TEB1043 Object Oriented Programming

Department of Computer and Information Sciences

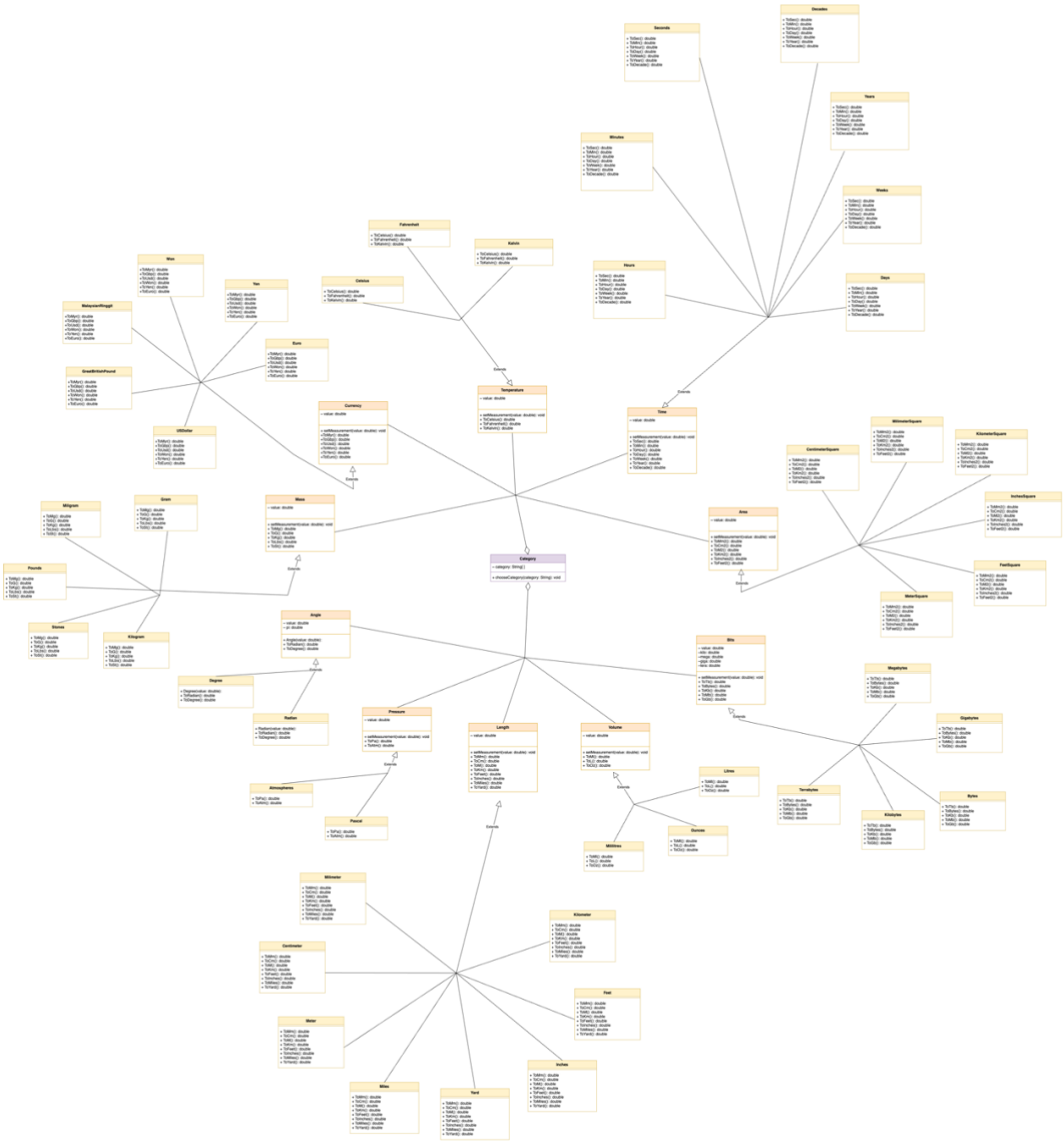
Prepared for:

Dr Mohamed Nordin bin Zakaria

Prepared by:

Name	Student ID	Program
Ahmad Zafri bin Mohd Ramli	21000556	Bachelor of Computer Science
Iman Kamil bin Suhaimi	21000175	Bachelor of Computer Science
Ammar Afif bin Mohd Khairie	21001232	Bachelor of Computer Science
Muhammad Aqil Fauzan bin Muhammad Hafidzi	21001113	Bachelor of Computer Science
Mohamad Izhar bin Ahmad Nordin	21001242	Bachelor of Computer Science
Danish Hariz bin Yusrizal	21001071	Bachelor of Computer Science
Imran Haqeem bin Mohd Khassan	21001366	Bachelor of Computer Science
Ikhwan Hanafi bin Irwan	21001245	Bachelor of Computer Science
Aiman Shuhaikal bin Shuhaimi	21001293	Bachelor of Computer Science
Muhammad Faris bin Mohd Rusli	21001346	Bachelor of Computer Science

Project UML

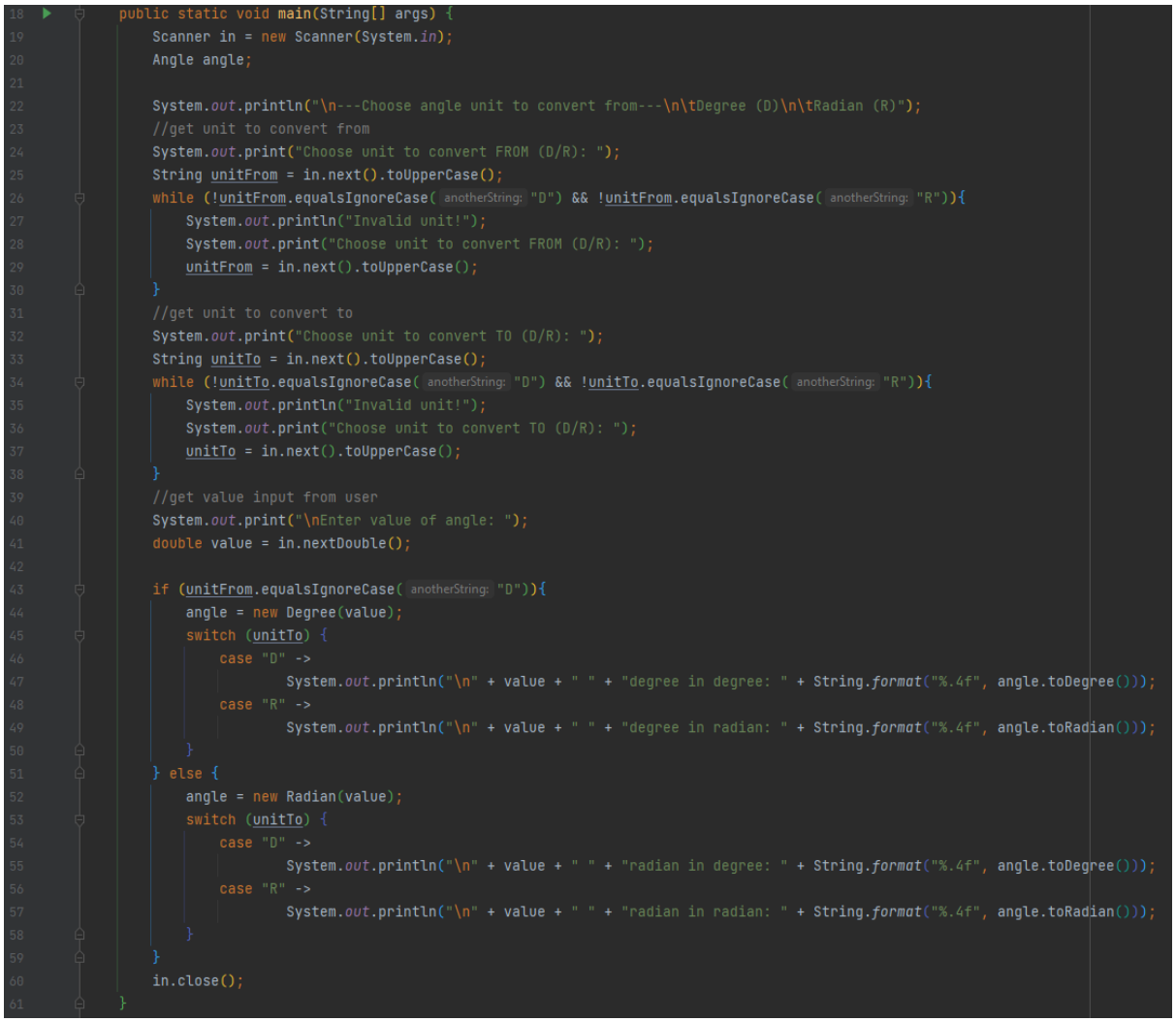


Members of Each Class

Class	Assignee
Length	Imran Haqeem
Mass	Muhammad Faris
Time	Aiman Shuhaikal
Temperature	Mohamad Izhar
Currency	Iman Kamil
Angle Category	Ammar Afif
Pressure	Ikhwan Hanafi
Area	Danish Hariz
Volume	Ahmad Zafri
Bits	Aqil Fauzan

Angle

Main Function (Test Code for Angle)



```
public static void main(String[] args) {
    Scanner in = new Scanner(System.in);
    Angle angle;

    System.out.println("\n---Choose angle unit to convert from---\n\tDegree (D)\n\tRadian (R)");
    //get unit to convert from
    System.out.print("Choose unit to convert FROM (D/R): ");
    String unitFrom = in.next().toUpperCase();
    while (!unitFrom.equalsIgnoreCase( anotherString: "D") && !unitFrom.equalsIgnoreCase( anotherString: "R")){
        System.out.println("Invalid unit!");
        System.out.print("Choose unit to convert FROM (D/R): ");
        unitFrom = in.next().toUpperCase();
    }
    //get unit to convert to
    System.out.print("Choose unit to convert TO (D/R): ");
    String unitTo = in.next().toUpperCase();
    while (!unitTo.equalsIgnoreCase( anotherString: "D") && !unitTo.equalsIgnoreCase( anotherString: "R")){
        System.out.println("Invalid unit!");
        System.out.print("Choose unit to convert TO (D/R): ");
        unitTo = in.next().toUpperCase();
    }
    //get value input from user
    System.out.print("\nEnter value of angle: ");
    double value = in.nextDouble();

    if (unitFrom.equalsIgnoreCase( anotherString: "D")){
        angle = new Degree(value);
        switch (unitTo) {
            case "D" ->
                    System.out.println("\n" + value + " " + "degree in degree: " + String.format("%.4f", angle.toDegree()));
            case "R" ->
                    System.out.println("\n" + value + " " + "degree in radian: " + String.format("%.4f", angle.toRadian()));
        }
    } else {
        angle = new Radian(value);
        switch (unitTo) {
            case "D" ->
                    System.out.println("\n" + value + " " + "radian in degree: " + String.format("%.4f", angle.toDegree()));
            case "R" ->
                    System.out.println("\n" + value + " " + "radian in radian: " + String.format("%.4f", angle.toRadian()));
        }
    }
    in.close();
}
```

Angle Class

```
2 inheritors
public abstract class Angle {

    //attributes
    protected double value;
    protected final double pi = Math.PI;

    //constructor
    public Angle(double value) { this.value = value; }

    //abstract methods headers
    2 implementations
    public abstract double toDegree();
    2 implementations
    public abstract double toRadian();
}

//create subclass of Angle
class Degree extends Angle{
    //constructor that creates Degree object and assign the value measurement
    public Degree(double value) { super(value); }

    //abstract methods with calculations (body)
    public double toDegree() { return value; }
    public double toRadian() { return (value * pi / 180); }
}

//create subclass of Angle
class Radian extends Angle{
    //constructor that creates Degree object and assign the value measurement
    public Radian(double value) { super(value); }

    //abstract methods with calculations (body)
    public double toDegree() { return(value * 180 / pi); }
    public double toRadian() { return value; }
}
```

Temperature

TempConverter.java | Temperature.java

src > TempConverter.java > TempConverter > main(String[])

```java
import java.text.DecimalFormat;
import java.text.NumberFormat;
import java.util.*;

public class TempConverter{
    Run | Debug
    public static void main(String[] args) throws Exception {
        Scanner input = new Scanner(System.in);
        NumberFormat formatter = new DecimalFormat(pattern: "#0.0");

        System.out.print(s: "Enter temperature value: ");
        double tempValue = input.nextDouble();

        System.out.print(s: "Enter temperature unit (C/F/K): ");
        String fromUnit = input.next();


        System.out.print(s: "Enter unit to convert to (C/F/K): ");
        String toUnit = input.next();
        input.close();


        // Creates an instance of the class Temperature
        Temperature initialTemp;
        if (fromUnit.equalsIgnoreCase(anotherString: "C")){
            initialTemp = new Celcius(tempValue); // Instantiates a Celcius object if user entered "C"
        }else if (fromUnit.equalsIgnoreCase(anotherString: "F")){
            initialTemp = new Fahrenheit(tempValue); // Instantiates a Fahrenheit object if user entered "F"
        }else if (fromUnit.equalsIgnoreCase(anotherString: "K")){
            initialTemp = new Kelvin(tempValue); // Instantiates a Kelvin object if user entered "K"
        }else{
            System.out.print(s: "Invalid temperature unit!");
            return;
        }

```

TempConverter.java | Temperature.java

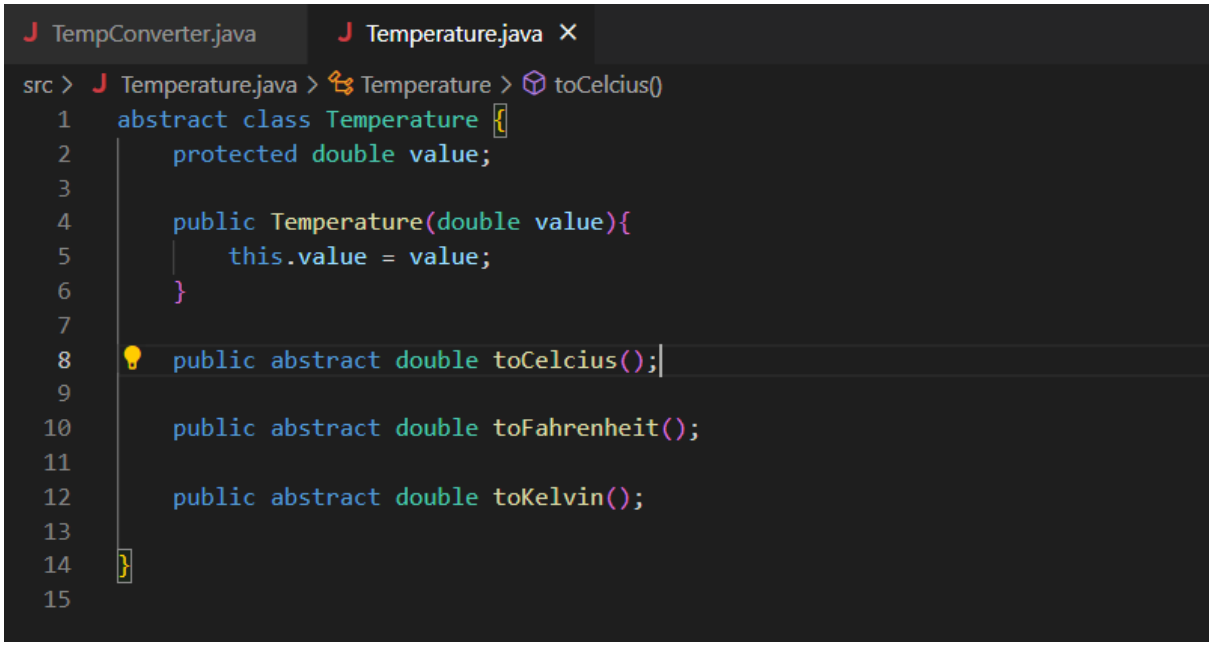
src > TempConverter.java > TempConverter > main(String[])

```
        // Declare a new variable to store converted unit
        double convertedTemp;
        if (toUnit.equalsIgnoreCase(anotherString: "C")){
            convertedTemp = initialTemp.toCelcius();
            System.out.println("Temperature in Celcius: " + formatter.format(convertedTemp));
        }else if (toUnit.equalsIgnoreCase(anotherString: "F")){
            convertedTemp = initialTemp.toFahrenheit();
            System.out.println("Temperature in Fahrenheit: " + formatter.format(convertedTemp));
        }else if (toUnit.equalsIgnoreCase(anotherString: "K")){
            convertedTemp = initialTemp.toKelvin();
            System.out.println("Temperature in Kelvin: " + formatter.format(convertedTemp));
        }else{
            System.out.print(s: "Invalid temperature unit!");
            return;
        }

    }
}

// Inherited classes
class Celcius extends Temperature{
    public Celcius(double value){
        super(value);
    }

    public double toCelcius(){
        return value;
    }

    public double toFahrenheit(){
        return (value * 9 / 5) + 32;
    }

    public double toKelvin(){
        return (value + 273.15);
    }

}
```

TempConverter.java | Temperature.java

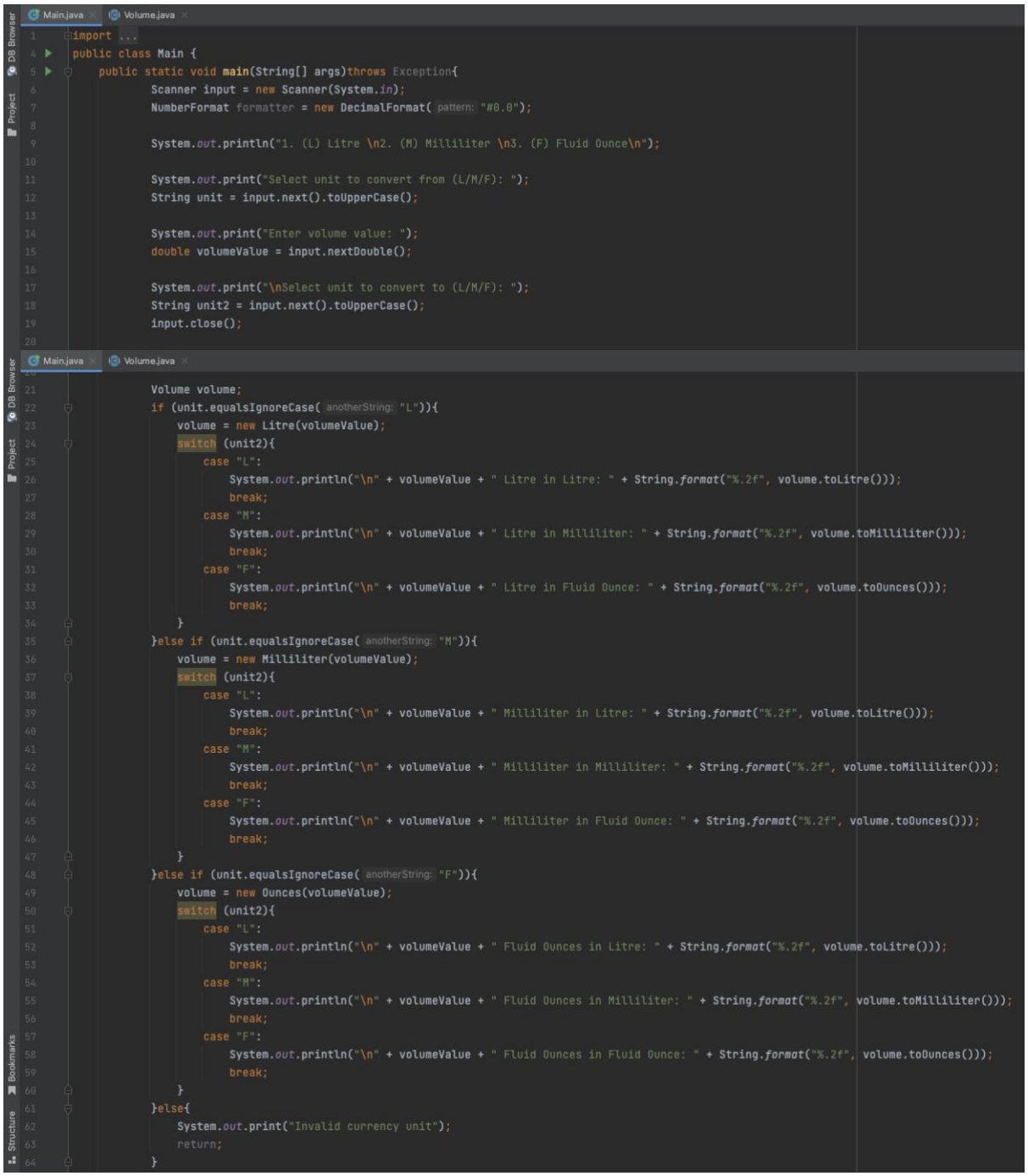
src > TempConverter.java > TempConverter > main(String[])

```java
}

class Fahrenheit extends Temperature{
    public Fahrenheit(double value){
        super(value);
    }

    public double toCelcius(){
        return (value - 32) * 5 / 9;
    }

    public double toFahrenheit(){
        return value;
    }

    public double toKelvin(){
        return toCelcius() + 273.15;
    }
}

class Kelvin extends Temperature{
    public Kelvin(double value){
        super(value);
    }

    public double toCelcius(){
        return (value - 273.15);
    }

    public double toFahrenheit(){
        return (value - 273.15) * 9 / 5 + 32;
    }

    public double toKelvin(){
        return value;
    }
}
```


