unit]

    def convert_length(self, value, from_unit, to_unit):
        # Convert to meters first
        to_meters = {
            "Meters": 1,
            "Feet": 0.3048,
            "Kilometers": 1000,
            "Miles": 1609.34,
            "Centimeters": 0.01,
            "Inches": 0.0254
        }

        meter_value = value * to_meters[from_unit]
        return meter_value / to_meters[to_unit]

    # ========================================
    # TAB 2: AGE CALCULATOR
    # ========================================
    def create_age_calculator_tab(self):
        # Create frame for Age Calculator
        age_frame = tk.Frame(self.notebook, bg="white")
        self.notebook.add(age_frame, text=" 🗓️ Age Calculator")

        # Main container
        container = tk.Frame(age_frame, bg="white")
        container.pack(expand=True, fill='both', padx=30, pady=30)

        # Title
        tk.Label(
            container,
            text="Age Calculator",
```

```
    font=("Arial", 20, "bold"),
    bg="white",
    fg="#e74c3c"
).pack(pady=(0, 30))

# Input Frame
input_frame = tk.LabelFrame(
    container,
    text="Enter Birth Details",
    font=("Arial", 12, "bold"),
    bg="white",
    padx=30,
    pady=20
)
input_frame.pack(pady=15)

# Date of Birth
dob_frame = tk.Frame(input_frame, bg="white")
dob_frame.pack(pady=10)

tk.Label(
    dob_frame,
    text="Date of Birth:",
    font=("Arial", 12),
    bg="white"
).pack(side=tk.LEFT, padx=10)

# Note: Using simple entry fields instead of DateEntry
# Day
tk.Label(dob_frame, text="Day:", font=("Arial", 10), bg="white").pack(side=tk.LEFT)
self.birth_day = tk.Spinbox(dob_frame, from_=1, to=31, width=5, font=("Arial", 10))
self.birth_day.pack(side=tk.LEFT, padx=5)

# Month
tk.Label(dob_frame, text="Month:", font=("Arial", 10), bg="white").pack(side=tk.LEFT)
self.birth_month = tk.Spinbox(dob_frame, from_=1, to=12, width=5, font=("Arial", 10))
self.birth_month.pack(side=tk.LEFT, padx=5)

# Year
tk.Label(dob_frame, text="Year:", font=("Arial", 10), bg="white").pack(side=tk.LEFT)
self.birth_year = tk.Spinbox(dob_frame, from_=1900, to=2025, width=8, font=("Arial", 10))
self.birth_year.delete(0, tk.END)
self.birth_year.insert(0, "2000")
self.birth_year.pack(side=tk.LEFT, padx=5)
```

```
    # Calculate Button
    tk.Button(
        container,
        text="Calculate Age",
        font=("Arial", 12, "bold"),
        bg="#e74c3c",
        fg="white",
        padx=30,
        pady=10,
        command=self.calculate_age,
        cursor="hand2"
    ).pack(pady=20)

    # Result Frame
    result_frame = tk.LabelFrame(
        container,
        text="Your Age",
        font=("Arial", 12, "bold"),
        bg="white",
        padx=20,
        pady=15
    )
    result_frame.pack(pady=10, fill='x')

    self.age_result = tk.Label(
        result_frame,
        text="Your age will be displayed here",
        font=("Arial", 13),
        bg="white",
        fg="#2c3e50",
        justify=tk.LEFT
    )
    self.age_result.pack(pady=10)

def calculate_age(self):
    try:
        # Get birth date
        day = int(self.birth_day.get())
        month = int(self.birth_month.get())
        year = int(self.birth_year.get())

        birth_date = datetime(year, month, day)
        today = datetime.now()
```

```
        # Calculate age
        years = today.year - birth_date.year
        months = today.month - birth_date.month
        days = today.day - birth_date.day

        # Adjust if necessary
        if days < 0:
            months -= 1
            # Get days in previous month
            if today.month == 1:
                prev_month_days = 31
            else:
                prev_month = today.month - 1
                if prev_month in [1, 3, 5, 7, 8, 10, 12]:
                    prev_month_days = 31
                elif prev_month in [4, 6, 9, 11]:
                    prev_month_days = 30
                else:
                    prev_month_days = 29 if today.year % 4 == 0 else 28
            days += prev_month_days

        if months < 0:
            years -= 1
            months += 12

        # Calculate total days
        total_days = (today - birth_date).days

        result_text = f"""
Years: {years}
Months: {months}
Days: {days}

Total Days: {total_days:,}
Total Months: {years * 12 + months}
        """

        self.age_result.config(text=result_text, fg="#27ae60")

    except ValueError:
        messagebox.showerror("Error", "Please enter a valid date!")