TempConverter.java
Temperature.java
src > Temperature.java > Temperature > toCelcius()
abstract class Temperature {
protected double value;
public Temperature(double value){
this.value = value;
}
public abstract double toCelcius();
public abstract double toFahrenheit();
public abstract double toKelvin();
}

Volume

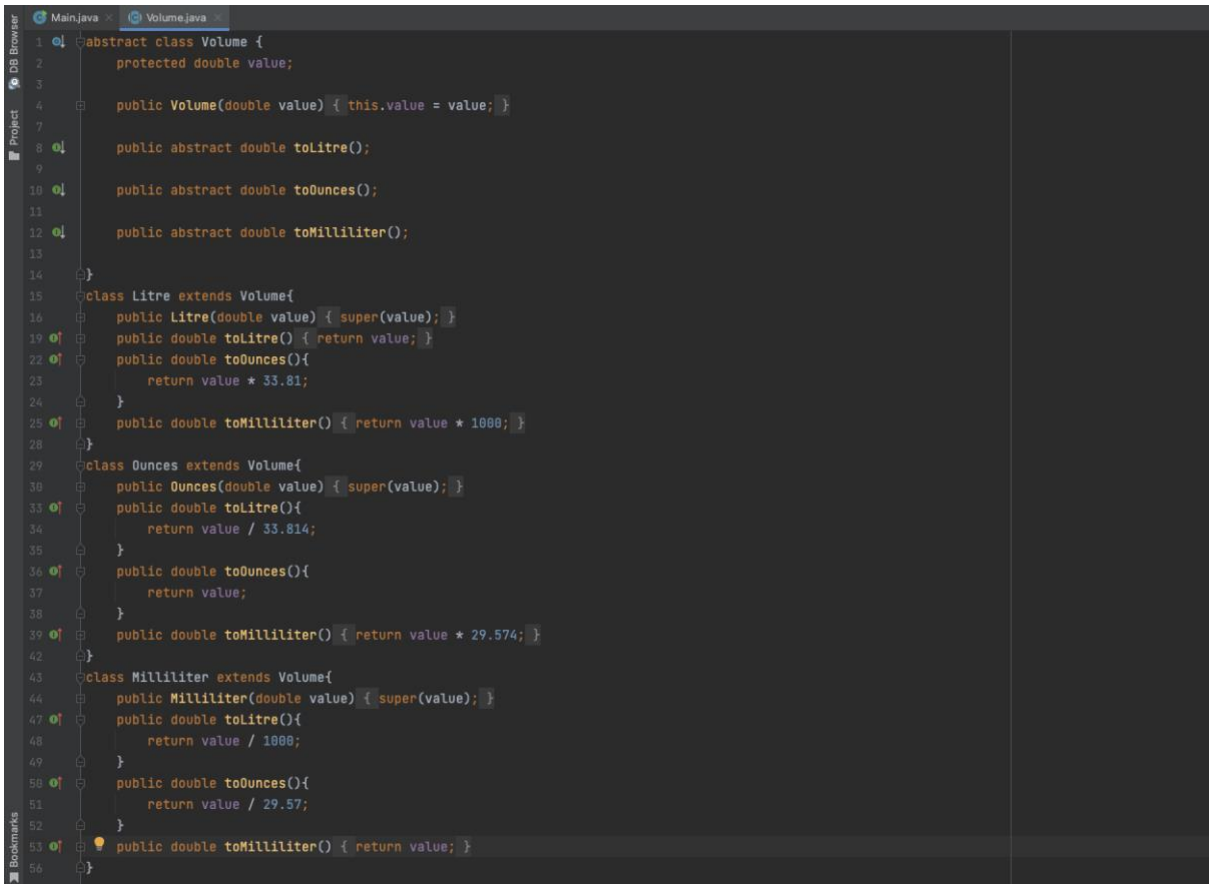
Main class



```java
import ...
public class Main {
    public static void main(String[] args)throws Exception{
            Scanner input = new Scanner(System.in);
            NumberFormat formatter = new DecimalFormat( pattern: "#0.0");

            System.out.println("1. (L) Litre \n2. (M) Milliliter \n3. (F) Fluid Ounce\n");

            System.out.print("Select unit to convert from (L/M/F): ");
            String unit = input.next().toUpperCase();

            System.out.print("Enter volume value: ");
            double volumeValue = input.nextDouble();

            System.out.print("\nSelect unit to convert to (L/M/F): ");
            String unit2 = input.next().toUpperCase();
            input.close();
```

```java
            Volume volume;
            if (unit.equalsIgnoreCase( anotherString: "L")){
                volume = new Litre(volumeValue);
                switch (unit2){
                    case "L":
                        System.out.println("\n" + volumeValue + " Litre in Litre: " + String.format("%.2f", volume.toLitre()));
                        break;
                    case "M":
                        System.out.println("\n" + volumeValue + " Litre in Milliliter: " + String.format("%.2f", volume.toMilliliter()));
                        break;
                    case "F":
                        System.out.println("\n" + volumeValue + " Litre in Fluid Ounce: " + String.format("%.2f", volume.toOunces()));
                        break;
                }
            }else if (unit.equalsIgnoreCase( anotherString: "M")){
                volume = new Milliliter(volumeValue);
                switch (unit2){
                    case "L":
                        System.out.println("\n" + volumeValue + " Milliliter in Litre: " + String.format("%.2f", volume.toLitre()));
                        break;
                    case "M":
                        System.out.println("\n" + volumeValue + " Milliliter in Milliliter: " + String.format("%.2f", volume.toMilliliter()));
                        break;
                    case "F":
                        System.out.println("\n" + volumeValue + " Milliliter in Fluid Ounce: " + String.format("%.2f", volume.toOunces()));
                        break;
                }
            }else if (unit.equalsIgnoreCase( anotherString: "F")){
                volume = new Ounces(volumeValue);
                switch (unit2){
                    case "L":
                        System.out.println("\n" + volumeValue + " Fluid Ounces in Litre: " + String.format("%.2f", volume.toLitre()));
                        break;
                    case "M":
                        System.out.println("\n" + volumeValue + " Fluid Ounces in Milliliter: " + String.format("%.2f", volume.toMilliliter()));
                        break;
                    case "F":
                        System.out.println("\n" + volumeValue + " Fluid Ounces in Fluid Ounce: " + String.format("%.2f", volume.toOunces()));
                        break;
                }
            }else{
                System.out.print("Invalid currency unit");
                return;
            }
```

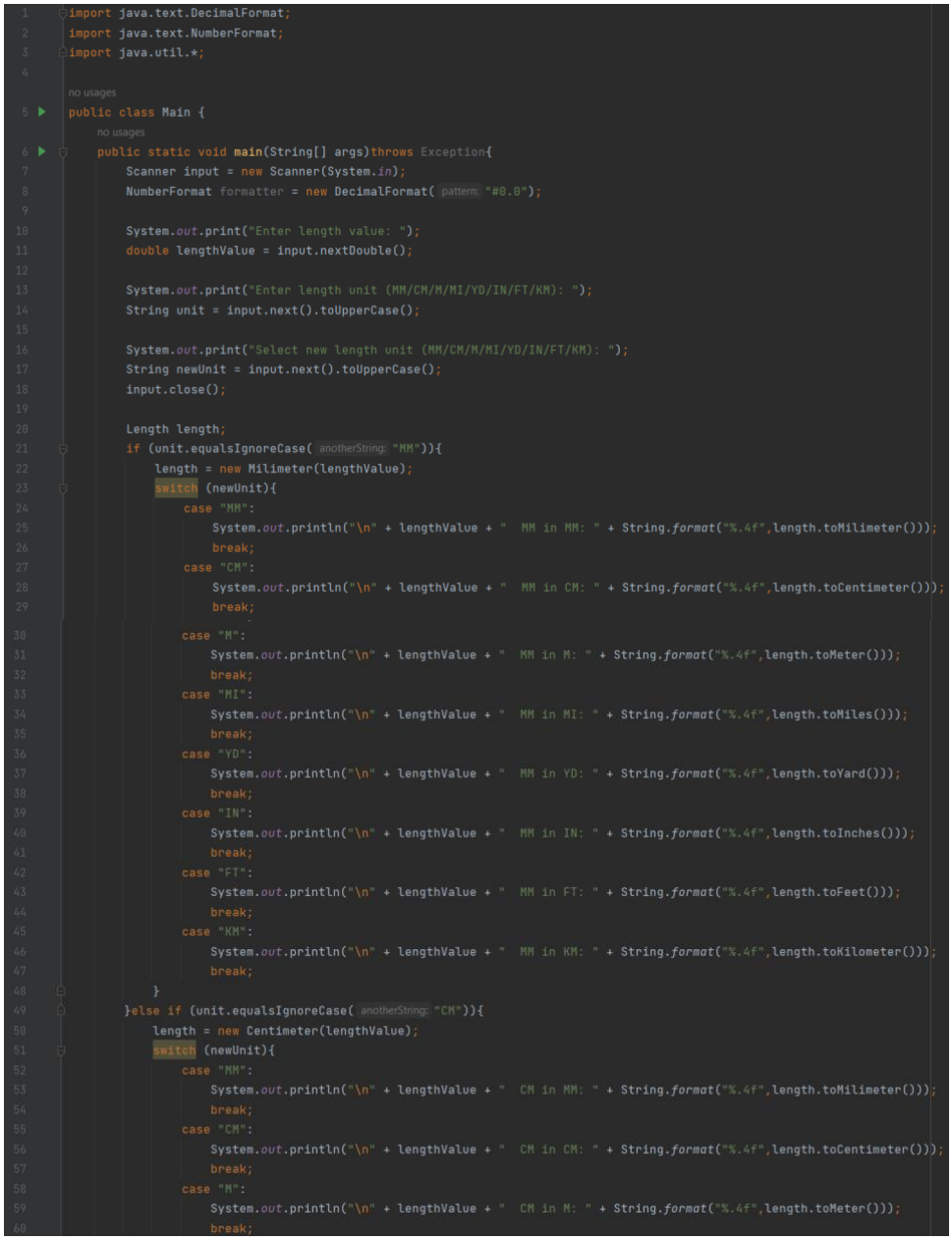
Volume Class




```java
abstract class Volume {
    protected double value;

    public Volume(double value) { this.value = value; }

    public abstract double toLitre();

    public abstract double toOunces();

    public abstract double toMilliliter();

}
class Litre extends Volume{
    public Litre(double value) { super(value); }
    public double toLitre() { return value; }
    public double toOunces(){
        return value * 33.81;
    }
    public double toMilliliter() { return value * 1000; }
}
class Ounces extends Volume{
    public Ounces(double value) { super(value); }
    public double toLitre(){
        return value / 33.814;
    }
    public double toOunces(){
        return value;
    }
    public double toMilliliter() { return value * 29.574; }
}
class Milliliter extends Volume{
    public Milliliter(double value) { super(value); }
    public double toLitre(){
        return value / 1000;
    }
    public double toOunces(){
        return value / 29.57;
    }
    public double toMilliliter() { return value; }
}
```

Length

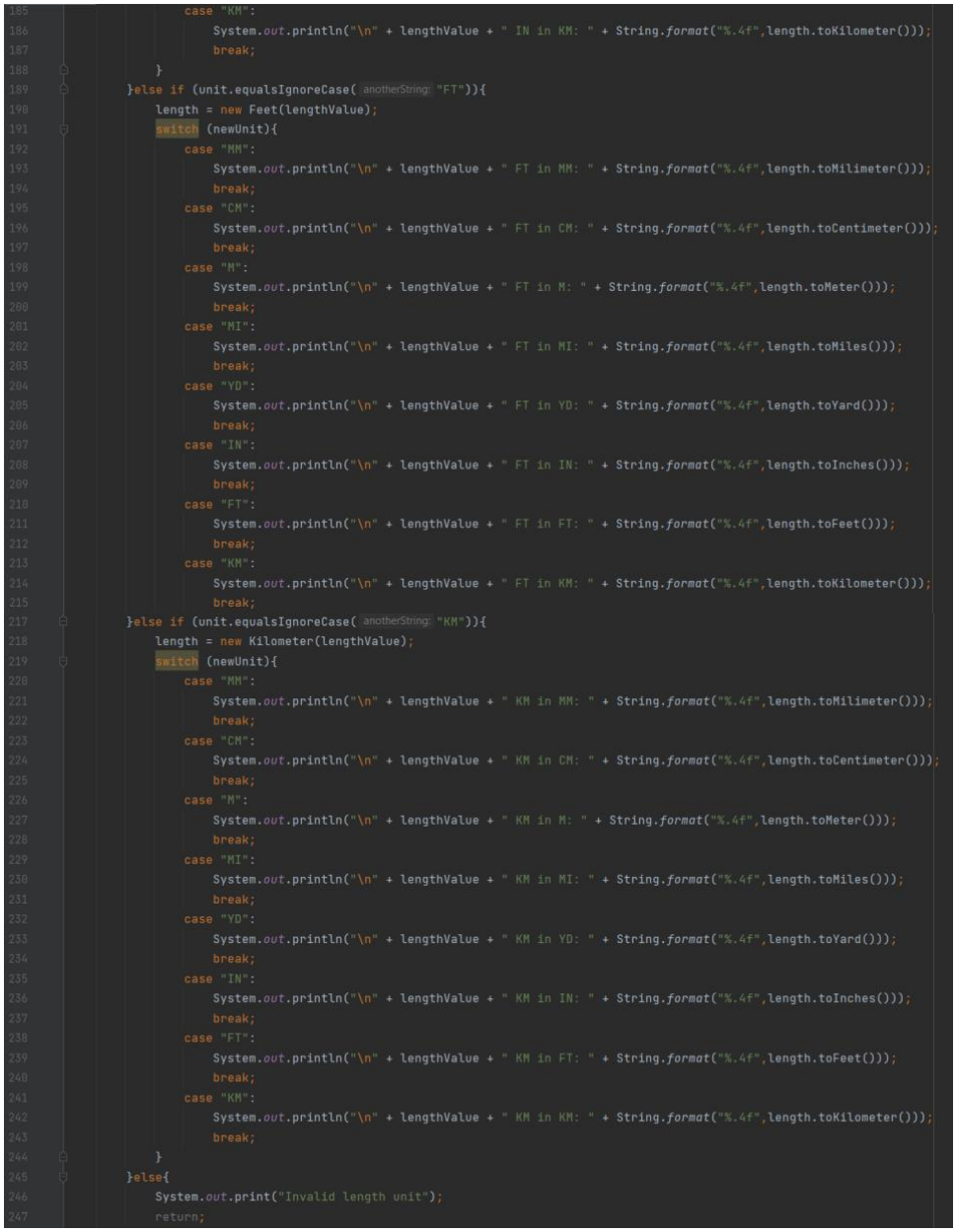
Main Class



```java
import java.text.DecimalFormat;
import java.text.NumberFormat;
import java.util.*;

no usages
public class Main {
    no usages
    public static void main(String[] args)throws Exception{
        Scanner input = new Scanner(System.in);
        NumberFormat formatter = new DecimalFormat( pattern: "#0.0");

        System.out.print("Enter length value: ");
        double lengthValue = input.nextDouble();

        System.out.print("Enter length unit (MM/CM/M/MI/YD/IN/FT/KM): ");
        String unit = input.next().toUpperCase();

        System.out.print("Select new length unit (MM/CM/M/MI/YD/IN/FT/KM): ");
        String newUnit = input.next().toUpperCase();
        input.close();

        Length length;
        if (unit.equalsIgnoreCase( anotherString: "MM")){
            length = new Milimeter(lengthValue);
            switch (newUnit){
                case "MM":
                    System.out.println("\n" + lengthValue + "  MM in MM: " + String.format("%.4f",length.toMilimeter()));
                    break;
                case "CM":
                    System.out.println("\n" + lengthValue + "  MM in CM: " + String.format("%.4f",length.toCentimeter()));
                    break;
                case "M":
                    System.out.println("\n" + lengthValue + "  MM in M: " + String.format("%.4f",length.toMeter()));
                    break;
                case "MI":
                    System.out.println("\n" + lengthValue + "  MM in MI: " + String.format("%.4f",length.toMiles()));
                    break;
                case "YD":
                    System.out.println("\n" + lengthValue + "  MM in YD: " + String.format("%.4f",length.toYard()));
                    break;
                case "IN":
                    System.out.println("\n" + lengthValue + "  MM in IN: " + String.format("%.4f",length.toInches()));
                    break;
                case "FT":
                    System.out.println("\n" + lengthValue + "  MM in FT: " + String.format("%.4f",length.toFeet()));
                    break;
                case "KM":
                    System.out.println("\n" + lengthValue + "  MM in KM: " + String.format("%.4f",length.toKilometer()));
                    break;
            }
        }else if (unit.equalsIgnoreCase( anotherString: "CM")){
            length = new Centimeter(lengthValue);
            switch (newUnit){
                case "MM":
                    System.out.println("\n" + lengthValue + "  CM in MM: " + String.format("%.4f",length.toMilimeter()));
                    break;
                case "CM":
                    System.out.println("\n" + lengthValue + "  CM in CM: " + String.format("%.4f",length.toCentimeter()));
                    break;
                case "M":
                    System.out.println("\n" + lengthValue + "  CM in M: " + String.format("%.4f",length.toMeter()));
                    break;
```

```
                case "MI":
                    System.out.println("\n" + lengthValue + "  CM in MI: " + String.format("%.4f",length.toMiles()));
                    break;
                case "YD":
                    System.out.println("\n" + lengthValue + "  CM in YD: " + String.format("%.4f",length.toYard()));
                    break;
                case "IN":
                    System.out.println("\n" + lengthValue + "  CM in IN: " + String.format("%.4f",length.toInches()));
                    break;
                case "FT":
                    System.out.println("\n" + lengthValue + "  CM in FT: " + String.format("%.4f",length.toFeet()));
                    break;
                case "KM":
                    System.out.println("\n" + lengthValue + "  CM in KM: " + String.format("%.4f",length.toKilometer()));
                    break;
            }
        }else if (unit.equalsIgnoreCase( anotherString: "M")){
            length = new Meter(lengthValue);
            switch (newUnit){
                case "MM":
                    System.out.println("\n" + lengthValue + "  M in MM: " + String.format("%.4f",length.toMilimeter()));
                    break;
                case "CM":
                    System.out.println("\n" + lengthValue + "  M in CM: " + String.format("%.4f",length.toCentimeter()));
                    break;
                case "M":
                    System.out.println("\n" + lengthValue + "  M in M: " + String.format("%.4f",length.toMeter()));
                    break;
                case "MI":
                    System.out.println("\n" + lengthValue + "  M in MI: " + String.format("%.4f",length.toMiles()));
                    break;
                case "YD":
                    System.out.println("\n" + lengthValue + "  M in YD: " + String.format("%.4f",length.toYard()));
                    break;
                case "IN":
                    System.out.println("\n" + lengthValue + "  M in IN: " + String.format("%.4f",length.toInches()));
                    break;
                case "FT":
                    System.out.println("\n" + lengthValue + "  M in FT: " + String.format("%.4f",length.toFeet()));
                    break;
                case "KM":
                    System.out.println("\n" + lengthValue + "  M in KM: " + String.format("%.4f",length.toKilometer()));
                    break;
            }
        }else if (unit.equalsIgnoreCase( anotherString: "MI")){
            length = new Miles(lengthValue);
            switch (newUnit){
                case "MM":
                    System.out.println("\n" + lengthValue + "  MI in MM: " + String.format("%.4f",length.toMilimeter()));
                    break;
                case "CM":
                    System.out.println("\n" + lengthValue + "  MI in CM: " + String.format("%.4f",length.toCentimeter()));
                    break;
                case "M":
                    System.out.println("\n" + lengthValue + "  MI in M: " + String.format("%.4f",length.toMeter()));
                    break;
                case "MI":
                    System.out.println("\n" + lengthValue + "  MI in MI: " + String.format("%.4f",length.toMiles()));
                    break;
                case "YD":
                    System.out.println("\n" + lengthValue + "  MI in YD: " + String.format("%.4f",length.toYard()));
                    break;
```

```
                case "IN":
                    System.out.println("\n" + lengthValue + "  MI in IN: " + String.format("%.4f",length.toInches()));
                    break;
                case "FT":
                    System.out.println("\n" + lengthValue + "  MI in FT: " + String.format("%.4f",length.toFeet()));
                    break;
                case "KM":
                    System.out.println("\n" + lengthValue + "  MI in KM: " + String.format("%.4f",length.toKilometer()));
                    break;
            }
        }else if (unit.equalsIgnoreCase( anotherString: "YD")){
            length = new Yard(lengthValue);
            switch (newUnit){
                case "MM":
                    System.out.println("\n" + lengthValue + "  YD in MM: " + String.format("%.4f",length.toMilimeter()));
                    break;
                case "CM":
                    System.out.println("\n" + lengthValue + "  YD in CM: " + String.format("%.4f",length.toCentimeter()));
                    break;
                case "M":
                    System.out.println("\n" + lengthValue + "  YD in M: " + String.format("%.4f",length.toMeter()));
                    break;
                case "MI":
                    System.out.println("\n" + lengthValue + "  YD in MI: " + String.format("%.4f",length.toMiles()));
                    break;
                case "YD":
                    System.out.println("\n" + lengthValue + "  YD in YD: " + String.format("%.4f",length.toYard()));
                    break;
                case "IN":
                    System.out.println("\n" + lengthValue + "  YD in IN: " + String.format("%.4f",length.toInches()));
                    break;
                case "FT":
                    System.out.println("\n" + lengthValue + "  YD in FT: " + String.format("%.4f",length.toFeet()));
                    break;
                case "KM":
                    System.out.println("\n" + lengthValue + "  YD in KM: " + String.format("%.4f",length.toKilometer()));
                    break;
            }
        }else if (unit.equalsIgnoreCase( anotherString: "IN")){
            length = new Inches(lengthValue);
            switch (newUnit){
                case "MM":
                    System.out.println("\n" + lengthValue + "  IN in MM: " + String.format("%.4f",length.toMilimeter()));
                    break;
                case "CM":
                    System.out.println("\n" + lengthValue + "  IN in CM: " + String.format("%.4f",length.toCentimeter()));
                    break;
                case "M":
                    System.out.println("\n" + lengthValue + " IN in M: " + String.format("%.4f",length.toMeter()));
                    break;
                case "MI":
                    System.out.println("\n" + lengthValue + " IN in MI: " + String.format("%.4f",length.toMiles()));
                    break;
                case "YD":
                    System.out.println("\n" + lengthValue + " IN in YD: " + String.format("%.4f",length.toYard()));
                    break;
                case "IN":
                    System.out.println("\n" + lengthValue + " IN in IN: " + String.format("%.4f",length.toInches()));
                    break;
                case "FT":
                    System.out.println("\n" + lengthValue + " IN in FT: " + String.format("%.4f",length.toFeet()));
                    break;
```



```
                case "KM":
                    System.out.println("\n" + lengthValue + " IN in KM: " + String.format("%.4f",length.toKilometer()));
                    break;
            }
        }else if (unit.equalsIgnoreCase( anotherString: "FT")){
            length = new Feet(lengthValue);
            switch (newUnit){
                case "MM":
                    System.out.println("\n" + lengthValue + " FT in MM: " + String.format("%.4f",length.toMilimeter()));
                    break;
                case "CM":
                    System.out.println("\n" + lengthValue + " FT in CM: " + String.format("%.4f",length.toCentimeter()));
                    break;
                case "M":
                    System.out.println("\n" + lengthValue + " FT in M: " + String.format("%.4f",length.toMeter()));
                    break;
                case "MI":
                    System.out.println("\n" + lengthValue + " FT in MI: " + String.format("%.4f",length.toMiles()));
                    break;
                case "YD":
                    System.out.println("\n" + lengthValue + " FT in YD: " + String.format("%.4f",length.toYard()));
                    break;
                case "IN":
                    System.out.println("\n" + lengthValue + " FT in IN: " + String.format("%.4f",length.toInches()));
                    break;
                case "FT":
                    System.out.println("\n" + lengthValue + " FT in FT: " + String.format("%.4f",length.toFeet()));
                    break;
                case "KM":
                    System.out.println("\n" + lengthValue + " FT in KM: " + String.format("%.4f",length.toKilometer()));
                    break;
        }else if (unit.equalsIgnoreCase( anotherString: "KM")){
            length = new Kilometer(lengthValue);
            switch (newUnit){
                case "MM":
                    System.out.println("\n" + lengthValue + " KM in MM: " + String.format("%.4f",length.toMilimeter()));
                    break;
                case "CM":
                    System.out.println("\n" + lengthValue + " KM in CM: " + String.format("%.4f",length.toCentimeter()));
                    break;
                case "M":
                    System.out.println("\n" + lengthValue + " KM in M: " + String.format("%.4f",length.toMeter()));
                    break;
                case "MI":
                    System.out.println("\n" + lengthValue + " KM in MI: " + String.format("%.4f",length.toMiles()));
                    break;
                case "YD":
                    System.out.println("\n" + lengthValue + " KM in YD: " + String.format("%.4f",length.toYard()));
                    break;
                case "IN":
                    System.out.println("\n" + lengthValue + " KM in IN: " + String.format("%.4f",length.toInches()));
                    break;
                case "FT":
                    System.out.println("\n" + lengthValue + " KM in FT: " + String.format("%.4f",length.toFeet()));
                    break;
                case "KM":
                    System.out.println("\n" + lengthValue + " KM in KM: " + String.format("%.4f",length.toKilometer()));
                    break;
            }
        }else{
            System.out.print("Invalid length unit");
            return;
```

Length Class

```java
abstract class Length {
    65 usages
    protected double value;

    8 usages
    public Length(double value) {this.value = value;}

    8 usages  8 implementations
    public abstract double toMilimeter();
    8 usages  8 implementations
    public abstract double toCentimeter();
    8 usages  8 implementations
    public abstract double toMeter();
    8 usages  8 implementations
    public abstract double toMiles();
    8 usages  8 implementations
    public abstract double toYard();
    8 usages  8 implementations
    public abstract double toInches();
    8 usages  8 implementations
    public abstract double toFeet();
    8 usages  8 implementations
    public abstract double toKilometer();

no usages
class Milimeter extends Length{
    no usages
    public Milimeter(double value) { super(value); }
    8 usages
    public double toMilimeter(){return value;}
```


public double toCentimeter() { return value / 10.0; }
8 usages
public double toMeter() {return value / 1000.0;}
8 usages
public double toMiles() { return value / 1609344.0; }
8 usages
public double toYard() { return value / 914.4; }
8 usages
public double toInches() { return value / 25.4; }
8 usages
public double toFeet(){return value / 304.8;}
8 usages
public double toKilometer() { return value / 1000000.0; }
}
no usages
class Centimeter extends Length{
no usages
public Centimeter(double value) { super(value); }
8 usages
public double toMilimeter() { return value * 10.0; }
8 usages
public double toCentimeter() { return value; }
8 usages
public double toMeter() {return value / 100.0;}
8 usages
public double toMiles(){return value / 160934.4;}

```
54      public double toYard(){return value / 91.44;}
        8 usages
55      public double toInches(){return value / 2.54;}
        8 usages
56      public double toFeet() { return value / 30.48; }
        8 usages
59      public double toKilometer() { return value / 100000.0; }
62  }
63
    no usages
64  class Meter extends Length {
        no usages
65      public Meter(double value) {super(value);}
66
        8 usages
67      public double toMilimeter() {return value * 1000.0;}
68
        8 usages
69      public double toCentimeter() {return value * 100.0;}
70
        8 usages
71      public double toMeter() {return value;}
72
        8 usages
73      public double toMiles() {return value / 1609.0;}
74
        8 usages
75      public double toYard() {return value * 1.094;}
76
```


8 usages
77 public double toInches() {return value * 39.37;}
78
8 usages
79 public double toFeet() {return value * 3.281;}
80
8 usages
81 public double toKilometer() {return value / 1000.0;}
82 }
no usages
83 class Miles extends Length {
no usages
84 public Miles(double value) { super(value); }
8 usages
87 public double toMilimeter() { return value * 1609344; }
8 usages
90 public double toCentimeter() { return value * 160934.4; }
93
8 usages
94 public double toMeter() { return value * 1609.344; }
97
8 usages
98 public double toMiles() { return value; }
101
8 usages
102 public double toYard() {return value * 1760;}
103
8 usages
104 public double toInches() { return value * 63360; }
107


public double toFeet() { return value * 5280; }
8 usages
public double toKilometer() { return value * 1.609344; }
}
no usages
class Yard extends Length {
no usages
public Yard(double value) { super(value); }
8 usages
public double toMilimeter() { return value * 914.4; }
8 usages
public double toCentimeter() { return value * 91.44; }
8 usages
public double toMeter() {
return value * 0.9144;
}
8 usages
public double toMiles() { return value / 1760; }
8 usages
public double toYard() { return value; }
8 usages
public double toInches() { return value * 36; }


public double toFeet() { return value * 3; }
8 usages
public double toKilometer() { return value * 0.0009144; }
}
no usages
class Inches extends Length {
no usages
public Inches(double value) { super(value); }
8 usages
public double toMilimeter() { return value * 25.4; }
8 usages
public double toCentimeter() { return value * 2.54; }
8 usages
public double toMeter() { return value * 0.0254; }
8 usages
public double toMiles() { return value / 63360; }
8 usages
public double toYard() { return value / 36; }
8 usages
public double toInches() { return value; }
8 usages
public double toFeet() { return value / 12; }

```
186     public double toKilometer() { return value * 0.0000254; }
189 }
190
        no usages
191 class Feet extends Length {
            no usages
192     public Feet(double value) { super(value); }
195
            8 usages
196     public double toMilimeter() { return value * 304.8; }
199
            8 usages
200     public double toCentimeter() {return value * 30.48;}
201
            8 usages
202     public double toMeter() { return value * 0.3048; }
205
            8 usages
206     public double toMiles() { return value / 5280; }
209
            8 usages
210     public double toYard() { return value / 3; }
213
            8 usages
214     public double toInches() { return value * 12; }
217
            8 usages
218     public double toFeet() { return value; }
221
            8 usages
222     public double toKilometer() { return value / 3280.84; }
```


class Kilometer extends Length {
no usages
public Kilometer(double value) { super(value); }
8 usages
public double toMilimeter() { return value * 1000000; }
8 usages
public double toCentimeter() { return value * 100000; }
8 usages
public double toMeter() { return value * 1000; }
8 usages
public double toMiles() { return value / 1.609344; }
8 usages
public double toYard() { return value * 1093.61; }
8 usages
public double toInches() { return value * 39370.1; }
8 usages
public double toFeet() { return value * 3280.84; }
8 usages
public double toKilometer() { return value; }
}

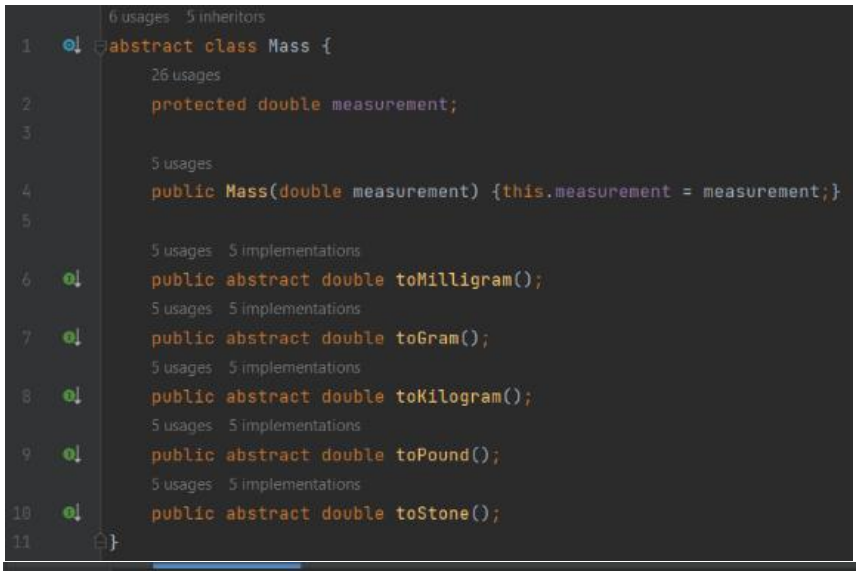
Mass

Main Class

```java
import java.util.*;
no usages
public class Main {
    no usages
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);

        System.out.print("Enter mass value: ");
        double massValue = input.nextDouble();

        System.out.print("Enter mass unit (MG/G/KG/LBS/ST): ");
        String unit = input.next().toUpperCase();

        System.out.print("Select new mass unit (MG/G/KG/LBS/ST): ");
        String newUnit = input.next().toUpperCase();
        input.close();

        Mass mass;
        if (unit.equalsIgnoreCase( anotherString: "MG")){
            mass = new Milligram(massValue);
            switch (newUnit){
                case "MG":
                    System.out.println("\n" + massValue + " MG in MG: " + String.format("%.4f",mass.toMilligram()));
                    break;
                case "G":
                    System.out.println("\n" + massValue + " MG in G: " + String.format("%.4f",mass.toGram()));
                    break;
                case "KG":
                    System.out.println("\n" + massValue + " MG in KG: " + String.format("%.4f",mass.toKilogram()));
                    break;
                case "LBS":
                    System.out.println("\n" + massValue + " MG in LBS: " + String.format("%.4f",mass.toPound()));
                    break;
                case "ST":
                    System.out.println("\n" + massValue + " MG in ST: " + String.format("%.4f",mass.toStone()));
                    break;
            }
        }else if (unit.equalsIgnoreCase( anotherString: "G")){
            mass = new Gram(massValue);
            switch (newUnit){
                case "MG":
                    System.out.println("\n" + massValue + " G in MG: " + String.format("%.4f",mass.toMilligram()));
                    break;
                case "G":
                    System.out.println("\n" + massValue + " G in G: " + String.format("%.4f",mass.toGram()));
                    break;
                case "KG":
                    System.out.println("\n" + massValue + " G in KG: " + String.format("%.4f",mass.toKilogram()));
                    break;
                case "LBS":
                    System.out.println("\n" + massValue + " G in LBS: " + String.format("%.4f",mass.toPound()));
                    break;
                case "ST":
                    System.out.println("\n" + massValue + " G in ST: " + String.format("%.4f",mass.toStone()));
                    break;
            }
```

```
        }else if (unit.equalsIgnoreCase( anotherString: "KG")){
            mass = new Kilogram(massValue);
            switch (newUnit){
                case "MG":
                    System.out.println("\n" + massValue + " KG in MG: " + String.format("%.4f",mass.toMilligram()));
                    break;
                case "G":
                    System.out.println("\n" + massValue + " KG in G: " + String.format("%.4f",mass.toGram()));
                    break;
                case "KG":
                    System.out.println("\n" + massValue + " KG in KG: " + String.format("%.4f",mass.toKilogram()));
                    break;
                case "LBS":
                    System.out.println("\n" + massValue + " KG in LBS: " + String.format("%.4f",mass.toPound()));
                    break;
                case "ST":
                    System.out.println("\n" + massValue + " KG in ST: " + String.format("%.4f",mass.toStone()));
                    break;
            }
        }else if (unit.equalsIgnoreCase( anotherString: "LBS")){
            mass = new Pound(massValue);
            switch (newUnit){
                case "MG":
                    System.out.println("\n" + massValue + " LBS in MG: " + String.format("%.4f",mass.toMilligram()));
                    break;
                case "G":
                    System.out.println("\n" + massValue + " LBS in G: " + String.format("%.4f",mass.toGram()));
                    break;
                case "KG":
                    System.out.println("\n" + massValue + " LBS in KG: " + String.format("%.4f",mass.toKilogram()));
                    break;
                case "LBS":
                    System.out.println("\n" + massValue + " LBS in LBS: " + String.format("%.4f",mass.toPound()));
                    break;
                case "ST":
                    System.out.println("\n" + massValue + " LBS in ST: " + String.format("%.4f",mass.toStone()));
                    break;
            }
        }else if (unit.equalsIgnoreCase( anotherString: "ST")){
            mass = new Stone(massValue);
            switch (newUnit){
                case "MG":
                    System.out.println("\n" + massValue + " ST in MG: " + String.format("%.4f",mass.toMilligram()));
                    break;
                case "G":
                    System.out.println("\n" + massValue + " ST in G: " + String.format("%.4f",mass.toGram()));
                    break;
                case "KG":
                    System.out.println("\n" + massValue + " ST in KG: " + String.format("%.4f",mass.toKilogram()));
                    break;
                case "LBS":
                    System.out.println("\n" + massValue + " ST in LBS: " + String.format("%.4f",mass.toPound()));
                    break;
                case "ST":
                    System.out.println("\n" + massValue + " ST in ST: " + String.format("%.4f",mass.toStone()));
                    break;
            }
        }
    }
}
```