# ==========================================
```

```
# TAB 3: BMI CALCULATOR
# ========================================
def create_bmi_calculator_tab(self):
    # Create frame for BMI Calculator
    bmi_frame = tk.Frame(self.notebook, bg="white")
    self.notebook.add(bmi_frame, text=" ⚖️ BMI Calculator")

    # Main container
    container = tk.Frame(bmi_frame, bg="white")
    container.pack(expand=True, fill='both', padx=30, pady=30)

    # Title
    tk.Label(
        container,
        text="BMI Calculator",
        font=("Arial", 20, "bold"),
        bg="white",
        fg="#9b59b6"
    ).pack(pady=(0, 20))

    # Input Frame
    input_frame = tk.LabelFrame(
        container,
        text="Enter Your Details",
        font=("Arial", 12, "bold"),
        bg="white",
        padx=30,
        pady=20
    )
    input_frame.pack(pady=15)

    # Weight Input
    weight_frame = tk.Frame(input_frame, bg="white")
    weight_frame.pack(pady=10)

    tk.Label(
        weight_frame,
        text="Weight:",
        font=("Arial", 12),
        bg="white",
        width=10,
        anchor='w'
    ).pack(side=tk.LEFT, padx=5)
```

```
self.weight_entry = tk.Entry(weight_frame, font=("Arial", 11), width=15)
self.weight_entry.pack(side=tk.LEFT, padx=5)

self.weight_unit = ttk.Combobox(
    weight_frame,
    values=["kg", "lbs"],
    font=("Arial", 10),
    state="readonly",
    width=8
)
self.weight_unit.pack(side=tk.LEFT, padx=5)
self.weight_unit.current(0)

# Height Input
height_frame = tk.Frame(input_frame, bg="white")
height_frame.pack(pady=10)

tk.Label(
    height_frame,
    text="Height:",
    font=("Arial", 12),
    bg="white",
    width=10,
    anchor='w'
).pack(side=tk.LEFT, padx=5)

self.height_entry = tk.Entry(height_frame, font=("Arial", 11), width=15)
self.height_entry.pack(side=tk.LEFT, padx=5)

self.height_unit = ttk.Combobox(
    height_frame,
    values=["cm", "meters", "feet"],
    font=("Arial", 10),
    state="readonly",
    width=8
)
self.height_unit.pack(side=tk.LEFT, padx=5)
self.height_unit.current(0)

# Calculate Button
tk.Button(
    container,
    text="Calculate BMI",
    font=("Arial", 12, "bold"),
```

```
            bg="#9b59b6",
            fg="white",
            padx=30,
            pady=10,
            command=self.calculate_bmi,
            cursor="hand2"
        ).pack(pady=20)

        # Result Frame
        result_frame = tk.LabelFrame(
            container,
            text="Your BMI Result",
            font=("Arial", 12, "bold"),
            bg="white",
            padx=20,
            pady=15
        )
        result_frame.pack(pady=10, fill='x')

        self.bmi_result = tk.Label(
            result_frame,
            text="Your BMI will be displayed here",
            font=("Arial", 13),
            bg="white",
            fg="#2c3e50",
            justify=tk.LEFT
        )
        self.bmi_result.pack(pady=10)

    def calculate_bmi(self):
        try:
            # Get weight in kg
            weight = float(self.weight_entry.get())
            if self.weight_unit.get() == "lbs":
                weight = weight * 0.453592  # Convert to kg

            # Get height in meters
            height = float(self.height_entry.get())
            if self.height_unit.get() == "cm":
                height = height / 100  # Convert to meters
            elif self.height_unit.get() == "feet":
                height = height * 0.3048  # Convert to meters

            # Calculate BMI
```

```
            bmi = weight / (height ** 2)

            # Determine category
            if bmi < 16:
                category = "Underweight (Severe thinness)"
                color = "#e74c3c"
            elif bmi < 17:
                category = "Underweight (Moderate thinness)"
                color = "#e67e22"
            elif bmi < 18.5:
                category = "Underweight (Mild thinness)"
                color = "#f39c12"
            elif bmi < 25:
                category = "Normal range"
                color = "#27ae60"
            elif bmi < 30:
                category = "Overweight (Pre-obese)"
                color = "#f39c12"
            elif bmi < 35:
                category = "Obese (Class I)"
                color = "#e67e22"
            elif bmi < 40:
                category = "Obese (Class II)"
                color = "#e74c3c"
            else:
                category = "Obese (Class III)"
                color = "#c0392b"

            result_text = f"""
Your BMI: {bmi:.2f}

Category: {category}

Weight: {weight:.2f} kg
Height: {height:.2f} meters
            """

            self.bmi_result.config(text=result_text, fg=color)

        except ValueError:
            messagebox.showerror("Error", "Please enter valid numbers for weight and height!")

# Run the application
if __name__ == "__main__":
```

```
root = tk.Tk()
app = MultiCalculatorApp(root)
root.mainloop()
```

Output :-