Mass Class



```
6 usages  5 inheritors
abstract class Mass {
    26 usages
    protected double measurement;

    5 usages
    public Mass(double measurement) {this.measurement = measurement;}

    5 usages  5 implementations
    public abstract double toMilligram();
    5 usages  5 implementations
    public abstract double toGram();
    5 usages  5 implementations
    public abstract double toKilogram();
    5 usages  5 implementations
    public abstract double toPound();
    5 usages  5 implementations
    public abstract double toStone();
}
```

```
1 usage
class Milligram extends Mass {
    1 usage
    public Milligram(double measurement) {super(measurement);}
    5 usages
    public double toMilligram() {return measurement;}
    5 usages
    public double toGram() {return measurement / 1000;}
    5 usages
    public double toKilogram() {return measurement / 1000000;}
    5 usages
    public double toPound() {return measurement / 453600;}
    5 usages
    public double toStone() {return measurement / 6350000;}
}

1 usage
class Gram extends Mass {
    1 usage
    public Gram(double measurement) {super(measurement);}
    5 usages
    public double toMilligram() {return measurement * 1000;}
    5 usages
    public double toGram() {return measurement;}
    5 usages
    public double toKilogram() {return measurement / 1000;}
    5 usages
    public double toPound() {return measurement / 453.6;}
    5 usages
    public double toStone() {return measurement / 6350;}
}
```

```java
class Kilogram extends Mass {
    1 usage
    public Kilogram(double measurement) {super(measurement);}
    5 usages
    public double toMilligram() {return measurement * 1000000;}
    5 usages
    public double toGram() {return measurement * 1000;}
    5 usages
    public double toKilogram() {return measurement;}
    5 usages
    public double toPound() {return measurement * 2.205;}
    5 usages
    public double toStone() {return measurement / 6.35;}
}

1 usage
class Pound extends Mass {
    1 usage
    public Pound(double measurement) {super(measurement);}
    5 usages
    public double toMilligram() {return measurement * 453600;}
    5 usages
    public double toGram() {return measurement * 453.6;}
    5 usages
    public double toKilogram() {return measurement / 2.265;}
    5 usages
    public double toPound() {return measurement;}
    5 usages
    public double toStone() {return measurement / 14;}
}
```

```java
class Stone extends Mass {
    1 usage
    public Stone(double measurement) {super(measurement);}
    5 usages
    public double toMilligram() {return measurement * 6350000;}
    5 usages
    public double toGram() {return measurement * 6350;}
    5 usages
    public double toKilogram() {return measurement * 6.35;}
    5 usages
    public double toPound() {return measurement * 14;}
    5 usages
    public double toStone() {return measurement;}
}
```

Time

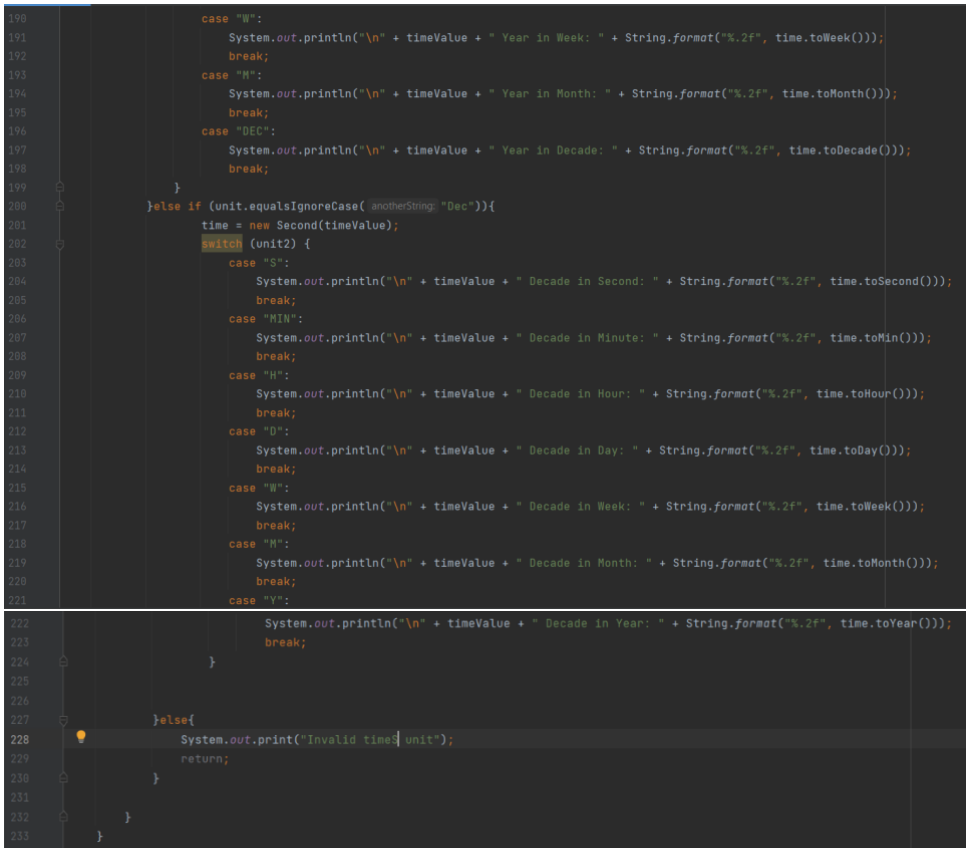
Main Class

Main.java Time.java

```java
import java.text.DecimalFormat;
import java.text.NumberFormat;
import java.util.Scanner;

no usages
public class Main {
    no usages
    public static void main(String[] args)throws Exception{
            Scanner input = new Scanner(System.in);
            NumberFormat formatter = new DecimalFormat( pattern: "#0.0");

            System.out.print("Enter Time value: ");
            double timeValue = input.nextDouble();

            System.out.println("1. (S) Second \n2. (Min) Minute \n3. (H) Hour \n4. (D) Day\n5. (W) Week\n6. (M) Month \n7. (Y) Year \n8. (Dec) Decade");

            System.out.print("Select unit(S/Min/H/D/W/M/Y/Dec) (for old Time): ");
            String unit = input.next().toUpperCase();

            System.out.print("Select unit(S/Min/H/H/D/W/M/Dec) (for new Time): ");
            String unit2 = input.next().toUpperCase();
            input.close();

            Time time;
            if (unit.equalsIgnoreCase( anotherString: "S")){
                time = new Second(timeValue);
                switch (unit2){
                    case "MIN":
                        System.out.println("\n" + timeValue + " Second in Minute: " + String.format("%.2f", time.toMin()));
                        break;
                    case "H":
                        System.out.println("\n" + timeValue + " Second in Hour: " + String.format("%.2f", time.toHour()));
                        break;
                    case "D":
                        System.out.println("\n" + timeValue + " Second in Day: " + String.format("%.2f", time.toDay()));
                        break;
                    case "W":
                        System.out.println("\n" + timeValue + " Second in Week: " + String.format("%.2f", time.toWeek()));
                        break;
                    case "M":
                        System.out.println("\n" + timeValue + " Second in Month: " + String.format("%.2f", time.toMonth()));
                        break;
                    case "Y":
                        System.out.println("\n" + timeValue + " Second in Year: " + String.format("%.2f", time.toYear()));
                        break;
                    case "DEC":
                        System.out.println("\n" + timeValue + " Second in Decade: " + String.format("%.2f", time.toDecade()));
                        break;
                }
            }else if (unit.equalsIgnoreCase( anotherString: "Min")){
                time = new Second(timeValue);
                switch (unit2){
                    case "S":
                        System.out.println("\n" + timeValue + " Min in Second: " + String.format("%.2f", time.toSecond()));
                        break;
                    case "H":
                        System.out.println("\n" + timeValue + " Min in Hour: " + String.format("%.2f", time.toHour()));
                        break;
                    case "D":
                        System.out.println("\n" + timeValue + " Min in Day: " + String.format("%.2f", time.toDay()));
                        break;
                    case "W":
                        System.out.println("\n" + timeValue + " Min in Week: " + String.format("%.2f", time.toWeek()));
```

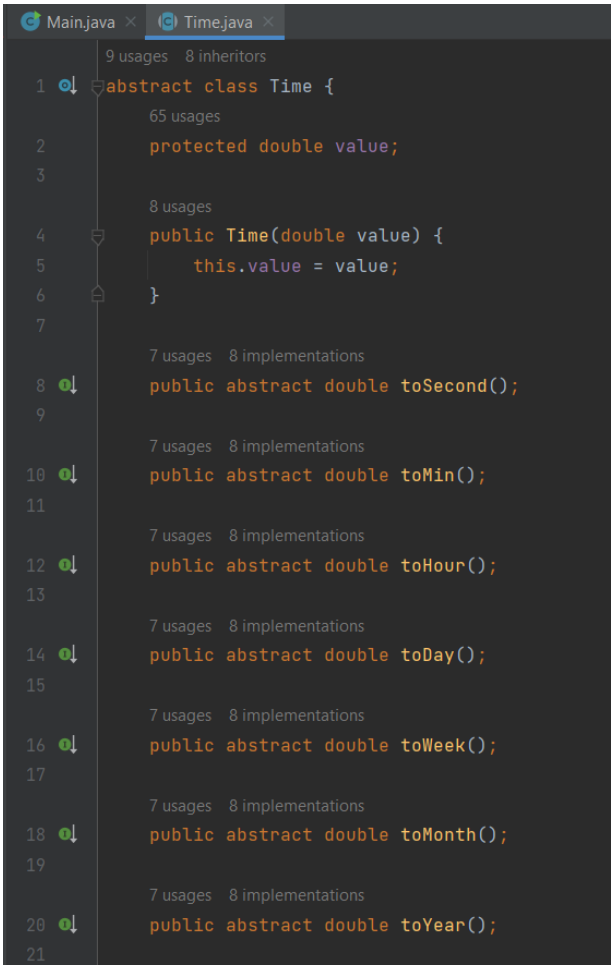
```
                    break;
                case "M":
                    System.out.println("\n" + timeValue + " Min in Month: " + String.format("%.2f", time.toMonth()));
                    break;
                case "Y":
                    System.out.println("\n" + timeValue + " Min in Year: " + String.format("%.2f", time.toYear()));
                    break;
                case "DEC":
                    System.out.println("\n" + timeValue + " Min in Decade: " + String.format("%.2f", time.toDecade()));
                    break;

            }
        }else if (unit.equalsIgnoreCase( anotherString: "H")){
            time = new Second(timeValue);
            switch (unit2){
                case "S":
                    System.out.println("\n" + timeValue + " Hour in Second: " + String.format("%.2f", time.toSecond()));
                    break;
                case "MIN":
                    System.out.println("\n" + timeValue + " Hour in Minute: " + String.format("%.2f", time.toMin()));
                    break;
                case "D":
                    System.out.println("\n" + timeValue + " Hour in Day: " + String.format("%.2f", time.toDay()));
                    break;
                case "W":
                    System.out.println("\n" + timeValue + " Hour in Week: " + String.format("%.2f", time.toWeek()));
                    break;
                case "M":
                    System.out.println("\n" + timeValue + " Hour in Month: " + String.format("%.2f", time.toMonth()));
                    break;
                case "Y":
                    System.out.println("\n" + timeValue + " Hour in Year: " + String.format("%.2f", time.toYear()));
                    break;
                case "DEC":
                    System.out.println("\n" + timeValue + " Hour in Decade: " + String.format("%.2f", time.toDecade()));
                    break;
            }
        }else if (unit.equalsIgnoreCase( anotherString: "D")){
            time = new Second(timeValue);
            switch (unit2){
                case "S":
                    System.out.println("\n" + timeValue + " Day in Second: " + String.format("%.2f", time.toSecond()));
                    break;
                case "MIN":
                    System.out.println("\n" + timeValue + " Day in Minute: " + String.format("%.2f", time.toMin()));
                    break;
                case "H":
                    System.out.println("\n" + timeValue + " Day in Hour: " + String.format("%.2f", time.toHour()));
                    break;
                case "W":
                    System.out.println("\n" + timeValue + " Day in Week: " + String.format("%.2f", time.toWeek()));
                    break;
                case "M":
                    System.out.println("\n" + timeValue + " Day in Month: " + String.format("%.2f", time.toMonth()));
                    break;
                case "Y":
                    System.out.println("\n" + timeValue + " Day in Year: " + String.format("%.2f", time.toYear()));
                    break;
                case "DEC":
                    System.out.println("\n" + timeValue + " Day in Decade: " + String.format("%.2f", time.toDecade()));
                    break;
            }
        }else if (unit.equalsIgnoreCase( anotherString: "W")){
            time = new Second(timeValue);
```

```
                switch (unit2){
                    case "S":
                        System.out.println("\n" + timeValue + " Week in Second: " + String.format("%.2f", time.toSecond()));
                        break;
                    case "MIN":
                        System.out.println("\n" + timeValue + " Week in Minute: " + String.format("%.2f", time.toMin()));
                        break;
                    case "H":
                        System.out.println("\n" + timeValue + " Week in Hour: " + String.format("%.2f", time.toHour()));
                        break;
                    case "D":
                        System.out.println("\n" + timeValue + " Week in Day: " + String.format("%.2f", time.toDay()));
                        break;
                    case "M":
                        System.out.println("\n" + timeValue + " Week in Month: " + String.format("%.2f", time.toMonth()));
                        break;
                    case "Y":
                        System.out.println("\n" + timeValue + " Week in Year: " + String.format("%.2f", time.toYear()));
                        break;
                    case "DEC":
                        System.out.println("\n" + timeValue + " Week in Decade: " + String.format("%.2f", time.toDecade()));
                        break;
                }
            }else if (unit.equalsIgnoreCase( anotherString: "M")) {
                time = new Second(timeValue);
                switch (unit2) {
                    case "S":
                        System.out.println("\n" + timeValue + " Month in Second: " + String.format("%.2f", time.toSecond()));
                        break;
                    case "MIN":
                        System.out.println("\n" + timeValue + " Month in Minute: " + String.format("%.2f", time.toMin()));
                        break;
                    case "H":
                        System.out.println("\n" + timeValue + " Month in Hour: " + String.format("%.2f", time.toHour()));
                        break;
                    case "D":
                        System.out.println("\n" + timeValue + " Month in Day: " + String.format("%.2f", time.toDay()));
                        break;
                    case "W":
                        System.out.println("\n" + timeValue + " Month in Week: " + String.format("%.2f", time.toWeek()));
                        break;
                    case "Y":
                        System.out.println("\n" + timeValue + " Month in Year: " + String.format("%.2f", time.toYear()));
                        break;
                    case "DEC":
                        System.out.println("\n" + timeValue + " Month in Decade: " + String.format("%.2f", time.toDecade()));
                        break;
                }
            }
               else if (unit.equalsIgnoreCase( anotherString: "Y")) {
                time = new Second(timeValue);
                switch (unit2) {
                    case "S":
                        System.out.println("\n" + timeValue + " Year in Second: " + String.format("%.2f", time.toSecond()));
                        break;
                    case "MIN":
                        System.out.println("\n" + timeValue + " Year in Minute: " + String.format("%.2f", time.toMin()));
                        break;
                    case "H":
                        System.out.println("\n" + timeValue + " Year in Hour: " + String.format("%.2f", time.toHour()));
                        break;
                    case "D":
                        System.out.println("\n" + timeValue + " Year in Day: " + String.format("%.2f", time.toDay()));
                        break;
```



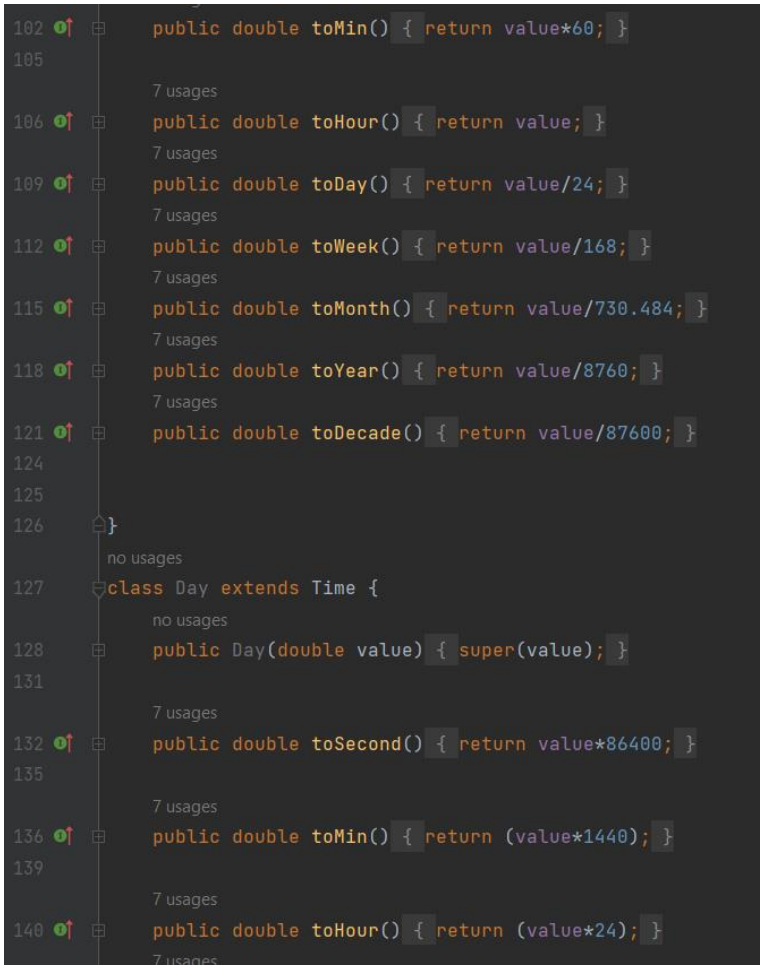
```
                case "W":
                    System.out.println("\n" + timeValue + " Year in Week: " + String.format("%.2f", time.toWeek()));
                    break;
                case "M":
                    System.out.println("\n" + timeValue + " Year in Month: " + String.format("%.2f", time.toMonth()));
                    break;
                case "DEC":
                    System.out.println("\n" + timeValue + " Year in Decade: " + String.format("%.2f", time.toDecade()));
                    break;
            }
        }else if (unit.equalsIgnoreCase( anotherString: "Dec")){
                time = new Second(timeValue);
                switch (unit2) {
                    case "S":
                        System.out.println("\n" + timeValue + " Decade in Second: " + String.format("%.2f", time.toSecond()));
                        break;
                    case "MIN":
                        System.out.println("\n" + timeValue + " Decade in Minute: " + String.format("%.2f", time.toMin()));
                        break;
                    case "H":
                        System.out.println("\n" + timeValue + " Decade in Hour: " + String.format("%.2f", time.toHour()));
                        break;
                    case "D":
                        System.out.println("\n" + timeValue + " Decade in Day: " + String.format("%.2f", time.toDay()));
                        break;
                    case "W":
                        System.out.println("\n" + timeValue + " Decade in Week: " + String.format("%.2f", time.toWeek()));
                        break;
                    case "M":
                        System.out.println("\n" + timeValue + " Decade in Month: " + String.format("%.2f", time.toMonth()));
                        break;
                    case "Y":
                        System.out.println("\n" + timeValue + " Decade in Year: " + String.format("%.2f", time.toYear()));
                        break;
                }


        }else{
            System.out.print("Invalid times unit");
            return;
        }

    }
}
```

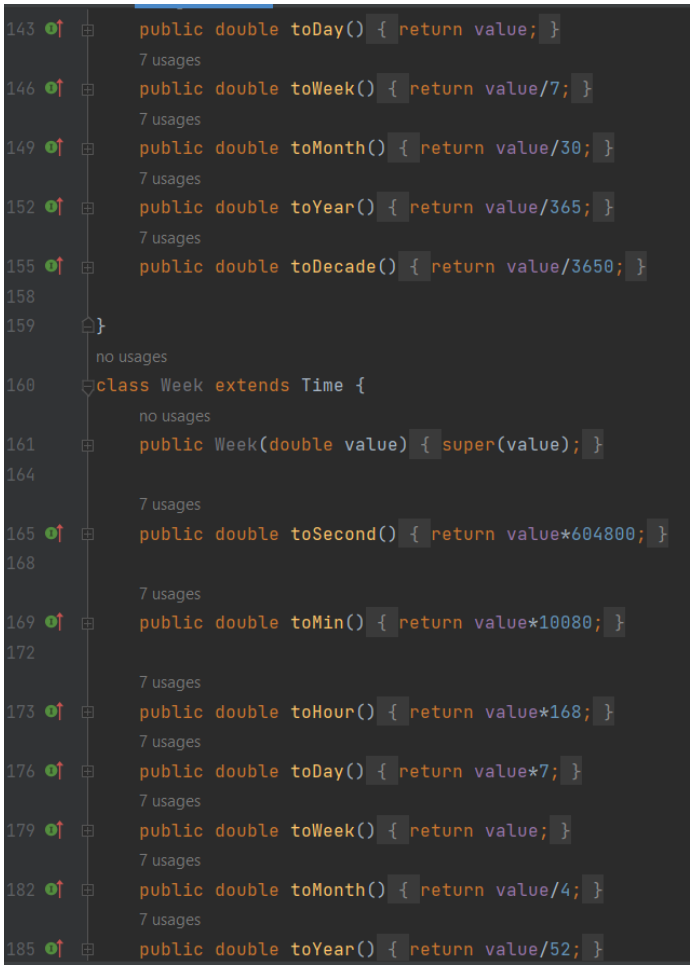
Time Class


Main.java
Time.java
9 usages 8 inheritors
abstract class Time {
65 usages
protected double value;
8 usages
public Time(double value) {
this.value = value;
}
7 usages 8 implementations
public abstract double toSecond();
7 usages 8 implementations
public abstract double toMin();
7 usages 8 implementations
public abstract double toHour();
7 usages 8 implementations
public abstract double toDay();
7 usages 8 implementations
public abstract double toWeek();
7 usages 8 implementations
public abstract double toMonth();
7 usages 8 implementations
public abstract double toYear();

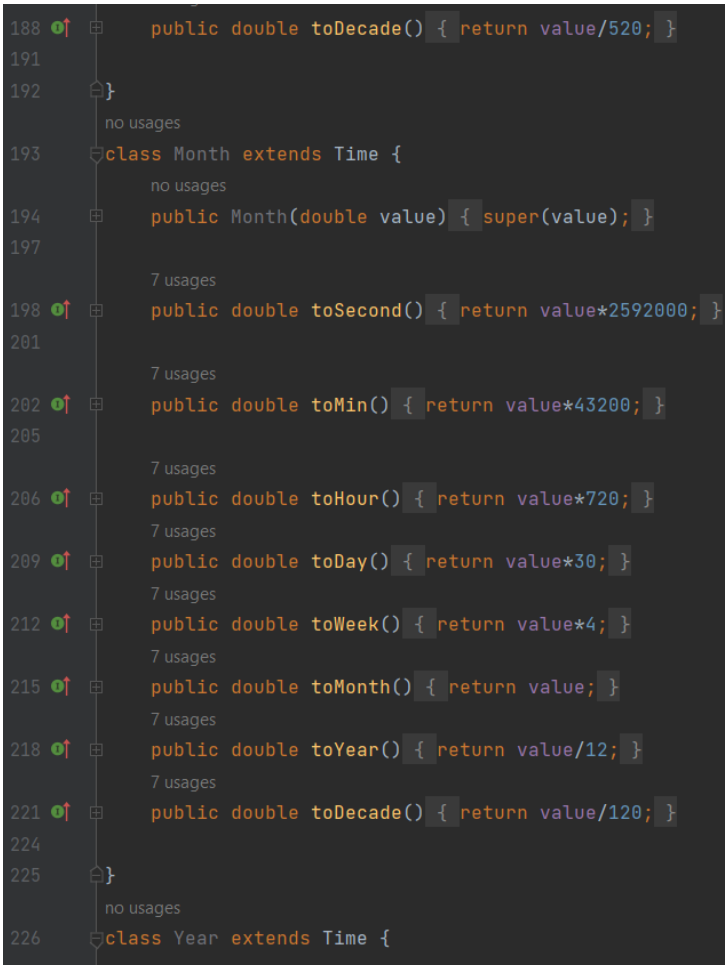
```
22      public abstract double toDecade();
23
24
25  }
    8 usages
26  class Second extends Time {
        8 usages
27      public Second(double value) { super(value); }
30
        7 usages
31      public double toSecond() { return value; }
34
        7 usages
35      public double toMin() { return (value/60); }
38
        7 usages
39      public double toHour() { return (value/3600); }
        7 usages
42      public double toDay() { return (value/86400); }
        7 usages
45      public double toWeek() { return value/604800; }
        7 usages
48      public double toMonth() { return value/2628000; }
        7 usages
51      public double toYear() { return value/31536000; }
        7 usages
54      public double toDecade() { return value/315360000; }
57
58  }
    no usages
59  class Minute extends Time {
```

```
60      public Minute(double value) { super(value); }
63
                7 usages
64      public double toSecond() { return value*60; }
67
                7 usages
68      public double toMin() { return value; }
71
                7 usages
72      public double toHour() { return (value/60); }
                7 usages
75      public double toDay() { return value/1440; }
                7 usages
78      public double toWeek() { return value/10080; }
                7 usages
81      public double toMonth() { return value/43829.1; }
                7 usages
84      public double toYear() { return value/525949; }
                7 usages
87      public double toDecade() { return value/5259492; }
90
91
92  }
        no usages
93  class Hour extends Time {
                no usages
94      public Hour(double value) { super(value); }
97
                7 usages
98      public double toSecond() { return value*3600; }
101
```

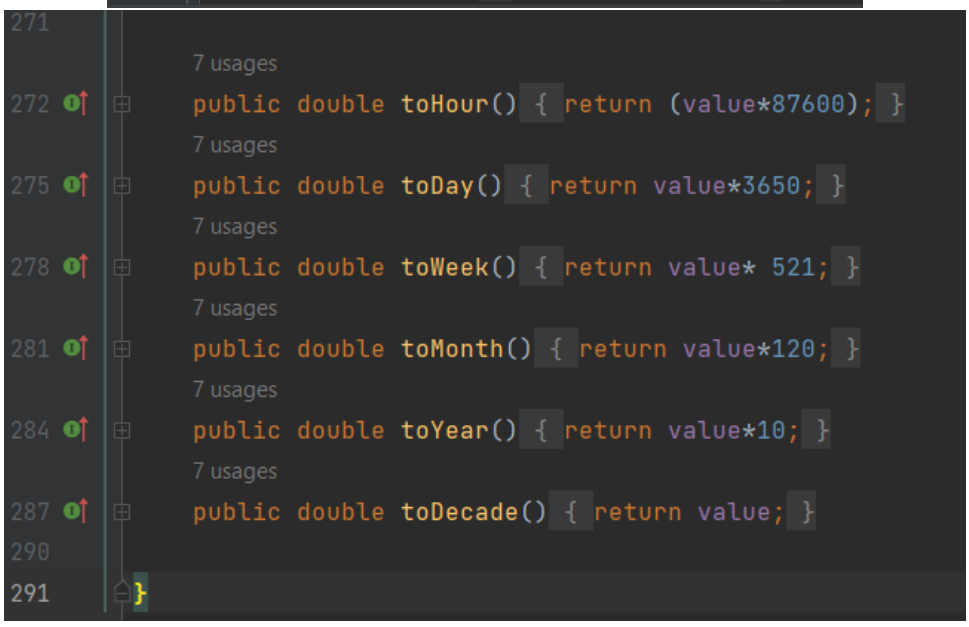

public double toMin() { return value*60; }
7 usages
public double toHour() { return value; }
7 usages
public double toDay() { return value/24; }
7 usages
public double toWeek() { return value/168; }
7 usages
public double toMonth() { return value/730.484; }
7 usages
public double toYear() { return value/8760; }
7 usages
public double toDecade() { return value/87600; }
}
no usages
class Day extends Time {
no usages
public Day(double value) { super(value); }
7 usages
public double toSecond() { return value*86400; }
7 usages
public double toMin() { return (value*1440); }
7 usages
public double toHour() { return (value*24); }
7 usages



```
143     public double toDay() { return value; }
        7 usages
146     public double toWeek() { return value/7; }
        7 usages
149     public double toMonth() { return value/30; }
        7 usages
152     public double toYear() { return value/365; }
        7 usages
155     public double toDecade() { return value/3650; }
158
159 }
    no usages
160 class Week extends Time {
        no usages
161     public Week(double value) { super(value); }
164
        7 usages
165     public double toSecond() { return value*604800; }
168
        7 usages
169     public double toMin() { return value*10080; }
172
        7 usages
173     public double toHour() { return value*168; }
        7 usages
176     public double toDay() { return value*7; }
        7 usages
179     public double toWeek() { return value; }
        7 usages
182     public double toMonth() { return value/4; }
        7 usages
185     public double toYear() { return value/52; }
```


188 public double toDecade() { return value/520; }
191
192 }
no usages
193 class Month extends Time {
no usages
194 public Month(double value) { super(value); }
197
7 usages
198 public double toSecond() { return value*2592000; }
201
7 usages
202 public double toMin() { return value*43200; }
205
7 usages
206 public double toHour() { return value*720; }
7 usages
209 public double toDay() { return value*30; }
7 usages
212 public double toWeek() { return value*4; }
7 usages
215 public double toMonth() { return value; }
7 usages
218 public double toYear() { return value/12; }
7 usages
221 public double toDecade() { return value/120; }
224
225 }
no usages
226 class Year extends Time {

```
227     public Year(double value) { super(value); }
230
            7 usages
231     public double toSecond() { return value*31536000; }
234
            7 usages
235     public double toMin() { return value*525600; }
238
            7 usages
239     public double toHour() { return value*8760; }
            7 usages
242     public double toDay() { return value*365; }
            7 usages
245     public double toWeek() { return value*52; }
            7 usages
248     public double toMonth() { return value*12; }
            7 usages
251     public double toYear() { return value; }
            7 usages
254     public double toDecade() { return value/10; }
257
258 }
        no usages
259 class Decade extends Time {
            no usages
260     public Decade(double value) { super(value); }
263
            7 usages
264     public double toSecond() { return value*315360000; }
267
            7 usages
268     public double toMin() { return (value*5259492); }
```



```
271
            7 usages
272     public double toHour() { return (value*87600); }
            7 usages
275     public double toDay() { return value*3650; }
            7 usages
278     public double toWeek() { return value* 521; }
            7 usages
281     public double toMonth() { return value*120; }
            7 usages
284     public double toYear() { return value*10; }
            7 usages
287     public double toDecade() { return value; }
290
291 }
```

Currency

Main Class

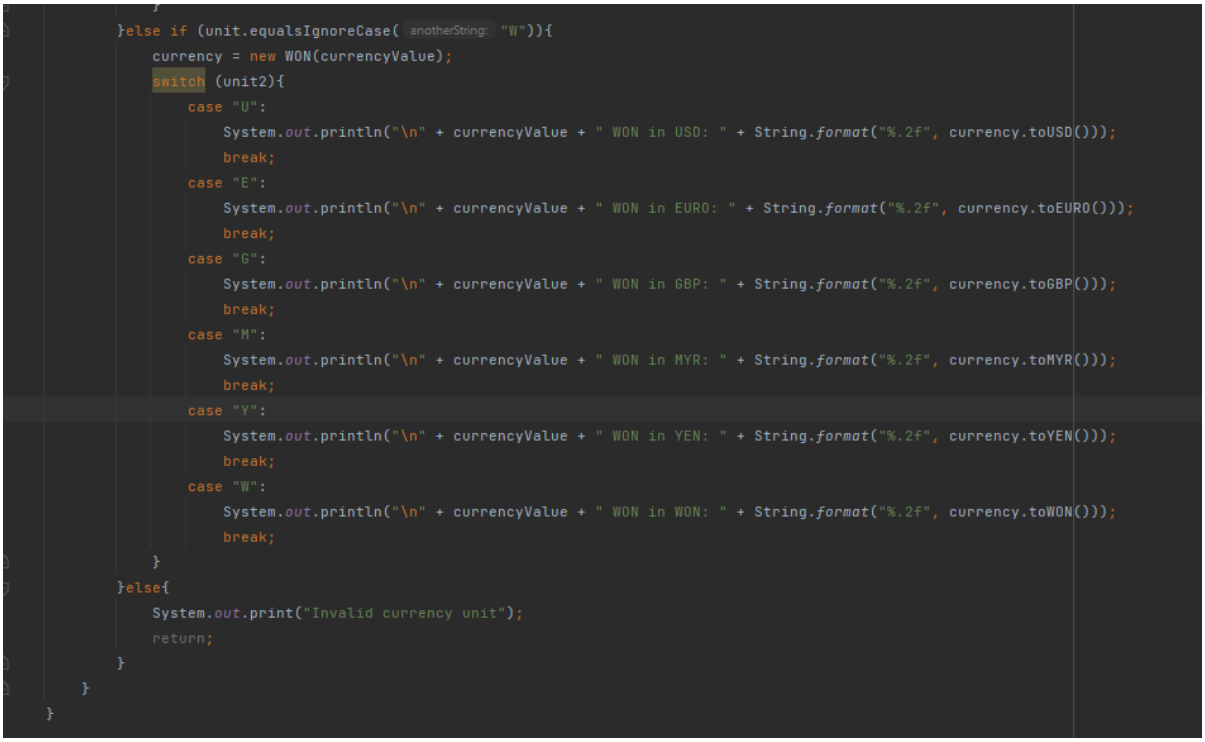
Main.java | Currency.java

```java
import java.text.DecimalFormat;
import java.text.NumberFormat;
import java.util.Scanner;

no usages
public class Main {
    no usages
    public static void main(String[] args)throws Exception{
            Scanner input = new Scanner(System.in);
            NumberFormat formatter = new DecimalFormat( pattern: "#0.0");

            System.out.print("Enter currency value: ");
            double currencyValue = input.nextDouble();

            System.out.println("1. (U) USD \n2. (G) GBP \n3. (M) MYR\n4. (E) EURO\n5. (Y) YEN\n6. (W) WON");

            System.out.print("Select unit(U/G/M/E/Y/W) (for old currency): ");
            String unit = input.next().toUpperCase();

            System.out.print("Select unit(U/G/M/E/Y/W) (for new currency): ");
            String unit2 = input.next().toUpperCase();
            input.close();

            Currency currency;
            if (unit.equalsIgnoreCase( anotherString: "U")){
                currency = new USD(currencyValue);
                switch (unit2){
                    case "U":
                        System.out.println("\n" + currencyValue + " USD in USD: " + String.format("%.2f", currency.toUSD()));
                        break;
                    case "E":
                        System.out.println("\n" + currencyValue + " USD in EURO: " + String.format("%.2f", currency.toEURO()));
                        break;
                    case "G":
                        System.out.println("\n" + currencyValue + " USD in GBP: " + String.format("%.2f", currency.toGBP()));
                        break;
                    case "M":
                        System.out.println("\n" + currencyValue + " USD in MYR: " + String.format("%.2f", currency.toMYR()));
                        break;
                    case "Y":
                        System.out.println("\n" + currencyValue + " USD in YEN: " + String.format("%.2f", currency.toYEN()));
                        break;
                    case "W":
                        System.out.println("\n" + currencyValue + " USD in WON: " + String.format("%.2f", currency.toWON()));
                        break;
                }
```

```
        }else if (unit.equalsIgnoreCase( anotherString: "E")){
            currency = new EURO(currencyValue);
            switch (unit2){
                case "U":
                    System.out.println("\n" + currencyValue + " EURO in USD: " + String.format("%.2f", currency.toUSD()));
                    break;
                case "E":
                    System.out.println("\n" + currencyValue + " EURO in EURO: " + String.format("%.2f", currency.toEURO()));
                    break;
                case "G":
                    System.out.println("\n" + currencyValue + " EURO in GBP: " + String.format("%.2f", currency.toGBP()));
                    break;
                case "M":
                    System.out.println("\n" + currencyValue + " EURO in MYR: " + String.format("%.2f", currency.toMYR()));
                    break;
                case "Y":
                    System.out.println("\n" + currencyValue + " EURO in YEN: " + String.format("%.2f", currency.toYEN()));
                    break;
                case "W":
                    System.out.println("\n" + currencyValue + " EURO in WON: " + String.format("%.2f", currency.toWON()));
                    break;
            }
        }else if (unit.equalsIgnoreCase( anotherString: "G")){
            currency = new GBP(currencyValue);
            switch (unit2){
                case "U":
                    System.out.println("\n" + currencyValue + " GBP in USD: " + String.format("%.2f", currency.toUSD()));
                    break;
                case "E":
                    System.out.println("\n" + currencyValue + " GBP in EURO: " + String.format("%.2f", currency.toEURO()));
                    break;
                case "G":
                    System.out.println("\n" + currencyValue + " GBP in GBP: " + String.format("%.2f", currency.toGBP()));
                    break;
                case "M":
                    System.out.println("\n" + currencyValue + " GBP in MYR: " + String.format("%.2f", currency.toMYR()));
                    break;
                case "Y":
                    System.out.println("\n" + currencyValue + " GBP in YEN: " + String.format("%.2f", currency.toYEN()));
                    break;
                case "W":
                    System.out.println("\n" + currencyValue + " GBP in WON: " + String.format("%.2f", currency.toWON()));
                    break;
            }
```

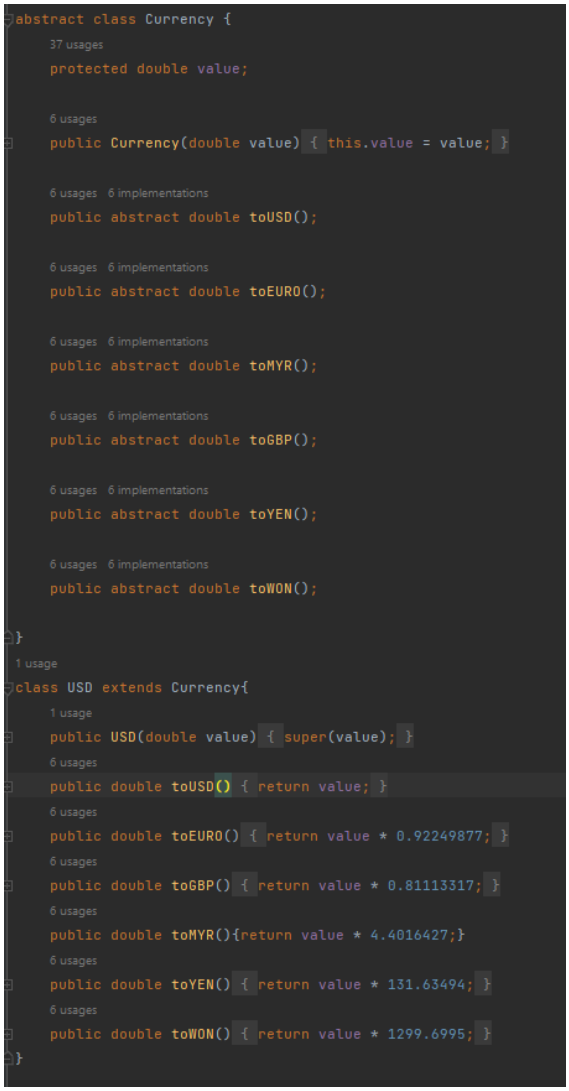
Main.java | Currency.java

```
89          }else if (unit.equalsIgnoreCase( anotherString: "M")){
90              currency = new MYR(currencyValue);
91              switch (unit2){
92                  case "U":
93                      System.out.println("\n" + currencyValue + " MYR in USD: " + String.format("%.2f", currency.toUSD()));
94                      break;
95                  case "E":
96                      System.out.println("\n" + currencyValue + " MYR in EURO: " + String.format("%.2f", currency.toEURO()));
97                      break;
98                  case "G":
99                      System.out.println("\n" + currencyValue + " MYR in GBP: " + String.format("%.2f", currency.toGBP()));
100                     break;
101                 case "M":
102                     System.out.println("\n" + currencyValue + " MYR in MYR: " + String.format("%.2f", currency.toMYR()));
103                     break;
104                 case "Y":
105                     System.out.println("\n" + currencyValue + " MYR in YEN: " + String.format("%.2f", currency.toYEN()));
106                     break;
107                 case "W":
108                     System.out.println("\n" + currencyValue + " MYR in WON: " + String.format("%.2f", currency.toWON()));
109                     break;
110             }
111         }else if (unit.equalsIgnoreCase( anotherString: "Y")){
112             currency = new YEN(currencyValue);
113             switch (unit2){
114                 case "U":
115                     System.out.println("\n" + currencyValue + " YEN in USD: " + String.format("%.2f", currency.toUSD()));
116                     break;
117                 case "E":
118                     System.out.println("\n" + currencyValue + " YEN in EURO: " + String.format("%.2f", currency.toEURO()));
119                     break;
120                 case "G":
121                     System.out.println("\n" + currencyValue + " YEN in GBP: " + String.format("%.2f", currency.toGBP()));
122                     break;
123                 case "M":
124                     System.out.println("\n" + currencyValue + " YEN in MYR: " + String.format("%.2f", currency.toMYR()));
125                     break;
126                 case "Y":
127                     System.out.println("\n" + currencyValue + " YEN in YEN: " + String.format("%.2f", currency.toYEN()));
128                     break;
129                 case "W":
130                     System.out.println("\n" + currencyValue + " YEN in WON: " + String.format("%.2f", currency.toWON()));
131                     break;
132             }
```



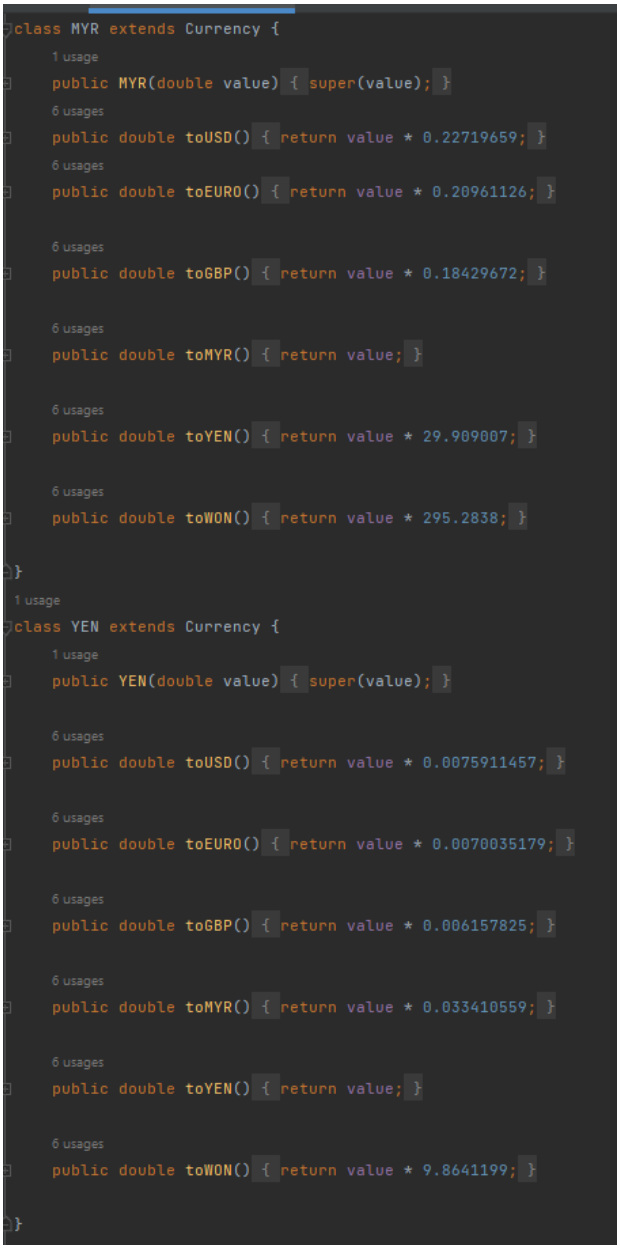
```
        }else if (unit.equalsIgnoreCase( anotherString: "W")){
            currency = new WON(currencyValue);
            switch (unit2){
                case "U":
                    System.out.println("\n" + currencyValue + " WON in USD: " + String.format("%.2f", currency.toUSD()));
                    break;
                case "E":
                    System.out.println("\n" + currencyValue + " WON in EURO: " + String.format("%.2f", currency.toEURO()));
                    break;
                case "G":
                    System.out.println("\n" + currencyValue + " WON in GBP: " + String.format("%.2f", currency.toGBP()));
                    break;
                case "M":
                    System.out.println("\n" + currencyValue + " WON in MYR: " + String.format("%.2f", currency.toMYR()));
                    break;
                case "Y":
                    System.out.println("\n" + currencyValue + " WON in YEN: " + String.format("%.2f", currency.toYEN()));
                    break;
                case "W":
                    System.out.println("\n" + currencyValue + " WON in WON: " + String.format("%.2f", currency.toWON()));
                    break;
            }
        }else{
            System.out.print("Invalid currency unit");
            return;
        }
    }
}
```

Currency Class



```
abstract class Currency {
    37 usages
    protected double value;

    6 usages
    public Currency(double value) { this.value = value; }

    6 usages  6 implementations
    public abstract double toUSD();

    6 usages  6 implementations
    public abstract double toEURO();

    6 usages  6 implementations
    public abstract double toMYR();

    6 usages  6 implementations
    public abstract double toGBP();

    6 usages  6 implementations
    public abstract double toYEN();

    6 usages  6 implementations
    public abstract double toWON();

}
 1 usage
class USD extends Currency{
    1 usage
    public USD(double value) { super(value); }
    6 usages
    public double toUSD() { return value; }
    6 usages
    public double toEURO() { return value * 0.92249877; }
    6 usages
    public double toGBP() { return value * 0.81113317; }
    6 usages
    public double toMYR(){return value * 4.4016427;}
    6 usages
    public double toYEN() { return value * 131.63494; }
    6 usages
    public double toWON() { return value * 1299.6995; }
}
```

```java
class EURO extends Currency{
    1 usage
    public EURO(double value) { super(value); }
    6 usages
    public double toUSD() { return value * 1.0839245; }
    6 usages
    public double toEURO() { return value; }

    6 usages
    public double toGBP() { return value * 0.87922779; }

    6 usages
    public double toMYR() { return value * 4.7711903; }
    6 usages
    public double toYEN() { return value * 142.68488; }
    6 usages
    public double toWON() { return value * 1408.5749; }
}

1 usage
class GBP extends Currency {
    1 usage
    public GBP(double value) { super(value); }

    6 usages
    public double toUSD() { return value * 1.2327927; }

    6 usages
    public double toEURO() { return value * 1.1373937; }

    6 usages
    public double toGBP() { return value; }

    6 usages
    public double toMYR() { return value * 5.4268354; }

    6 usages
    public double toYEN() { return value * 162.28008; }

    6 usages
    public double toWON() { return value * 1601.9348; }

}
```



```
class MYR extends Currency {
    1 usage
    public MYR(double value) { super(value); }
    6 usages
    public double toUSD() { return value * 0.22719659; }
    6 usages
    public double toEURO() { return value * 0.20961126; }

    6 usages
    public double toGBP() { return value * 0.18429672; }

    6 usages
    public double toMYR() { return value; }

    6 usages
    public double toYEN() { return value * 29.909007; }

    6 usages
    public double toWON() { return value * 295.2838; }

}
1 usage
class YEN extends Currency {
    1 usage
    public YEN(double value) { super(value); }

    6 usages
    public double toUSD() { return value * 0.0075911457; }

    6 usages
    public double toEURO() { return value * 0.0070035179; }

    6 usages
    public double toGBP() { return value * 0.006157825; }

    6 usages
    public double toMYR() { return value * 0.033410559; }

    6 usages
    public double toYEN() { return value; }

    6 usages
    public double toWON() { return value * 9.8641199; }

}
```

```java
class WON extends Currency {
    1 usage
    public WON(double value) { super(value); }

    6 usages
    public double toUSD() { return value * 0.00076963052; }

    6 usages
    public double toEURO() { return value * 0.00071010101; }

    6 usages
    public double toGBP() { return value * 0.00062434176; }

    6 usages
    public double toMYR() { return value * 0.0033875979; }

    6 usages
    public double toYEN() { return value * 0.10137495; }

    6 usages
    public double toWON() { return value; }

}
```

Pressure

Main class

Main.java | Pressure.java

```java
import java.text.DecimalFormat;
import java.text.NumberFormat;
import java.util.*;

public class Main {
    public static void main(String[] args)throws Exception {
        Scanner input = new Scanner(System.in);
        NumberFormat formatter = new DecimalFormat( pattern: "#0.0000");

        System.out.print("Enter a Pressure Value: ");
        double pressureValue = input.nextDouble();

        System.out.print("Enter the Pressure Unit (PA/ATM): ");
        String unit1 = input.next().toUpperCase();

        System.out.print("Enter the Pressure Unit conversion (PA/ATM): ");
        String unit2 = input.next().toUpperCase();
        input.close();
```

```java
        Pressure pressure;
        if (unit1.equalsIgnoreCase( anotherString: "PA")) {
            pressure = new Pascal(pressureValue);
            switch (unit2) {
                case "PA":
                    System.out.println("Pressure in Pascal to Pascal: " + formatter.format(pressure.toPascal()));
                    break;
                case "ATM":
                    System.out.println("Pressure in Pascal to Atmosphere: " + formatter.format(pressure.toAtmosphere()));
            }
        } else if (unit1.equalsIgnoreCase( anotherString: "ATM")) {
            pressure = new Atmosphere(pressureValue);
            switch (unit2) {
                case "PA":
                    System.out.println("Pressure in Atmosphere to Pascal: " + formatter.format(pressure.toPascal()));
                    break;
                case "ATM":
                    System.out.println("Pressure in Atmosphere to Atmosphere: " + formatter.format(pressure.toAtmosphere()));
            }
        } else {
            System.out.print("Invalid Pressure Unit");
            return;
        }
    }
}
```

Pressure class

Main.java | Pressure.java

```java
abstract class Pressure {
    protected double value;

    public Pressure(double value) {
        this.value = value;
    }

    public abstract double toPascal();

    public abstract double toAtmosphere();
}

class Pascal extends Pressure {
    public Pascal(double value) {
        super(value);
    }

    public double toPascal() {
        return value;
    }

    public double toAtmosphere() {
        return value / 101325;
    }
}

class Atmosphere extends Pressure {
    public Atmosphere(double value) {
        super(value);
    }

    public double toPascal() {
        return value * 101325;
    }

    public double toAtmosphere() {
        return value;
    }
}
```

Area

Main class

```java
import java.text.DecimalFormat;
import java.text.NumberFormat;
import java.util.Scanner;

no usages
public class Main {
    no usages
    public static void main(String[] args)throws Exception{
        Scanner input = new Scanner(System.in);
        NumberFormat formatter = new DecimalFormat( pattern: "#0.0");

        System.out.print("Enter currency value: ");
        double areaValue = input.nextDouble();

        System.out.println("1. (SF) Square Meter \n2. (SI) Square Inches \n3. (SKM) Square Kilometer\n4. (SF) Square Foot\n5. (SMILI) Square Millimeter\n6. (SCM) Square Centimeter");

        System.out.print("Select unit(SF/SI/SKM/SF/SMILI/SCM) (for old currency): ");
        String unit = input.next().toUpperCase();

        System.out.print("Select unit(SF/SI/SKM/SF/SMILI/SCM) (for new currency): ");
        String unit2 = input.next().toUpperCase();
        input.close();

        Area area;
        if (unit.equalsIgnoreCase( anotherString: "SF")){
            area = new SquareM(areaValue);
            switch (unit2){
                case "SF":
                    System.out.println("\n" + areaValue + " SquareM in SquareM: " + String.format("%.2f", area.toSquareM()));
                    break;
                case "SM":
                    System.out.println("\n" + areaValue + " SquareM in SquareF: " + String.format("%.2f", area.toSquareF()));
                    break;
                case "SI":
                    System.out.println("\n" + areaValue + " SquareM in SquareI: " + String.format("%.2f", area.toSquareI()));
                    break;
                case "SK":
                    System.out.println("\n" + areaValue + " SquareM in SquareKM: " + String.format("%.2f", area.toSquareKM()));
                    break;
                case "SMILI":
                    System.out.println("\n" + areaValue + " SquareM in SquareMili: " + String.format("%.2f", area.toSquareMili()));
                    break;
                case "SCM":
                    System.out.println("\n" + areaValue + " SquareM in SquareCM: " + String.format("%.2f", area.toSquareCM()));
                    break;
            }
```

Main.java Area.java

```java
        }else if (unit.equalsIgnoreCase( anotherString: "SM")){
            area = new SquareF(areaValue);
            switch (unit2){
                case "SF":
                    System.out.println("\n" + areaValue + " SquareF in SquareM: " + String.format("%.2f", area.toSquareM()));
                    break;
                case "SM":
                    System.out.println("\n" + areaValue + " SquareF in SquareF: " + String.format("%.2f", area.toSquareF()));
                    break;
                case "SI":
                    System.out.println("\n" + areaValue + " SquareF in SquareI: " + String.format("%.2f", area.toSquareI()));
                    break;
                case "SK":
                    System.out.println("\n" + areaValue + " SquareF in SquareKM: " + String.format("%.2f", area.toSquareKM()));
                    break;
                case "SMILI":
                    System.out.println("\n" + areaValue + " SquareF in SquareMili: " + String.format("%.2f", area.toSquareMili()));
                    break;
                case "SCM":
                    System.out.println("\n" + areaValue + " SquareF in SquareCM: " + String.format("%.2f", area.toSquareCM()));
                    break;
            }
        }else if (unit.equalsIgnoreCase( anotherString: "SI")){
            area = new SquareI(areaValue);
            switch (unit2){
                case "SF":
                    System.out.println("\n" + areaValue + " SquareI in SquareM: " + String.format("%.2f", area.toSquareM()));
                    break;
                case "SM":
                    System.out.println("\n" + areaValue + " SquareI in SquareF: " + String.format("%.2f", area.toSquareF()));
                    break;
                case "SI":
                    System.out.println("\n" + areaValue + " SquareI in SquareI: " + String.format("%.2f", area.toSquareI()));
                    break;
                case "SK":
                    System.out.println("\n" + areaValue + " SquareI in SquareKM: " + String.format("%.2f", area.toSquareKM()));
                    break;
                case "SMILI":
                    System.out.println("\n" + areaValue + " SquareI in SquareMili: " + String.format("%.2f", area.toSquareMili()));
                    break;
                case "SCM":
                    System.out.println("\n" + areaValue + " SquareI in SquareCM: " + String.format("%.2f", area.toSquareCM()));
                    break;
            }
```

Main.java Area.java

```
        }else if (unit.equalsIgnoreCase( anotherString: "SK")){
            area = new SquareKM(areaValue);
            switch (unit2){
                case "SF":
                    System.out.println("\n" + areaValue + " SquareKM in SquareM: " + String.format("%.2f", area.toSquareM()));
                    break;
                case "SM":
                    System.out.println("\n" + areaValue + " SquareKM in SquareF: " + String.format("%.2f", area.toSquareF()));
                    break;
                case "SI":
                    System.out.println("\n" + areaValue + " SquareKM in SquareI: " + String.format("%.2f", area.toSquareI()));
                    break;
                case "SK":
                    System.out.println("\n" + areaValue + " SquareKM in SquareKM: " + String.format("%.2f", area.toSquareKM()));
                    break;
                case "SMILI":
                    System.out.println("\n" + areaValue + " SquareKM in SquareMili: " + String.format("%.2f", area.toSquareMili()));
                    break;
                case "SCM":
                    System.out.println("\n" + areaValue + " SquareKM in SquareCM: " + String.format("%.2f", area.toSquareCM()));
                    break;
            }
        }else if (unit.equalsIgnoreCase( anotherString: "SMILI")){
            area = new SquareMili(areaValue);
            switch (unit2){
                case "SF":
                    System.out.println("\n" + areaValue + " SquareMili in SquareM: " + String.format("%.2f", area.toSquareM()));
                    break;
                case "SM":
                    System.out.println("\n" + areaValue + " SquareMili in SquareF: " + String.format("%.2f", area.toSquareF()));
                    break;
                case "SI":
                    System.out.println("\n" + areaValue + " SquareMili in SquareI: " + String.format("%.2f", area.toSquareI()));
                    break;
                case "SK":
                    System.out.println("\n" + areaValue + " SquareMili in SquareKM: " + String.format("%.2f", area.toSquareKM()));
                    break;
                case "SMILI":
                    System.out.println("\n" + areaValue + " SquareMili in SquareMili: " + String.format("%.2f", area.toSquareMili()));
                    break;
                case "SCM":
                    System.out.println("\n" + areaValue + " SquareMili in SquareCM: " + String.format("%.2f", area.toSquareCM()));
                    break;
            }
```

```
        }else if (unit.equalsIgnoreCase( anotherString: "SCM")){
            area = new SquareCM(areaValue);
            switch (unit2){
                case "SF":
                    System.out.println("\n" + areaValue + " SquareCM in SquareM: " + String.format("%.2f", area.toSquareM()));
                    break;
                case "SM":
                    System.out.println("\n" + areaValue + " SquareCM in SquareF: " + String.format("%.2f", area.toSquareF()));
                    break;
                case "SI":
                    System.out.println("\n" + areaValue + " SquareCM in SquareI: " + String.format("%.2f", area.toSquareI()));
                    break;
                case "SK":
                    System.out.println("\n" + areaValue + " SquareCM in SquareKM: " + String.format("%.2f", area.toSquareKM()));
                    break;
                case "SMILI":
                    System.out.println("\n" + areaValue + " SquareCM in SquareMili: " + String.format("%.2f", area.toSquareMili()));
                    break;
                case "SCM":
                    System.out.println("\n" + areaValue + " SquareCM in SquareCM: " + String.format("%.2f", area.toSquareCM()));
                    break;
            }
        }else{
            System.out.print("Invalid area unit");
            return;
        }
    }
}
```

Area class

```java
abstract class Area {
    37 usages
    protected double value;

    6 usages
    public Area(double value) { this.value = value; }

    6 usages  6 implementations
    public abstract double toSquareM();

    6 usages  6 implementations
    public abstract double toSquareF();

    6 usages  6 implementations
    public abstract double toSquareI();
    6 usages  6 implementations
    public abstract double toSquareKM();
    6 usages  6 implementations
    public abstract double toSquareMili();

    6 usages  6 implementations
    public abstract double toSquareCM();

}
 1 usage
class SquareM extends Area{
    1 usage
    public SquareM(double value) { super(value); }
    6 usages
    public double toSquareM() { return value; }
    6 usages
    public double toSquareF() { return value * 10.76391; }
    6 usages
    public double toSquareI() { return value * 15504; }
    6 usages
    public double toSquareKM(){return value * (10*-6);}
    6 usages
    public double toSquareMili() { return value * 1000000; }
    6 usages
    public double toSquareCM() { return value * 10000; }
}
 1 usage
```

```java
class SquareF extends Area{
    1 usage
    public SquareF(double value) { super(value); }
    6 usages
    public double toSquareF() { return value; }
    6 usages
    public double toSquareM() { return value * 0.09290304; }
    6 usages
    public double toSquareI() { return value * 144; }
    6 usages
    public double toSquareKM(){return value * 0.000000092903;}
    6 usages
    public double toSquareMili() { return value * 92903; }
    6 usages
    public double toSquareCM() { return value * 929.03; }
}
1 usage
class SquareI extends Area{
    1 usage
    public SquareI(double value) { super(value); }
    6 usages
    public double toSquareI() { return value; }
    6 usages
    public double toSquareF() { return value * 0.00694444; }
    6 usages
    public double toSquareM() { return value * 0.00064516; }
    6 usages
    public double toSquareKM(){return value * 0.00000000064516;}
    6 usages
    public double toSquareMili() { return value * 645.16; }
    6 usages
    public double toSquareCM() { return value * 6.4516; }
}
```

```java
class SquareKM extends Area{
    1 usage
    public SquareKM(double value) { super(value); }
    6 usages
    public double toSquareKM() { return value; }
    6 usages
    public double toSquareF() { return value * 0.00001076; }
    6 usages
    public double toSquareI() { return value * 1550003100; }
    6 usages
    public double toSquareM(){return value * 1000000;}
    6 usages
    public double toSquareMili() { return value * 0.000000000001; }
    6 usages
    public double toSquareCM() { return value * (10*10); }
}
1 usage
class SquareMili extends Area{
    1 usage
    public SquareMili(double value){super(value);}
    6 usages
    public double toSquareMili() { return value; }
    6 usages
    public double toSquareF() { return value * 0.000010764; }
    6 usages
    public double toSquareI() { return value * 0.00155; }
    6 usages
    public double toSquareKM(){return value * (10*12);}
    6 usages
    public double toSquareM() { return value * 0.000001; }
    6 usages
    public double toSquareCM() { return value * 0.01; }
}
```

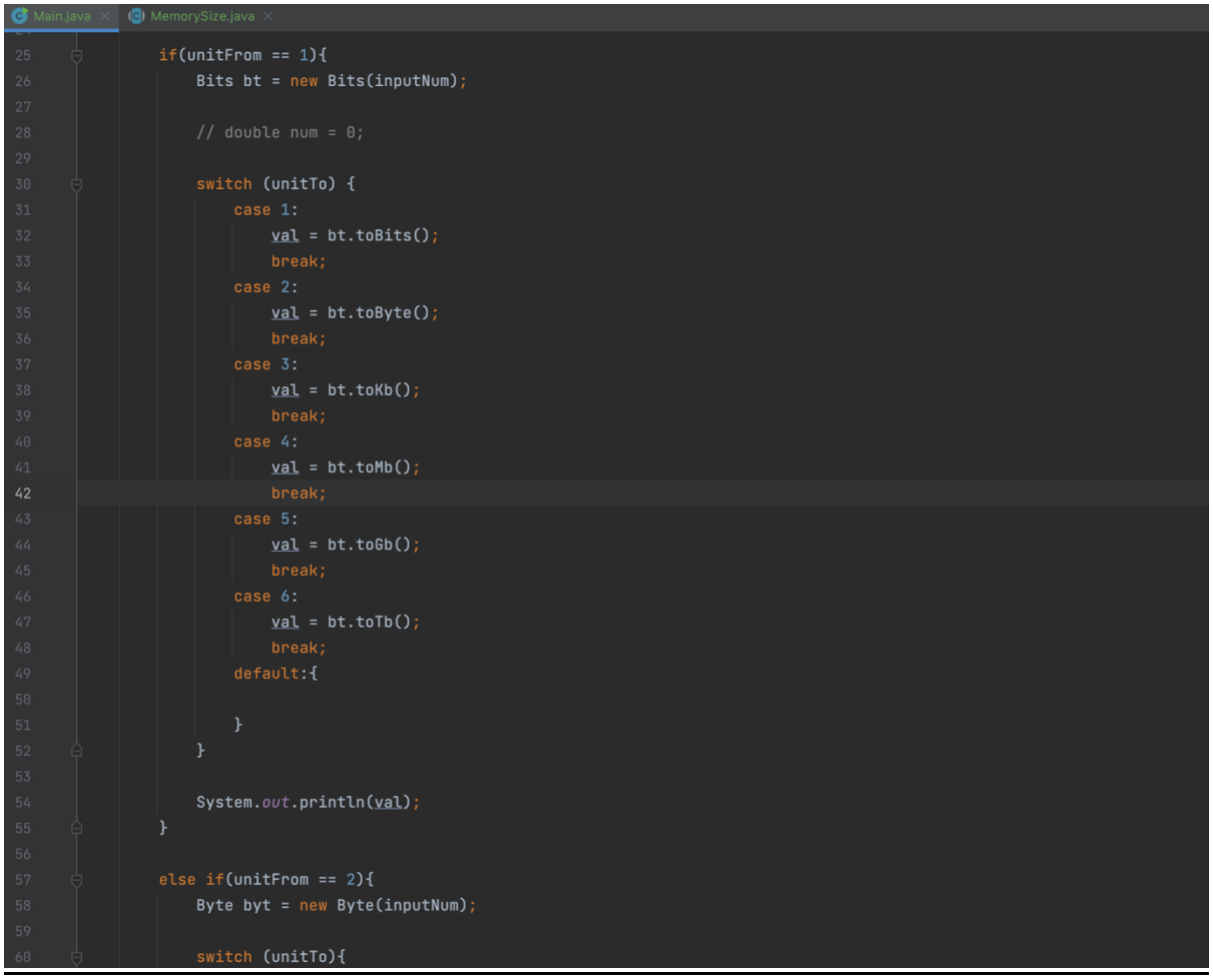
```java
class SquareCM extends Area{
    1 usage
    public SquareCM(double value) { super(value); }
    6 usages
    public double toSquareCM() { return value; }
    6 usages
    public double toSquareF() { return value * 0.0010763888889; }
    6 usages
    public double toSquareI() { return value * 0.155; }
    6 usages
    public double toSquareKM(){return value * (10*10);}
    6 usages
    public double toSquareMili() { return value * 100; }
    6 usages
    public double toSquareM() { return value * 0.0001; }
}
```

Memory Size

Main class

Main.java | MemorySize.java

```java
import java.util.Scanner;

new *
public class Main {
    new *
    public static void main(String[] args) {
        String[] units = {"Bits", "Byte (B)", "Kilobyte (KB)", "Megabyte (MB)", "Gigabyte (GB)", "Terabyte (TB)"};

        double val = 0;

        Scanner input = new Scanner(System.in);

        System.out.print("Enter value: ");
        double inputNum = input.nextDouble();

        for(int i = 0; i< units.length; i++){
            System.out.println("[" + (i+1) + "] " + units[i]);

        }

        System.out.print("Choose initial unit: ");
        int unitFrom = input.nextInt();

        System.out.print("Choose conversion unit: ");
        int unitTo = input.nextInt();
```


Main.java
MemorySize.java
if(unitFrom == 1){
Bits bt = new Bits(inputNum);
// double num = 0;
switch (unitTo) {
case 1:
val = bt.toBits();
break;
case 2:
val = bt.toByte();
break;
case 3:
val = bt.toKb();
break;
case 4:
val = bt.toMb();
break;
case 5:
val = bt.toGb();
break;
case 6:
val = bt.toTb();
break;
default:{
}
}
System.out.println(val);
}
else if(unitFrom == 2){
Byte byt = new Byte(inputNum);
switch (unitTo){

Main.java MemorySize.java

```
    else if(unitFrom == 2){
        Byte byt = new Byte(inputNum);

        switch (unitTo){
            case 1:
                val = byt.toBits();
                break;
            case 2:
                val = byt.toByte();
                break;
            case 3:
                val = byt.toKb();
                break;
            case 4:
                val = byt.toMb();
                break;
            case 5:
                val = byt.toGb();
                break;
            case 6:
                val = byt.toTb();
                break;
            default:{

            }
        }

        System.out.println(val);
    }

    else if(unitFrom == 3){
        KiloBytes kb = new KiloBytes(inputNum);

        switch (unitTo) {
            case 1:
                val = kb.toBits();
```

Main.java MemorySize.java

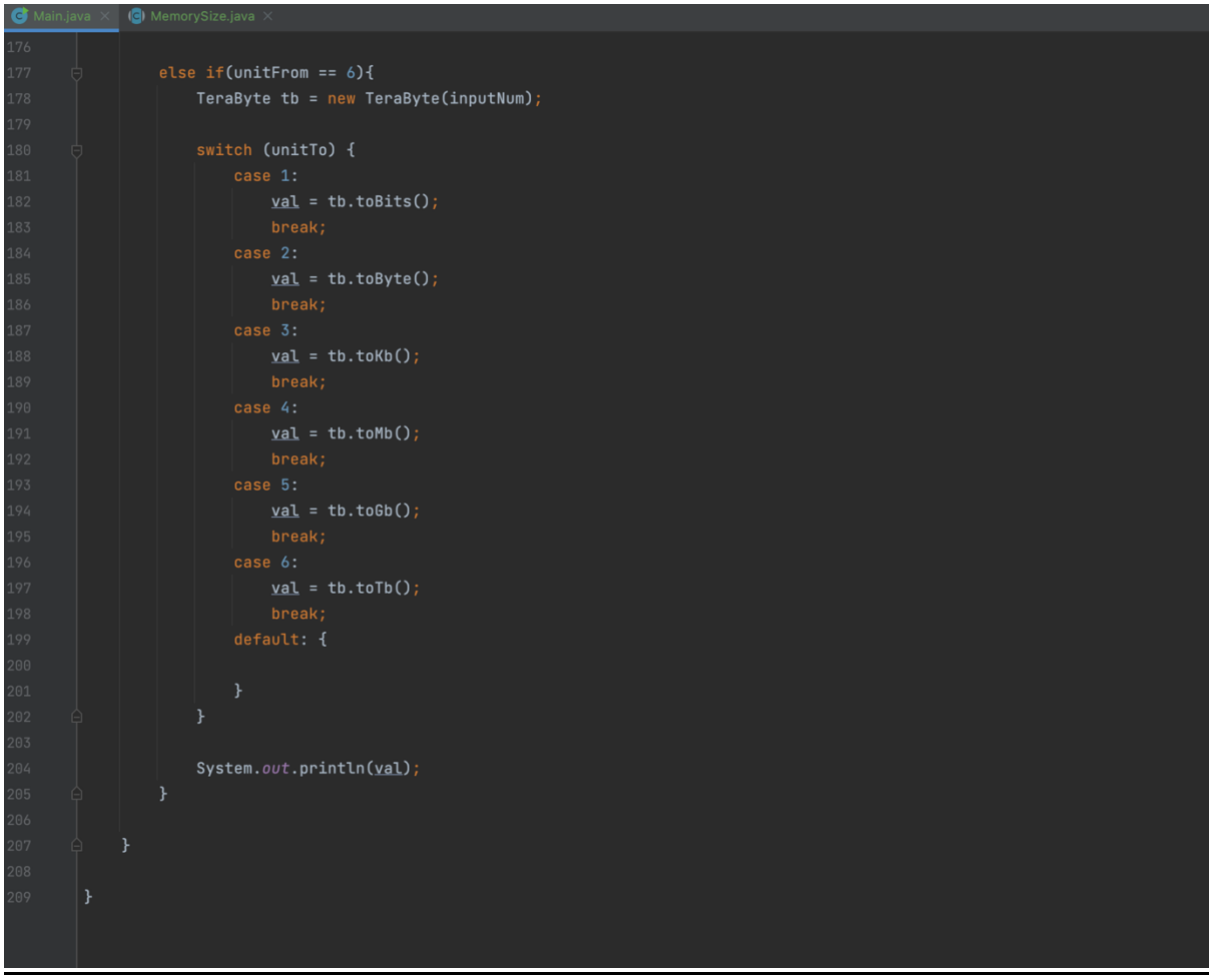
```
87        else if(unitFrom == 3){
88            KiloBytes kb = new KiloBytes(inputNum);
89
90            switch (unitTo) {
91                case 1:
92                    val = kb.toBits();
93                    break;
94                case 2:
95                    val = kb.toByte();
96                    break;
97                case 3:
98                    val = kb.toKb();
99                    break;
100               case 4:
101                   val = kb.toMb();
102                   break;
103               case 5:
104                   val = kb.toGb();
105                   break;
106               case 6:
107                   val = kb.toTb();
108                   break;
109               default: {
110
111               }
112           }
113
114           System.out.println(val);
115       }
116
117       else if(unitFrom == 4){
118           MegaByte mb = new MegaByte(inputNum);
119
120           switch (unitTo) {
121               case 1:
122                   val = mb.toBits();
```

Main.java | MemorySize.java

```
117        else if(unitFrom == 4){
118            MegaByte mb = new MegaByte(inputNum);
119
120            switch (unitTo) {
121                case 1:
122                    val = mb.toBits();
123                    break;
124                case 2:
125                    val = mb.toByte();
126                    break;
127                case 3:
128                    val = mb.toKb();
129                    break;
130                case 4:
131                    val = mb.toMb();
132                    break;
133                case 5:
134                    val = mb.toGb();
135                    break;
136                case 6:
137                    val = mb.toTb();
138                    break;
139                default: {
140
141                }
142            }
143
144            System.out.println(val);
145        }
146
147        else if(unitFrom == 5){
148            GigaByte gb = new GigaByte(inputNum);
149
150            switch (unitTo) {
151                case 1:
152                    gb.toBits();
```

Main.java MemorySize.java

```
147        else if(unitFrom == 5){
148            GigaByte gb = new GigaByte(inputNum);
149
150            switch (unitTo) {
151                case 1:
152                    gb.toBits();
153                    break;
154                case 2:
155                    gb.toByte();
156                    break;
157                case 3:
158                    gb.toKb();
159                    break;
160                case 4:
161                    gb.toMb();
162                    break;
163                case 5:
164                    gb.toGb();
165                    break;
166                case 6:
167                    gb.toTb();
168                    break;
169                default: {
170
171                }
172            }
173
174            System.out.println(val);
175        }
176
177        else if(unitFrom == 6){
178            TeraByte tb = new TeraByte(inputNum);
179
180            switch (unitTo) {
181                case 1:
182                    val = tb.toBits();
```


Main.java
MemorySize.java
176
177 else if(unitFrom == 6){
178 TeraByte tb = new TeraByte(inputNum);
179
180 switch (unitTo) {
181 case 1:
182 val = tb.toBits();
183 break;
184 case 2:
185 val = tb.toByte();
186 break;
187 case 3:
188 val = tb.toKb();
189 break;
190 case 4:
191 val = tb.toMb();
192 break;
193 case 5:
194 val = tb.toGb();
195 break;
196 case 6:
197 val = tb.toTb();
198 break;
199 default: {
200
201 }
202 }
203
204 System.out.println(val);
205 }
206
207 }
208
209 }

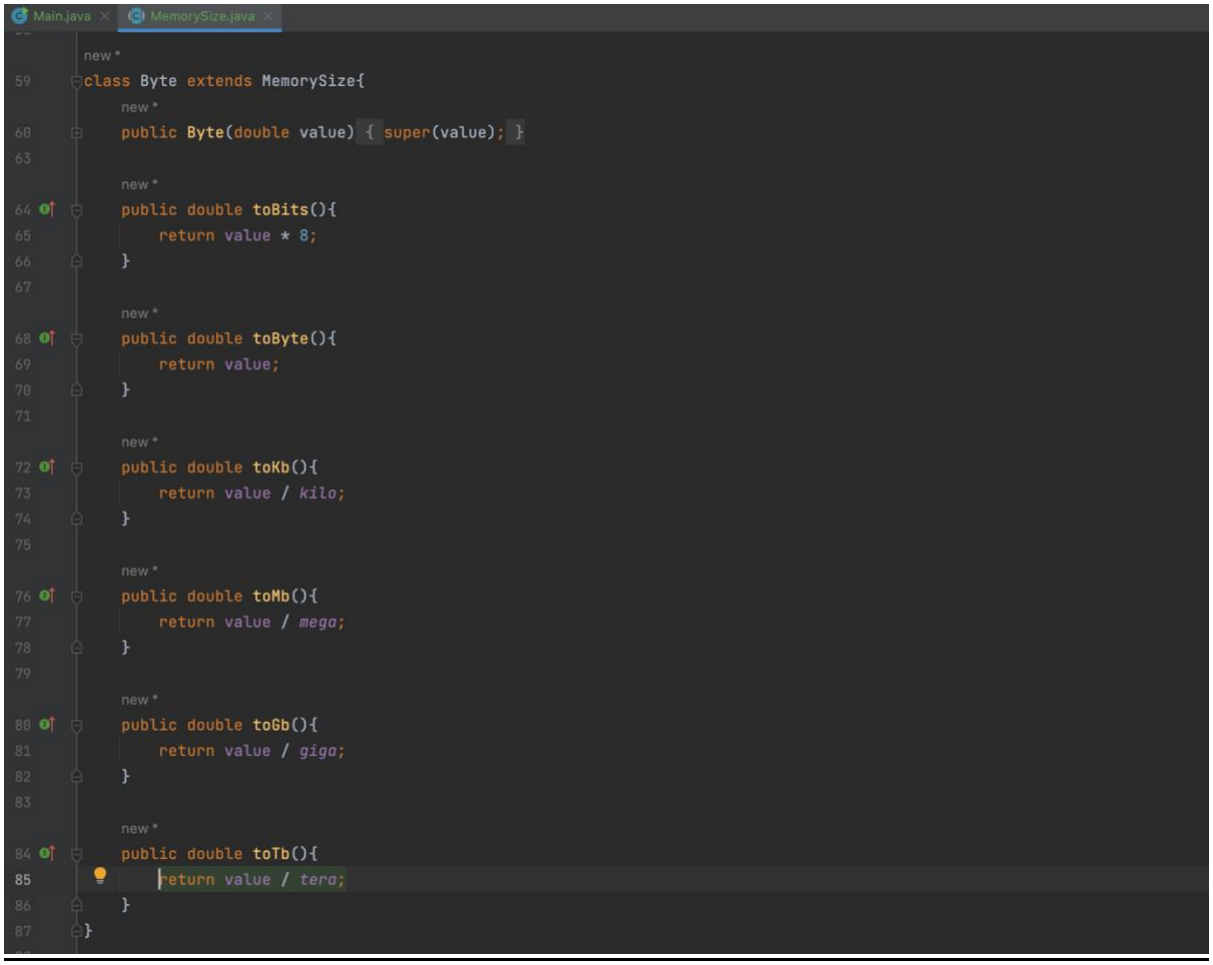
Memory Size class

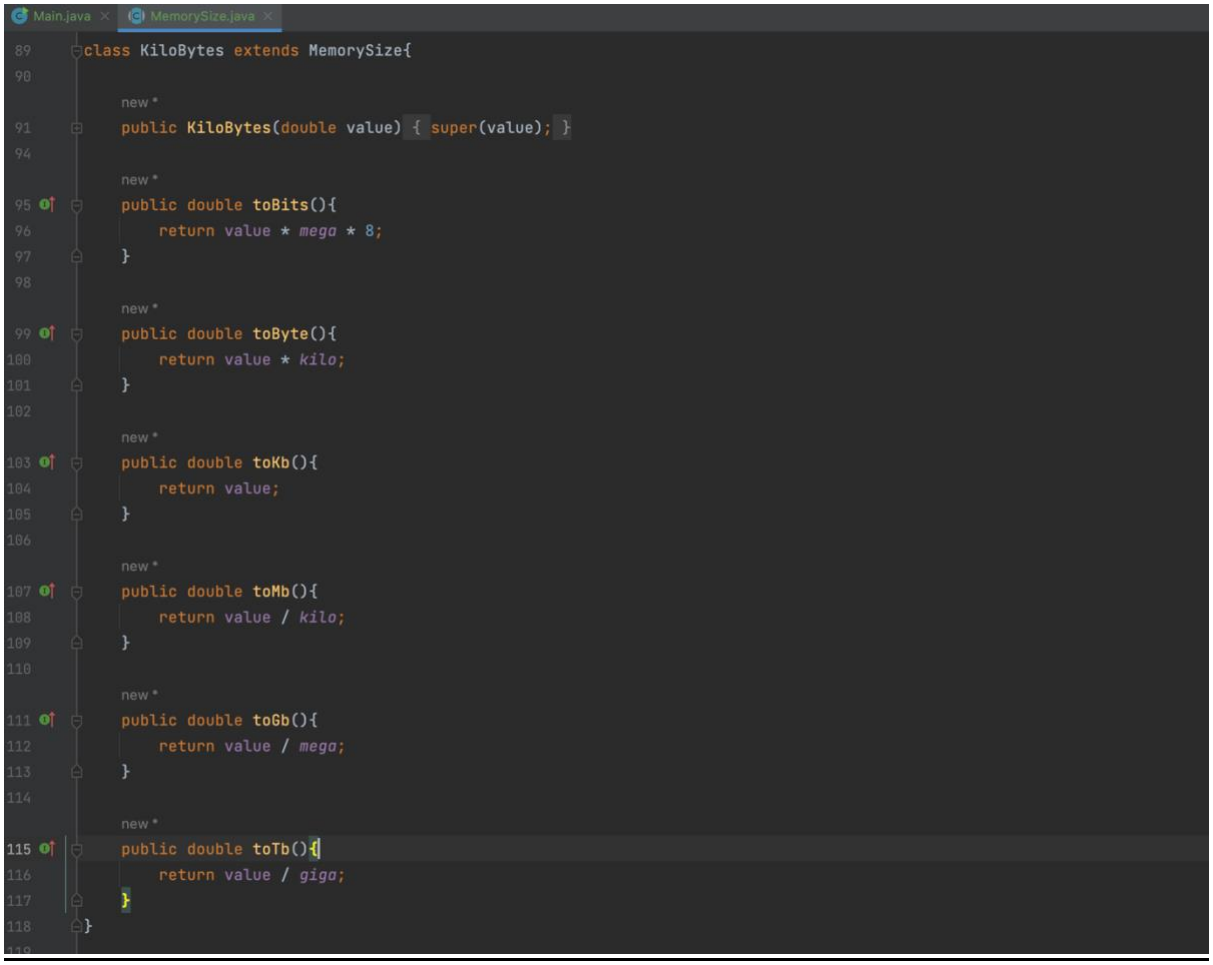
Main.java | MemorySize.java

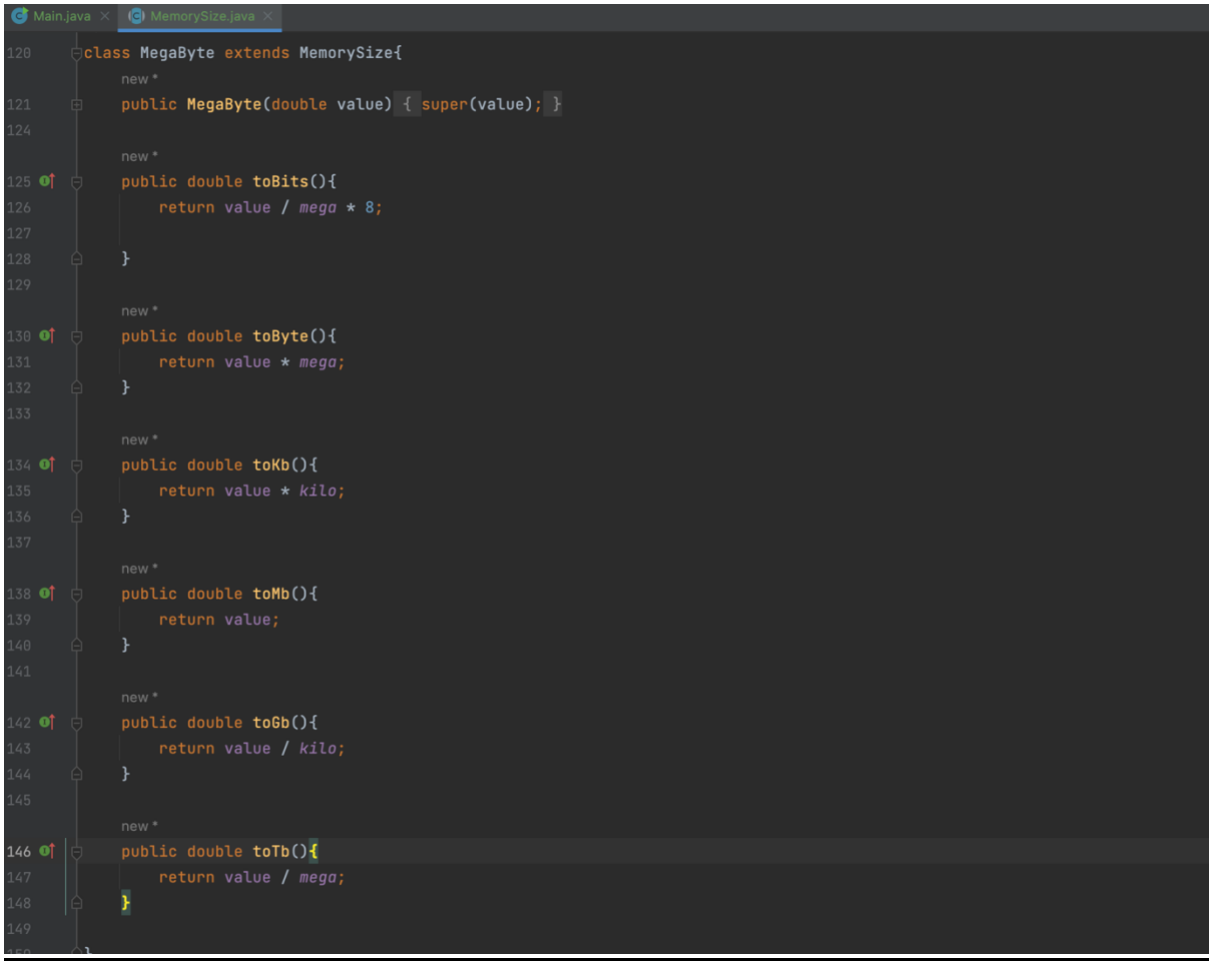
```
6 inheritors  new *
public abstract class MemorySize {

    protected double value;


    static double kilo = 1024;
    static double mega = kilo * kilo;
    static double giga = mega * kilo;
    static double tera = giga * kilo;

    new *
    public MemorySize(double value) { this.value = value; }

    6 implementations  new *
    abstract double toBits();
    6 implementations  new *
    abstract double toByte();
    6 implementations  new *
    abstract double toKb();
    6 implementations  new *
    abstract double toMb();
    6 implementations  new *
    abstract double toGb();
    6 implementations  new *
    abstract double toTb();

}
```

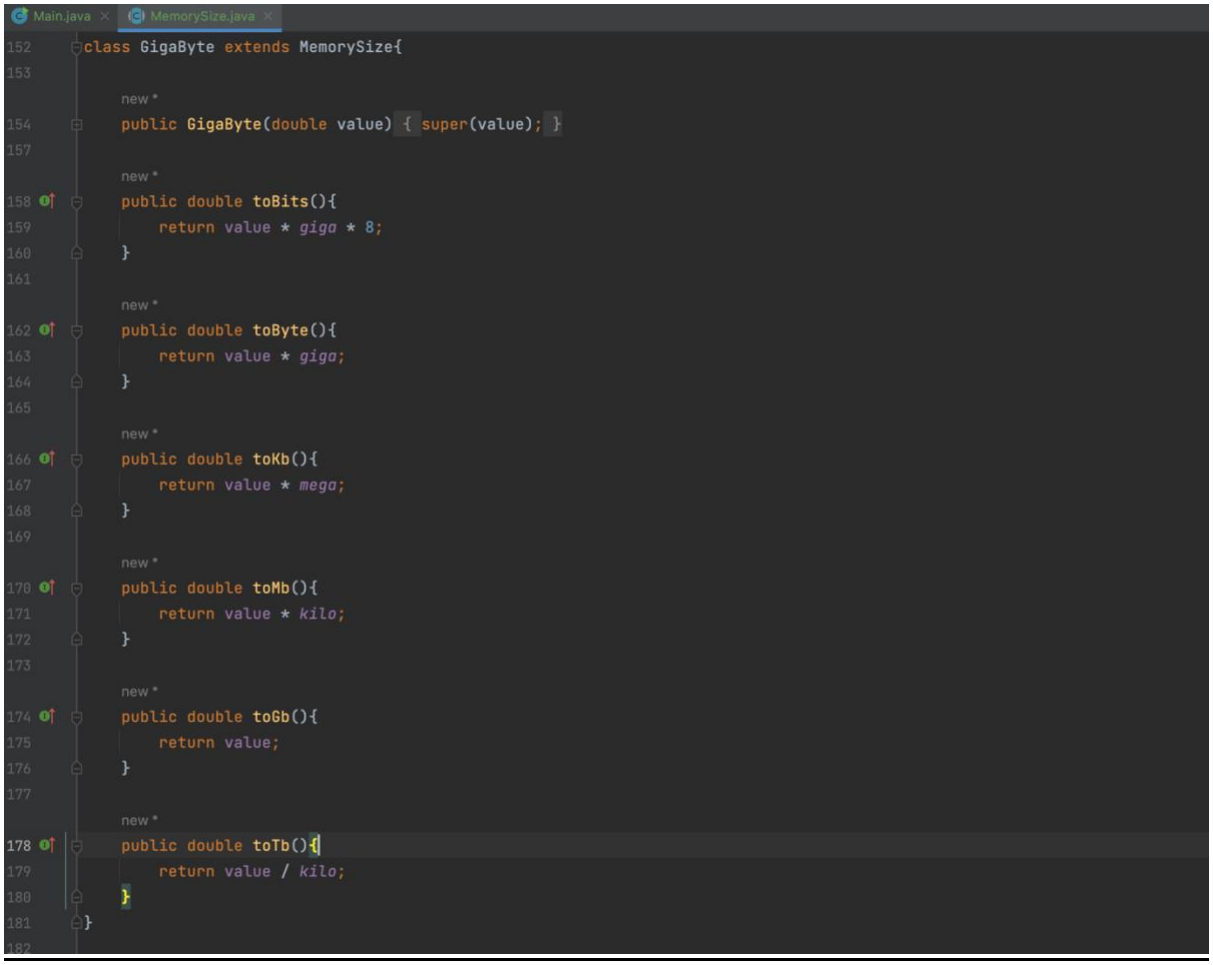
Main.java | MemorySize.java

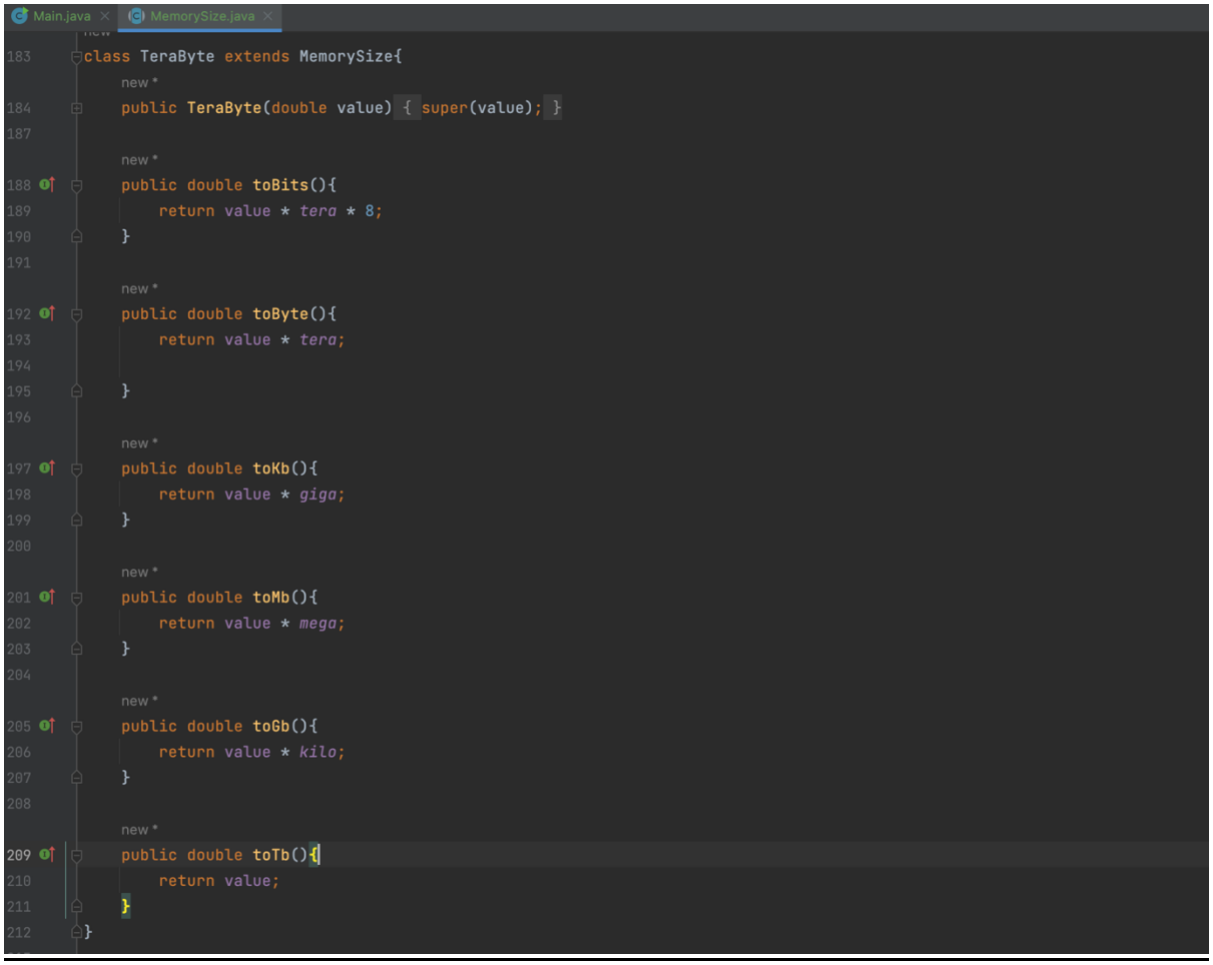
```
new *
class Bits extends MemorySize{
    new *
    public Bits(double value) { super(value); }

    new *
    public double toBits() { return value; }

    new *
    public double toByte() { return value / 8; }

    new *
    public double toKb(){

        return (value / 8) / kilo;
    }

    new *
    public double toMb(){

        return (value / 8) / mega;

    }

    new *
    public double toGb(){

        return (value / 8) / giga;
    }

    new *
    public double toTb() {

        return (value / 8) / tera;
    }
}
```


Main.java
MemorySize.java
new *
class Byte extends MemorySize{
new *
public Byte(double value) { super(value); }
new *
public double toBits(){
return value * 8;
}
new *
public double toByte(){
return value;
}
new *
public double toKb(){
return value / kilo;
}
new *
public double toMb(){
return value / mega;
}
new *
public double toGb(){
return value / giga;
}
new *
public double toTb(){
return value / tera;
}
}


Main.java
MemorySize.java
class KiloBytes extends MemorySize{
new *
public KiloBytes(double value) { super(value); }
new *
public double toBits(){
return value * mega * 8;
}
new *
public double toByte(){
return value * kilo;
}
new *
public double toKb(){
return value;
}
new *
public double toMb(){
return value / kilo;
}
new *
public double toGb(){
return value / mega;
}
new *
public double toTb(){
return value / giga;
}
}


Main.java
MemorySize.java
class MegaByte extends MemorySize{
new *
public MegaByte(double value) { super(value); }
new *
public double toBits(){
return value / mega * 8;
}
new *
public double toByte(){
return value * mega;
}
new *
public double toKb(){
return value * kilo;
}
new *
public double toMb(){
return value;
}
new *
public double toGb(){
return value / kilo;
}
new *
public double toTb(){
return value / mega;
}


Main.java
MemorySize.java
class GigaByte extends MemorySize{
new *
public GigaByte(double value) { super(value); }
new *
public double toBits(){
return value * giga * 8;
}
new *
public double toByte(){
return value * giga;
}
new *
public double toKb(){
return value * mega;
}
new *
public double toMb(){
return value * kilo;
}
new *
public double toGb(){
return value;
}
new *
public double toTb(){
return value / kilo;
}
}


Main.java
MemorySize.java
class TeraByte extends MemorySize{
new *
public TeraByte(double value) { super(value); }
new *
public double toBits(){
return value * tera * 8;
}
new *
public double toByte(){
return value * tera;
}
new *
public double toKb(){
return value * giga;
}
new *
public double toMb(){
return value * mega;
}
new *
public double toGb(){
return value * kilo;
}
new *
public double toTb(){
return value;
}
}

Category Class

-used to organize the options chosen by the user.

Main Function (Test Code for Category)

```java
public static void main(String[] args) {
    Scanner in = new Scanner(System.in);
    System.out.println("WELCOME TO UNICON");

    System.out.println("CATEGORIES:");
    System.out.println("\tAREA");
    System.out.println("\tANGLE");
    System.out.println("\tBITS");
    System.out.println("\tCURRENCY");
    System.out.println("\tLENGTH");
    System.out.println("\tMASS");
    System.out.println("\tTIME");
    System.out.println("\tTEMPERATURE");
    System.out.println("\tPRESSURE");
    System.out.println("\tVOLUME");
    System.out.print("\nChoose a category: ");
    String choice = in.next().toUpperCase();

    Category.chooseCategory(choice);

}
```

Main Function (Test Code for Category)

Category.java

```java
import java.util.Scanner;

public class Category {

    //list of categories available
    private final String[] category = {"AREA", "ANGLE", "BITS", "CURRENCY", "LENGTH", "MASS",
                                       "TIME", "TEMPERATURE", "PRESSURE", "VOLUME"};

    //choose conversion category based on user choice
    public static void chooseCategory(String category){
        switch (category) {
            case "AREA" -> Area.convert();
            case "ANGLE" -> Angle.convert();
            case "BITS" -> Bits.convert();
            case "CURRENCY" -> Currency.convert();
            case "LENGTH" -> Length.convert();
            case "MASS" -> Mass.convert();
            case "TIME" -> Time.convert();
            case "Temperature" -> Temperature.convert();
            case "PRESSURE" -> Pressure.convert();
            case "VOLUME" -> Volume.convert();
        }
    }
}
```